Information Management software

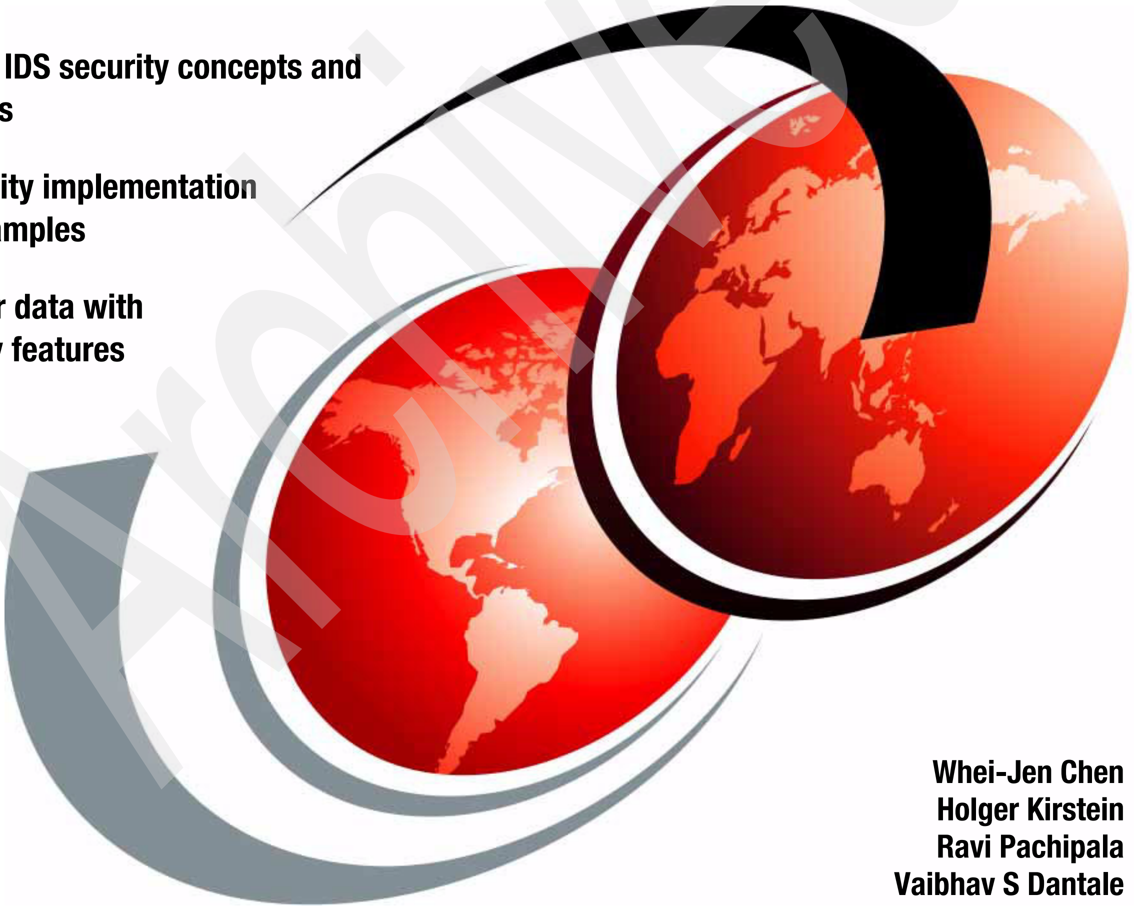
IBM

Security and Compliance Solutions for IBM Informix Dynamic Server

Understand IDS security concepts and technologies

Learn security implementation through examples

Protect your data with IDS security features



Whei-Jen Chen
Holger Kirstein
Ravi Pachipala
Vaibhav S Dantale

Redbooks

ibm.com/redbooks

International Technical Support Organization

Security and Compliance Solutions for IBM Informix Dynamic Server

March 2008

SG24-7556-00

Note: Before using this information and the product it supports, read the information in "Notices" on page vii.

First Edition (March 2008)

This edition applies to IBM Informix Dynamic Server Version 11.

Contents

Notices

Trademarks

Preface

In this IBM® Redbooks® publication, we discuss, in detail, the security features available in IBM Informix® Database Server (IDS). These enriched IDS security features provide you with the capability to protect your data and comply with regulatory requirements.

We discuss how IDS integrates with operating system security functions for user authentication and user permissions. Role separation divides the security duty among administrators. Auditing enables the database server to log sensitive operations performed by users and administrators for analysis and identifying system misuses.

Discretionary access control (DAC) is the primary access control mechanism that enables access to SQL objects using privileges and roles. Using label-based access control (LBAC), you can control read and write access of users to individual rows and columns at the table level. We then discuss how to secure server-server and server-client communication in an IDS environment, as well as address the security issues for backup and restore.

The team that wrote this book

This book was produced by a team of specialists from around the world working at the International Technical Support Organization, San Jose Center.

Whei-Jen Chen is a Project Leader at the International Technical Support Organization, San Jose Center. She has extensive experience in application development, database design and modeling, and DB2® system administration. Whei-Jen is an IBM Certified Solutions Expert in Database Administration and Application Development, as well as an IBM Certified IT Specialist.

Holger Kirstein is a Resolution Team Engineer with the European Informix support team. He joined the Informix support team in 1996 and has over 15 years of experience in application development and support for Informix database servers and Informix clients. He holds a Master of Applied Computer Science from Technische Universität, Dresden.

Ravi Pachipala has been a member of the IDS R&D security team since June 2006. Earlier in his career, Ravi was a member of Informix R&D teams for security and XPS, before IBM acquired Informix. Ravi holds a Master of

Technology degree in computer science from the Indian Institute of Technology, Kharagpur.

Vaibhav S Dantale is s Staff Software Engineer in IBM Information Management Division, in Bangalore, India. After joining IBM Informix in 2004, he worked on several IDS components. In recent years, his primary focus has been on IDS security features. Before joining IBM, he worked on a banking application providing technical support and developing new modules in which the Informix server products Informix Standard Engine (SE) and Informix Dynamic Server (IDS) were the primary database servers used. During later years of his career before joining IBM, he worked as a Data Mart Developer for one of the world's largest European organizations, dealing in airfreight and logistics, where he explored Informix Extended Parallel Server (XPS). Vaibhav holds a bachelor's degree in Computer Technology from Nagpur University, India.

Figure 1 Left to right: Holger, Ravi, and Vaibhav

Acknowledgements

Thanks to the following people for their contributions to this project:

Jonathan Leffler
David Desautels
Manoj Mohan
Joaquim Zuzarte
IBM Software Group, Informix Dynamic Server Development, USA

Emma Jacobs, Sangam Racherla, Deanna Polm
International Technical Support Organization

Become a published author

Join us for a two- to six-week residency program! Help write a book dealing with specific products or solutions, while getting hands-on experience with leading-edge technologies. You will have the opportunity to team with IBM technical professionals, Business Partners, and Clients.

Your efforts help increase product acceptance and customer satisfaction. As a bonus, you develop a network of contacts in IBM development labs, and increase your productivity and marketability.

Find out more about the residency program, browse the residency index, and apply online at:

ibm.com/redbooks/residencies.html

Comments welcome

Your comments are important to us!

We want our books to be as helpful as possible. Send us your comments about this book or other IBM Redbooks in one of the following ways:

- Use the online **Contact us** review Redbooks form found at:

  ibm.com/redbooks

- Send your comments in an e-mail to:

  redbooks@us.ibm.com

- Mail your comments to:

  IBM Corporation, International Technical Support Organization
  Dept. HYTD Mail Station P099
  2455 South Road
  Poughkeepsie, NY 12601-5400

1

Technology overview

In this chapter we provide an overview of the security features available in IBM Informix Database Server (IDS). We cover the following topics:

- Regulatory compliance overview
- IBM Informix Dynamic Server overview
- Security
  - OS security
  - Network security
  - IDS database security

This introduction should help you understand the IDS security features and examples further explained in later chapters.

1.1 Regulatory compliance overview

Due to the ubiquitousness of data and multi-user systems, data security has become a major concern nowadays in all sectors of business industries.

To secure data, each industry has regulatory compliances in place for all enterprises to comply with. Some of the standards are:

- Focus on consumer.
  - PCI Data Security Standard: This standard is a collaborative effort between major credit card companies and is designed to protect customers' personal information.
  - California Senate Bill 1386: This amendment requires that companies provide consumers with notices of security breaches involving unencrypted personal information.
- Focus on health.

  Health Insurance Portability and Accountability Act (HIPAA): This act is designed to protect all forms of personal health information by defending patients' rights to have their health information kept private.
- Focus on financial industry.

  Gramm-Leach-Bliley Act (GLBA): This act requires that financial institutions comply with the privacy provisions that mandate controls over customers' nonpublic personal information (NPI) with respect to usage, protection, and distribution.
- Focus on all industries.

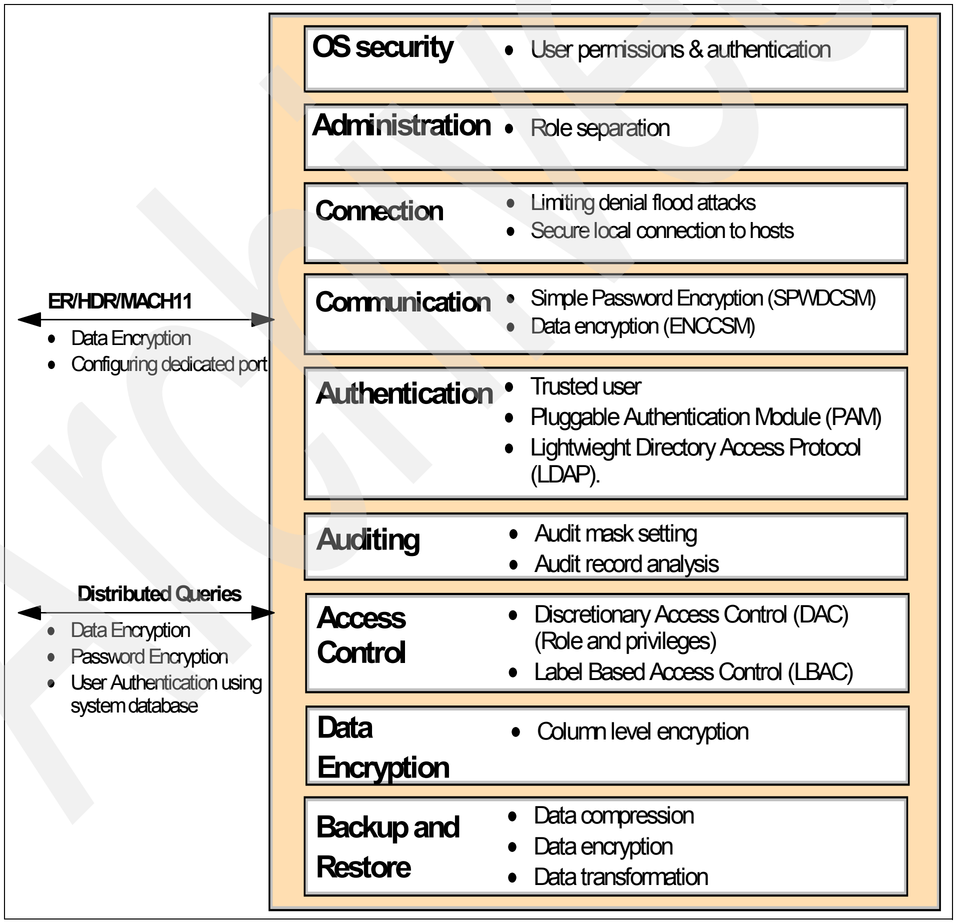
  Sarbanes-Oxley Act: The redesigned federal regulation of public company corporate governance and reporting obligations by demanding accountability and assurance of financial reporting by executives, auditors, securities analysts, and legal counsel.

1.2 IBM Informix Dynamic Server overview

IBM Informix Dynamic Server (IDS) is a multithreaded Relational Database Management System (RDBMS) that is widely used in Online Transaction Processing (OLTP) and Decision Support System (DSS) applications. IDS enables high-performance data processing and provides failover and replication capabilities.

IDS provides all aspects of data security, which includes authentication before access, access control after authentication, and confidentiality after access. IDS supports the Pluggable Authentication Module (PAM) mechanism for authentication, network, and data encryption mechanisms to ensure data integrity and confidentiality. Role separation allows the user to protect the administrator utility from unauthorized users. Features like label-based access control (LBAC) and discretionary access control (DAC) allow the user to imply security checks at table and column level. IDS provides data compression, data transformation, and data encryption mechanisms for offline data to ensure the security of data that is out of the control of IDS. Audit features allow the user to monitor the user activities in order to discover any malicious activity performed by an authorized user.

Figure 1-1 illustrates an overview of the security features offered by IDS.



OS security
• User permissions & authentication
Administration
• Role separation
Connection
• Limiting denial flood attacks
• Secure local connection to hosts
Communication
• Simple Password Encryption (SPWDCSM)
• Data encryption (ENCCSM)
Authentication
• Trusted user
• Pluggable Authentication Module (PAM)
• Lightwieght Directory Access Protocol (LDAP).
Auditing
• Audit mask setting
• Audit record analysis
Access Control
• Discretionary Access Control (DAC) (Role and privileges)
• Label Based Access Control (LBAC)
Data Encryption
• Column level encryption
Backup and Restore
• Data compression
• Data encryption
• Data transformation
ER/HDR/MACH11
• Data Encryption
• Configuring dedicated port
Distributed Queries
• Data Encryption
• Password Encryption
• User Authentication using system database

Figure 1-1 Overview of security features offered by IDS

1.3 Security

Security can be defined as *ensuring that no unauthorized user is accessing or altering your data*. You can ensure your data security by providing access permissions only to authorized users, auditing the events, and encrypting the data while transmitting over the network. You can establish security checks at different levels starting from login to the table column access.

In this section, we give brief introductions of all security features provided by IDS and some security tools that can be used in conjunction with IDS.

1.3.1 OS security

OS-level security starts from *authenticating* users by verifying their user IDs and passwords. Only authenticated users are allowed to enter the system. Once authenticated, *user permissions* enforces the access policies such as what directories or files are allowed to be accessed by the user.

IDS uses the trusted user mechanism provided by OS for network communication and the pluggable authentication module for the user-defined authentication mechanism.

User authentication

In simple words, authentication can be defined as the mechanism of verifying whether the requesting user is a valid user.

IDS supports the following user authentication methods:

- Trusted user

  If the client and the server are different hosts, both hosts should have trusted relationships to allow the client to access the database server. You can establish a trusted relationship between remote hosts and users by configuring the hosts.equiv file. You can also allow access to specific users only by setting the .rhosts file in a specific user's home directory. This is a traditional mechanism developed by Berkley Systems Division (BSD) to authenticate users on the system.

- Pluggable authentication module (PAM)

  A traditional login and password is no longer sufficient for authenticating a user. A pluggable authentication module is a well-defined framework for supporting different authentication modules. A PAM module determines whether a simple password is sufficient or other challenges are required to authenticate users. IDS supports the PAM framework. You can develop and implement different authentication modules to have your own authentication mechanism access the IDS server.

- Lightweight Directory Access Protocol

  Windows® does not support the PAM framework, but you can use the Lightweight Directory Access Protocol (LDAP) authentication module in place of the PAM module on Windows. Configuration of LDAP is the same as of PAM.

User permissions

IDS uses a user permission facility provided by OS to protect IDS utilities, IDS configuration files, and the output files generated by IDS. User *Informix* is a super user for IDS and can perform all activities related to IDS. However, other users are required to have specific permission to perform specific activity. IDS utilities and directories are secured by default. The server installation process sets the proper permissions for all IDS directories and utilities.

Only the user with database server administrator (dbsa) permissions can start and stop the IDS engine. IDS also protects other utilities and directories by setting its permissions so that only dbsa, aao, dbssao, or informix users can execute them. Permission of directory INFORMIXDIR is set to 755 and owner:group to informix:informix so that no user other than Informix can alter its contents.

Before starting the server, the database server utilities check for the valid permission of the following objects:

- The $INFORMIXDIR directory and some subdirectories.
- The ONCONFIG file must belong to the dbsa group and the file must not have write permissions to others.
- The sqlhosts file should have the owner as *Informix* and the group as *Informix* or dbsa user. Under default configuration, this file resides under $INFORMIXDIR/etc. Other users should not have write permissions to this file. If the INFORMIXSQLHOSTS variable is defined, then the owner and group are not checked. However, others should not have write permission.

Most of the IDS utilities run as dbsa users that belongs to Informix group.

1.3.2 Network security

In today's network-centric business model, network security has become an integral part of IDS database management.

Network security starts from user authentication. Once a user is authenticated, a *firewall* enforces security policies such as which user can access which service. If user has all permissions to access service, communication over the network starts. But there is the threat of hacking or tampering with data over a network, hence, secured communication has the utmost importance. OS provides different utilities such as remsh, rsh, and ssh for executing commands on a remote machine. ssh is considered the most secured utility for remote command execution or remote login.

ssh

Secure shell (ssh) is used to log into a remote machine and execute a command on it. It provides secured encrypted communication between two untrusted machines over an unsecured network. You can use ssh for transferring data or executing any Informix command on a remote machine to ensure the security of data transferred over the network.

firewall

A firewall is software configured to allow or deny access to users. It serves the purpose of protecting a host, network, database, or application from any unauthorized access request over the network. You can secure the database server by putting it behind a firewall.

These utilities indirectly protect IDS and ensure secured data transfer over a network.

1.3.3 IBM IDS database security topics overview

This section gives a high-level description of the different security features offered by IDS. It contains an overview of security features available for database administration, server-server connectivity, client-server connectivity, auditing, and SQL-level security.

Administration

Database administration mainly comprises activities like RDBMS installation, configuration, transaction monitoring, data backup, and recovery. From a security perspective, configuration and monitoring are the major activities. It is not a good practice to have a single person perform all of these activities. Hence, IDS provides a feature called *role separation*. With role separation enabled, you can assign different roles to different users. Pre-defined roles in IDS are:

- Database system administrator (dbsa), who maintains and tunes the database server
- Database system security officer (dbsso), who sets and maintains audit masks
- Audit analysis officer (aao), who analyzes audit reports

You can assign these different roles to different users during server installation.

Security for loading external module

IDS provides a configuration parameter DB_LIBRARY_PATH for controlling loading of an external module. If it is set, the database server allows loading of external modules from only defined directories.

The value of DB_LIBRARY_PATH should be the absolute path of the directory. If you want to define multiple directories, use commas as separators. For example:

```
DB_LIBRARY_PATH $INFORMIXDIR/extend, $INFORMIXDIR/udr
```

Client-server connection and authentication

IDS is based on a client-server architecture. A client can be a local or a remote user. A local user can connect to the database server using shared memory or the TCP/IP protocol, whereas a remote user can connect using the TCP/IP protocol. IDS provides various solutions to prevent unauthorized access to a database server to make your database server more secure.

For example:

- Support for pluggable authentication module (PAM)
- Support for LDAP authentication module for Windows
- Support for password and data encryption using CSM modules
- Provides $INFORMIXDIR/dbssodir/seccfg to restrict non-listed users from accessing the database server
- Provides the SECURITY_LOCALCONNECTION configuration parameter
- Limits denial-of-service (DOS) flood attacks by setting MAX_INCOMPLETE_CONNECTIONS

Most of above solutions can be used in conjunction with each other.

Server-server connection and authentication

IDS supports communication between servers on the same machine as well as over the network. Distributed queries, enterprise replication, high data availability replication, and shared disk are a few of the features that uses this functionality.

Distributed queries

The IDS server allows you to query data located in more than one database of different versions of IDS. Different databases can be on same host or on remote hosts. This feature is called *distributed queries*. Remote connections are authenticated using an r-command system devised by Berkeley System Division. Whenever there is a request for connection from the remote host, a daemon running on the local system checks the hosts.equiv file or the .rhosts file in the user's home directory. If the entry of the trusted host is present, the requesting remote host is treated as a trusted host and a database connection is allowed.

If PAM is set on the servers, distributed queries do not work. Distributed connections cannot respond to challenges because when a distributed connection can be made cannot be predicted. Authentication on the remote servers must be done within the database.

In such a scenario, for each remote user to establish a user authentication mechanism, the system administrator has to enter details of authorized users in the sysauth table in the sysuser database.

Data replication

Data replication refers to the process of representing database objects at more than one distinct site. IDS data replication capabilities provide several secondary server configuration options.

Table 1-1 shows the available data replicating secondary servers.

Table 1-1 Data replication secondary server types

Server type	Description
High availability data replication (HDR) secondary server	Provides synchronous data replication for the dynamic server. Use an HDR secondary server if you require a hot standby. Configuring an HDR secondary server provides a way to maintain a backup copy of the entire database server that applications can access quickly in the event of a catastrophic failure of the primary server.
Shared-disk (SD) secondary server	A server that shares disk space with a primary server. The primary server has write access to a disk or disk array, while all SD secondary servers have read-only access. An SD secondary server does not maintain a copy of the physical database on its own disk space; rather, it shares disks with the primary server.

Server type	Description
Remote standalone (RS) secondary server	A server that is updated asynchronously from the primary server. RS secondary servers can be geographically distant from the primary server, serving as remote back-up servers in disaster-recovery scenarios. Each RS secondary server maintains a complete copy of the database, with updates transmitted asynchronously from the primary server over secure network connections.

IDS provides different configuration parameters for ensuring authenticated access to data and secured transmission of data over the network:

- sqlhosts entry

  You can configure the sqlhosts file to set a particular entry only for ER or HDR transactions by adding s=6. This allow you to restrict any process that tries to use that connection for regular SQL.

  ```
  ol_myserv onsoctcp aix_mach online s=6
  ```

- ENCRYPT_CDR

  You can set configuration parameter ENCRYPT_CDR so that ER connections use the encryption mechanism. This ensures that the data transferred over the network is encrypted and secured.

  ```
  ENCRYPT_CDR [012]
  ```

0	No encryption
1	ENCRYPT_SMX
2	Must encrypt

- ENCRYPT_HDR

  You can set the configuration parameter ENCRYPT_HDR so that HDR connections uses the encryption mechanism. This ensures that the data transferred over the network is encrypted and secured.

  ```
  ENCRYPT_HDR [01]
  ```

0	No encryption
1	Must encrypt

- ENCRYPT_SMX

  Server Multiplexer Group (SMX) is a communications interface that supports encrypted multiplexed network connections between servers in high availability environments. SMX provides a reliable, secure, high-performance communication mechanism between database server instances. You can set ENCRYPT_SMX to define the level of encryption for high availability.

  ```
  ENCRYPT_SMX[012]
  ```

1	Encryption is used for SMX transactions only when the database server being connected to also supports encryption.
2	Only connections to encrypted database servers are allowed.
0	Disables encryption between servers.

- ENCRYPT_SWITCH

  You can set configuration parameter ENCRYPT_SWITCH to define the frequency at which ciphers or secret keys should be renegotiated.

  ```
  ENCRYPT_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

- ENCRYPT_CIPHER

  You can set the ENCRYPT_CIPHER configuration parameter to define which ciphers and modes can be used by the current database session.

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

  – all

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

    ```
    ENCRYPT_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:

    ```
    ENCRYPT_CIPHERS allbut<bf1,bf2>
    ```

  – cipher:mode

    Specifies the ciphers and modes. Separate cipher-mode pairs with a comma.

For example:

```
ENCRYPT_CIPHERS des3:cbc,des3:ofb
```

- ENCRYPT_MAC

  You can set the ENCRYPT_MAC configuration parameter to control the level of message authentication code generation. This parameter is used for enterprise replication and high-availability data replication only. One or more of the following options can be set as the value of this parameter:

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

- ENCRYPT_MACFILE

  You can set the ENCRYPT_MACFILE configuration parameter specifying a list of the full path names of MAC key files to be used for enterprise replication and high-availability data replication only. For example:

  ```
  ENCRYPT_MACFILE /usr/local/bin/mac1.dat,
  /usr/local/bin/mac2.dat,builtin
  ```

  All or some of the above-mentioned configuration parameters can be used in best-suited combination for your desired data security.

Auditing

Auditing creates a record of selected activities that users perform. IDS conforms to all regulatory compliances and ensures that all necessary details for auditing purposes are provides. IDS provides two main utilities, *onaudit* and *onshowaudit*, for auditing purposes. The onaudit utility is used to set the masks that specify the activities to be logged in an audit trail. You can set masks for a particular user as well. The audit trail report generated by IDS is in simple text format. It will contains audit trails of all defined masks for all users. The onshowaudit utility is used to read this audit trail report. To make reading audit trail report easy you can use some of the options with the onshowaudit utility, which filters out unnecessary information from the audit trail report.

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 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

The dbsso user can set different audit masks using the *onaudit* utility to get the desired transactions to be audited. An aao user can only analyze audit records using the *onshowaudit* utility. A dbsso user cannot analyze audit records. The user Informix, being a super user, can perform both the roles.

Database security with SQL

IDS provides some SQL syntaxes to enable security checks for particular types of SQL operations or data access. You can secure the database with these SQL options.

Discretionary access control (DAC)

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 DAC:

- User roles

  A role is a classification of tasks. You can use the CREATE ROLE statement to define a role.

- Setting permission to create a database

  You can use the DBCREATE_PERMISSION configuration parameter to give permission to create a database to only specified users.

- Security for external routine

  IDS allows users to register external routines and execute them. Letting any user register an external routine without any security checks is the biggest threat for a database. IDS provides security checks for registering external routines or libraries. You can set IFX_EXTEND_ROLE to 1 to enable the security check and then only the users with extend privileges can register the external routine or library. *GRANT extend to username* and *REVOKE extend from username* are SQL statements to grant or revoke extend permissions.

- Database and table access privileges

  You can control database access by granting specific privileges to a specific user. Database access privileges are DBA, CONNECT, and RESOURCE. Table privileges are ALTER, DELETE, INDEX, INSERT, EXECUTE, SELECT, UPDATE, REFERENCES, and ALL.

- Creating a view

  You can create a view of selected columns of the table that you want to allow users to read. This way you can mask the secured columns of the table from users.

Label-based access control

This is a new feature added in IDS 11. This feature provides data security at a more granule level.

Label-based access control is an implementation of multi-level security (MLS) that enables you 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 IDS, you can configure the LBAC objects to meet your security requirements.

The LBAC objects includes

- 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 data. You can have one or more security policies on your system, depending on your organization's needs.

- 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 whether the user can have access. If the user was not granted any label, access is automatically blocked.

- 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.

Column-level encryption

The column-level encryption mechanism helps in storing sensitive data in an encrypted format. You can store sensitive information like an ATM pin number or a credit card password in encrypted format. Only users who can provide the secret password can decrypt the data and read it. IDS provides built-in encryption functions, ENCRYPT_AES() and ENCRYPT_TDES(), to encrypt data in columns. You can encrypt columns containing only the following data types or smart large object data types:

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

Backup and restore

IDS provides *ontape* and *onbar* utilities to back up and restore data. The onbar uses storage manager for backup and restores the data, whereas ontape does not use a storage manager. Informix Storage Manager (ISM) is the default IDS storage manager.

Ontape can store data on magnetic tape, file system, and disk, whereas onbar can store data on all storage devices supported by the storage manager configured for backup and restore.

IDS provides you with the capability to store and restore data in encrypted format or using some external filter logic so that the data cannot be restored and read by anybody. This is done through setting the configuration parameters and the environment variable. This ensures the security of backed up data in media that is out of control of IDS.

The data backup security related configuration parameters and environment variable are:

- ISM_COMPRESSION

  If this environment variable is set to TRUE in the environment where the onbar process makes the request, the ISM server uses a data compression algorithm to store or retrieve the data specified in that request. If it is set to FALSE or is not present, the ISM server does not use compression.

- ISM_ENCRYPTION

  If this environment variable is set to TRUE or XOR in the environment where the onbar process makes the request, the ISM server uses encryption to store or retrieve the data specified in that request. If it is set to NONE or is not present, the ISM server does not use encryption.

- BACKUP_FILTER

  You can set this configuration parameter if you want to use external programs as filter plug-ins to transform data to a different format prior to a backup.

- RESTORE_FILTER

  You should set this configuration parameter to restore a data that is transformed using external programs as filter plug-ins prior to a backup.

2

Role separation

The default IDS database server installation endows the user *informix* with all duties and permissions that the database server can grant. This elevates the user *informix* to the same level as the root or administrator user on the operating system and requires special account protection.

To allow tight security and easy implementation, the IDS provides the capability to split the common database security tasks, like user access control or managing, and analyzing the audit. The ability to separate the duties also avoids the potential risk of multiple persons working with the Informix account.

Most of the time, security tasks are related to dealing with sensitive data that can only be handled by specially trained people in specific departments. The IDS role separation capability gives a company the ability to leverage the strength of various departments. Furthermore, removing the security tasks out of the daily duty of dbsa allows them to concentrate more on their core competencies like resource administration, tuning, and availability management.

In this chapter, we describe in detail the various roles that the database server has, how to set up the role separation, and what are the security focus points when using this feature.

2.1 User responsibilities in a database environment

From a security perspective, it makes sense to split the administration duties to different employees in a company to minimize the risks of misusing the database content. To support this duty separation, IDS has the following predefined roles:

- Database server administrator (dbsa)
  - User account responsible for configuring, tuning, and maintaining the server instances based on the local IDS database server installation
  - Duties include the startup and shutdown of the database server, disk space management, backup and restore, performance tuning, and troubleshooting
- Database system security officer (dbsso)

  Defines the audit masks for particular users working on the local database server instance
- Audit analysis officer (aao)

  Responsible for audit configuration and audit trail analysis
- Database administrator (dba)
  - Maintains a specific database in the local database server.
  - Permission was either automatically given to the database creator or granted by the database creator.
  - Does not necessarily include database server privileges
- Operating system administrator (osa)
  - OSA defines and maintains the local user accounts including the aao, dbsa, and dbsso groups.
  - Installs the IDS product, responsible for changing required group permissions for role separation.
  - Maintains the kernel parameter settings and resource limits such as files and memory.
- Users

  Local or remote user accounts that are used to run database-based applications.
- Privileged users

  IDS defines the informix and the root user as privileged users.

2.2 Enabling role separation on IDS

In a general IDS database server installation, there is no split of the default security roles. The privileged user *informix* assumes the duty on all the defined database tasks. The IDS database server provides several ways to set up the split, also referred as role separation. You can either enable the role separation while installing the product or make changes manually after the database server is installed. Applying the changes manually also supports changing the administration roles from one user group to another.

2.2.1 Define role separation during IDS server installation

In general, IDS provides three types of installation methods:

- Interactive, ASCII-based installation (available on UNIX®)
- Interactive GUI-based installation (on UNIX and Windows)
- Silent, option file based installation (on UNIX and Windows)

You can enable the role separation on all three types of IDS installation. In this section we discuss the different ways to enable the roles in detail.

ASCII-based interactive installation

On UNIX systems, the IDS distribution provides the ASCII-based interactive installation as the default installation method. After untaring the distribution file of the IDS bundle, you are required to call the ids_install script. This script is situated in the top directory, where you have unpacked the tar file. Calling the ids_install without the -gui option provides you with an ASCII-based interactive interface for the installation of the product. In case you specify a role separation for the current IDS installation, you are asked to specify all additional required user groups. A sample specification of the role separation related questions is shown in Example 2-1.

Example 2-1 Installation with role separation by the ASCII-based interface

```
Select the products you would like to install:

   To select/deselect a product or to change its setup type, type its number:

   Product                                           Setup Type
   ------------------------------------------------- ------------------------
    1. [ ] IBM Informix IConnect
    2. [x] IBM Informix Client-SDK                    Typical
    3. [x] IBM Informix Dynamic Server                Typical
    4. [x] IBM Informix JDBC Driver Version 3.10
   Other options:

    0. Continue installing

   Enter command [0]

Press 1 for Next, 2 for Previous, 3 to Cancel or 4 to Redisplay [1]

Do you want to enable role separation?

[ ] 1 - Yes
[X] 2 - No

To select an item enter its number, or 0 when you are finished: [0] 1

Enter 0 to continue or 1 to make another selection: [0]

Press 1 for Next, 2 for Previous, 3 to Cancel or 4 to Redisplay [1] 1

Role Separation Properties

Database System Security Officer Group [informix] idx_dsso
Auditing Analysis Officer Group [informix] idx_aao
Users Group (Leave blank to allow all groups) [] ids_users

Press 1 for Next, 2 for Previous, 3 to Cancel or 4 to Redisplay [1] 1

Would you like to create an IDS demo instance?

[ ] 1 - Create Demo
[X] 2 - Do Not Create Demo

To select an item enter its number, or 0 when you are finished: [0]
```

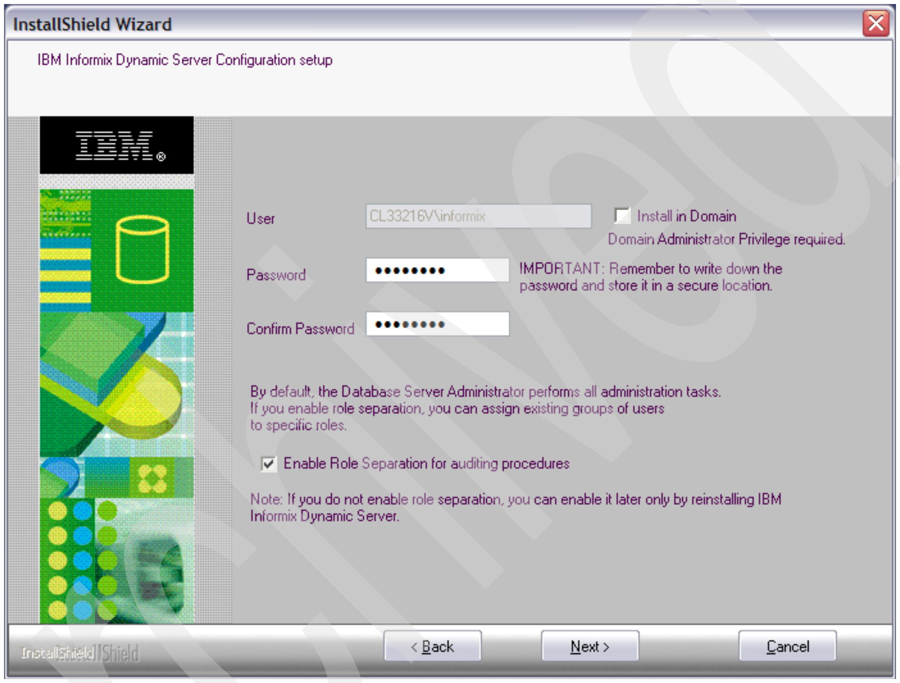
For a verification of the success and a description of how to distinguish your role separation settings from the default installation, refer to the discussion about the directory permissions in 2.2.2, “Define role separation manually on UNIX” on page 26.

GUI-based IDS installation

A GUI interface provides a more user-friendly installation process for the administration users. The GUI interface for the IDS installation on UNIX is based on Java™. On Windows, the IDS installation used the standard Windows Installer interface.

Windows

The GUI-based installation is the default IDS installation method on Windows. In order to configure role separation with the interactive GUI installation method, you have to choose the custom installation path. The role separation is disabled in the typical installation. During the custom installation, if you enable role separation, additional panels are presented, providing the fields for configuring the additional users and groups need for aao, dbsa, and dbsso. See Figure 2-1 and Figure 2-2 on page 22.


InstallShield Wizard

IBM Informix Dynamic Server Configuration setup

User: CL33216V\informix

[ ] Install in Domain

Domain Administrator Privilege required.

Password: ••••••••

IMPORTANT: Remember to write down the password and store it in a secure location.

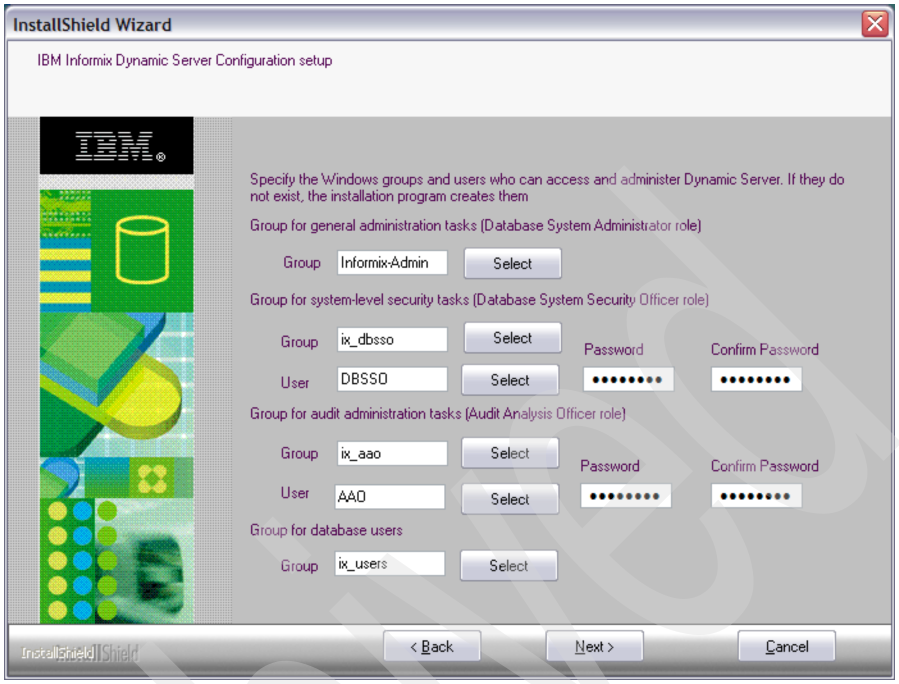
Confirm Password: ••••••••

By default, the Database Server Administrator performs all administration tasks. If you enable role separation, you can assign existing groups of users to specific roles.

[✓] Enable Role Separation for auditing procedures

Note: If you do not enable role separation, you can enable it later only by reinstalling IBM Informix Dynamic Server.

Figure 2-1 Defining roles for role separation on Windows



Specify the Windows groups and users who can access and administer Dynamic Server. If they do not exist, the installation program creates them

Group for general administration tasks (Database System Administrator role)

Group	Informix-Admin	Select		

Group for system-level security tasks (Database System Security Officer role)

			Password	Confirm Password
Group	ix_dbsso	Select		
User	DBSSO	Select	••••••••	••••••••

Group for audit administration tasks (Audit Analysis Officer role)

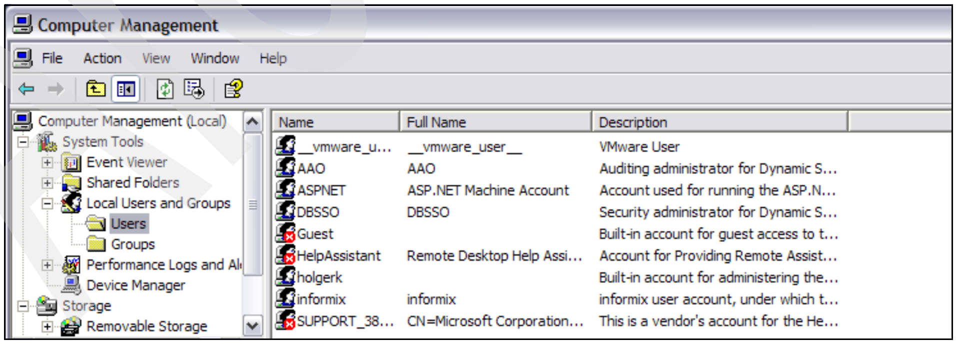
			Password	Confirm Password
Group	ix_aao	Select		
User	AAO	Select	••••••••	••••••••

Group for database users

Group	ix_users	Select

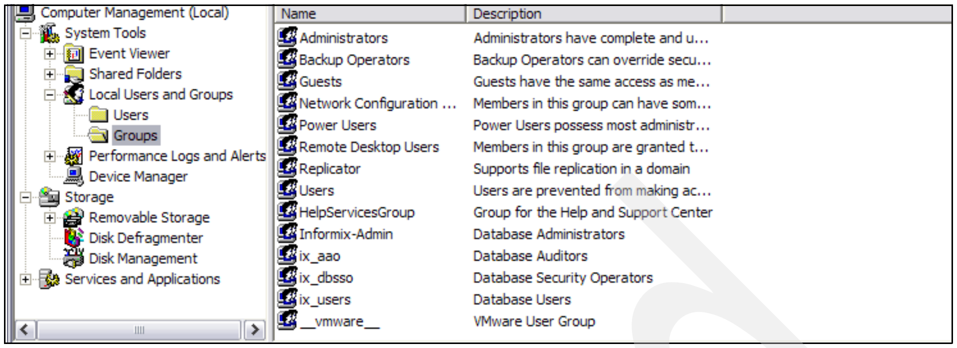
Figure 2-2 Panel for group and user definition required by role separation

After the installation is complete, in addition to the informix user account and the Informix-Admin group, the specified groups and users configured for the dbsso, aao, and dbsa roles are created. You can see them in the Computer Management in the Control Panel, as shown Figure 2-3 and Figure 2-4 on page 23.



Name	Full Name	Description
__vmware_u...	__vmware_user__	VMware User
AAO	AAO	Auditing administrator for Dynamic S...
ASPNET	ASP.NET Machine Account	Account used for running the ASP.N...
DBSSO	DBSSO	Security administrator for Dynamic S...
Guest		Built-in account for guest access to t...
HelpAssistant	Remote Desktop Help Assi...	Account for Providing Remote Assist...
holgerk		Built-in account for administering the...
informix	informix	informix user account, under which t...
SUPPORT_38...	CN=Microsoft Corporation...	This is a vendor's account for the He...

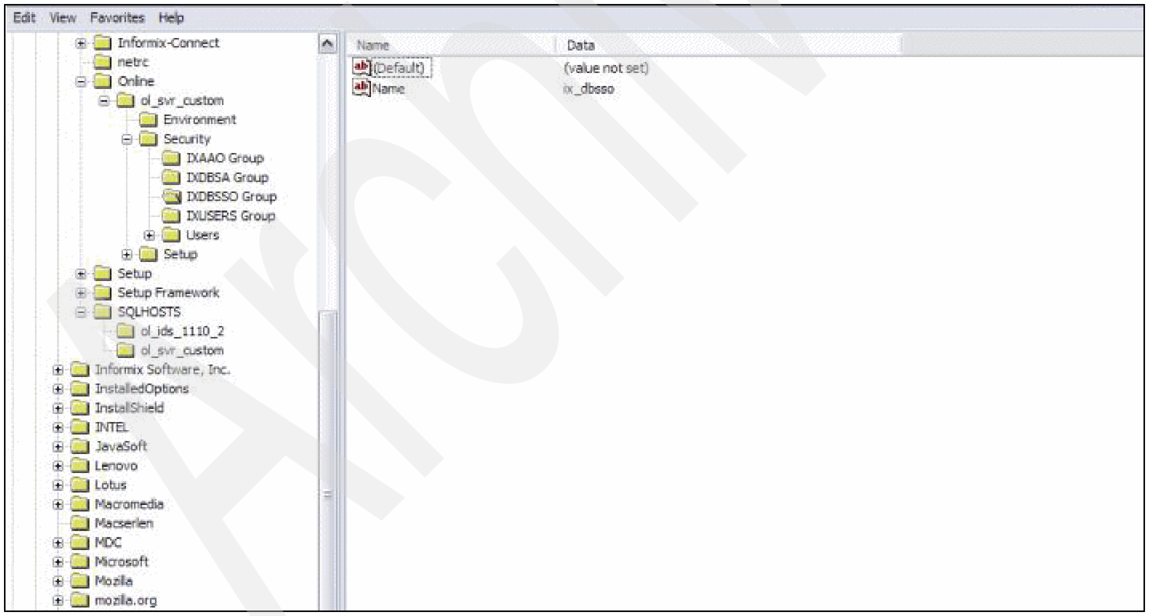
Figure 2-3 Users created on the local Windows machine after IDS installation with roles



Name	Description
Administrators	Administrators have complete and u…
Backup Operators	Backup Operators can override secu…
Guests	Guests have the same access as me…
Network Configuration …	Members in this group can have som…
Power Users	Power Users possess most administr…
Remote Desktop Users	Members in this group are granted t…
Replicator	Supports file replication in a domain
Users	Users are prevented from making ac…
HelpServicesGroup	Group for the Help and Support Center
Informix-Admin	Database Administrators
ix_aao	Database Auditors
ix_dbsso	Database Security Operators
ix_users	Database Users
__vmware__	VMware User Group

Figure 2-4 New groups created by the IDS installation on Windows with role separation

Different from the IDS installation on UNIX, the settings for the role separation are not based on file system permissions. There are registry entries for each database server installation. With the registry editor, you can find them under **HKEY_LOCAL_MACHINE** → **SOFTWARE-Informix** → **Online** → **<instancename** → **Security**. Figure 2-5 shows the snapshot of possible role separation configuration.



Name	Data
(Default)	(value not set)
Name	ix_dbsso

Figure 2-5 Registry settings for IXDBSSO after an IDS installation with role separation

UNIX

You can invoke the GUI interface by specifying the -gui option in the ids_install call. The specification of the appropriate parameters like the aao and dbsso groups can be done on a separate panel after enabling the role separation in the default installation process. The UNIX installation does not offer different types of installation like the typical installation, as Windows does. This allows you to enable the role separation all the time during the installation process.

There is another difference between the installation on Windows and UNIX. You are required to create manually the groups on UNIX regardless of which type of installation you use. On Windows, the specified user and groups are automatically created by the installation routine.

Silent installation of the IDS server with role separation

In addition to the interactive installation, IDS also provides silent batch installation that is a script-based, unattended installation on both platforms. For this installation type, you have to maintain an option file for the installation procedure. The option file contains the settings of all the parameters needed for a successful installation.

UNIX

On UNIX, the IDS distribution contains a bundle.ini file in the base directory. In order to use the silent installation, you have to apply your changes to the appropriate parameters to this file before you start ids_install. Example 2-2 shows sample settings for the role separation-related parameter in the bundle.ini file.

Example 2-2 bundle.ini settings for role separation

```
#Where to find the bundle.ini file
# ls -al $INFORMIXDIR
total 66430
drwxr-xr-x  25 informix informix    1024 Oct 24 15:23 .
drwxr-xr-x  20 informix informix     512 Oct 11 12:24 ..
drwxrwxr-x   2 informix informix     512 Feb 14  2007 aaodir
drwxr-xr-x   2 informix informix    2048 Sep  6 13:46 bin
-rw-r--r--   1 root     other      32496 Jun 19 07:25 bundle.ini

#Which parameter have to be changed
# grep rolesep bundle.ini
-SW SERVER/IIF.jar rolesepenable.roleSep="on"
-SW SERVER/IIF.jar rolesep.dbsso_g="dbsso"
-SW SERVER/IIF.jar rolesep.aao_g="aao"
-SW SERVER/IIF.jar rolesep.user_g="ids_users"
```

```
#How to run the silent installation on UNIX
# ids_install  -silent  -options bundle.ini

#Verification
#ls -al
insgesamt 22
drwxr-xr-x  20 informix informix  768 2007-10-24 15:46 .
drwxrwxrwx  11 informix informix  464 2007-10-24 15:43 ..
drwxrwxr-x   2 informix aao       104 2007-10-24 15:44 aaodir
drwxr-xr-x   2 informix informix 2632 2007-10-24 15:44 bin
drwxrwxr-x   2 informix dbsso     104 2007-10-24 15:44 dbssodir
drwxr-xr-x   4 informix informix   96 2007-10-24 15:44 demo
drwxr-xr-x   3 informix informix   72 2007-10-24 15:44 doc
...

# cat $INFORMIXDIR/dbsso/seccfg
IXUSERS=ids_users
```

Windows

On Windows, there are two ways to generate the content of the option file required for a silent batch installation:

- Using server.ini file

  The server.ini file is located on the installation medium. Copy the original content from the distributed file into a new file and apply all the necessary changes. The changes are mostly related to passwords, machine names, and so on. The tokens that have to be replaced are in <>.

  Example 2-3 shows the content from the sample server.ini, which has to be customized. Replace the bolded text with your required settings.

  Example 2-3 Tokens in the sample server.ini that have to be replaced

  ```
  type server.ini
  [{0F0B0E22-611B-48D7-A6D0-138DCBE1354C}-Informix user setup-0]
  User=<Machine>\informix
  Install in Domain=0
  Password=<Informix Password>
  Confirm Password=<Confirm Password>
  Enable Role Separation=0
  Result=1
  [{0F0B0E22-611B-48D7-A6D0-138DCBE1354C}-Role separation user
  setup-0]
  DBSA Group Account=Informix-Admin
  ```

```
DBSSO Group Account=ix_dbsso
DBSSO User Account=DBSSO
DBSSO Password=<DBSSO Password>
DBSSO Confirm Password=<Confirm DBSSO Password>
AAO Group Account=ix_aao
AAO User Account=AAO
AAO Password=<AAO Password>
AAO Confirm Password=<Confirm AAO Password>
IDS Users Group Account=ix_users
```

- Using the setup.exe option

  Another way of generating a syntactically correct option file is the use of the reply option in the setup.exe. You can understand this as a master installation on a test machine. Your definitions related to install options, passwords, user settings, installation path, and server parameter are tracked in a file, which can be used as the template option file for all subsequent installations. The master installation is done in an interactive mode, but the reply file generated by this master installation can be used for subsequent installations in silent mode.

  Example 2-4 shows the steps generating an option file and how to use the file on a subsequent installation.

Example 2-4 Generate an option file with the reply option and install IDS silently

```
#Step 1 generate the template file silent.ini from a
masterinstallation
C:\temp\IIF>setup.exe -r -f1"c:\temp\iif\silent.ini"

#Step2 Apply Changes to the silent.ini file which are different
#Step3 Run a silent installation on a target machine
C:\temp\IIF>setup.exe -s -f1"c:\temp\iif\silent.ini"
-f2"c:\temp\iif\logfile.log"
```

2.2.2 Define role separation manually on UNIX

Role separation can be enabled in an already installed IDS database server. This way also provides the possibility to change already specified aao, dbsa, or dbsso groups. The changes are applied to the existing server instances after a reboot of the server.

Permissions on directories in $INFORMIXDIR

The groups for the aao, dbsa, or dbsso in the local IDS installation are specified by the group permissions of the appropriate directories in the IDS installation directory.

Example 2-5 shows the default settings on installation directories. The group owning the directories is in all cases the informix group. The dUties Of Aao, Dbsa, And Dbsso Are Executed by the users in the informix group.

Example 2-5 Default permission after installation in the $INFORMIXDIR directory

```
$ls -al $INFORMIXDIR
drwxrwxr-x  2 informix informix   128 2007-10-24 15:39 aaodir
drwxrwxr-x  2 informix informix   104 2007-10-24 18:20 dbssodir
drwxrwxr-x  3 informix informix  2688 2007-10-24 18:30 etc
```

To separate the roles, change the group owner of the directories of aao, dbsa, and dbsso. In our example, to make the relationship more obvious, we name the groups taking over the responsibilities for the specific tasks similar to the name of the task. After we changed the permissions in the way as shown in Example 2-6, the users belonging to the aao group act as the aao, the users in the dbsso group act as the dbsso, and the users in the dbsa group act as the server administrator.

Example 2-6 Permissions in the $INFORMIXDIR after applying the roles manually

```
drwxrwxr-x  2 informix aao       128 2007-10-24 15:39 aaodir
drwxrwxr-x  2 informix dbsso     104 2007-10-24 18:20 dbssodir
drwxrwxr-x  3 informix dbsa     2688 2007-10-24 18:30 etc
...
```

You can switch off the role separation by changing the group permissions of the appropriate directories back to the default settings of informix. After a restart of the server instances based on this installation, the role separation is disabled.

Verify your changes

After changing the permissions of the appropriate subdirectories in $INFORMIXDIR, restarting the database server is not required if you only want to check the settings for the aao and dbsso. If you want to use the new roles and the user blocking, you are required to restart the database server in order to apply your changes.

Example 2-7 and Example 2-8 demonstrate how to verify the settings for the aao and dbsso groups. Example 2-7 shows the behavior of the auditing utilities used by a user belonging to the aao group. A user in the aao group is able to start up the audit and examine the output generated by the audit protocol. However, only a member in the dbsso group can set up the audit masks to enable auditing for specific users and events.

Example 2-7 Permissions for a user belonging to the aao group

```
$ id
uid=1003(idx_aao) gid=102(aao) groups=101(users),102(aao)

#enable auditing
$ onaudit -l 1
Onaudit -- Audit Subsystem Configuration Utility
 Copyright IBM Corporation 1996, 2004 All rights reserved.
$

#try to set up an audit mask for a specific user -- failed
$ onaudit -m -u user99 -e +CRTAB
Onaudit -- Audit Subsystem Configuration Utility
 Copyright IBM Corporation 1996, 2004 All rights reserved.

Must be a DBSSO to execute this action.

#examine the audit files
$ onshowaudit

ONSHOWAUDIT Secure Audit Utility
INFORMIX-SQL Version 10.00.UC7
Copyright IBM Corporation 1996, 2004 All rights reserved.

Software Serial Number AAA#B000000
ONLN|2007-10-24
14:54:32.000|CL33216V|19046|on1110shm|user99|-388:CRTB:sysmaster:0:test
:user99:0:-
```

Example 2-8 shows that users belonging to the dbsso group are able to run the onaudit utility, but they are not able to examine the audit protocol file.

Example 2-8 Checking the permissions for the dbsso group

```
$id
uid=1000(idx_dbsso) gid=100(dbsso)
groups=100(dbsso),16(dialout),17(audio),33(video)
```

```
#Set up an audit mask for user99 -- CREATE TABLE shall be audited
$ onaudit -a -u user99 -e +CRTB
Onaudit -- Audit Subsystem Configuration Utility
 Copyright IBM Corporation 1996, 2004 All rights reserved.

#Verify audit event masks for users
$ onaudit -o -u user99
Onaudit -- Audit Subsystem Configuration Utility
 Copyright IBM Corporation 1996, 2004 All rights reserved.

user99    -       CRTB

#Try to examine the audit event -- failed.
$ onshowaudit

Must be an AAO to run this program.
```

In this section we looked at the audit utilities with a special focus on the specific user roles. This includes the discussion of which user is able to perform which tasks. More detailed information about the IDS auditing facility, including setup, maintaining audit masks, and examination of the audit protocol, is given in Chapter 3, “Auditing” on page 35.

2.2.3 Define role separation manually on Windows

On Windows, the maintenance for the role separation values is different from what is on UNIX. On UNIX, IDS maintains the groups for the different user roles by the permissions on the appropriate directories. On Windows, IDS maintains the values in the registry for each local configured database server instance. In order to change the current settings, you have to use a registry editor to change the values. The values belonging to the different security groups are stored in the registry under **HKEY_LOCAL_MACHINE** → **SOFTWARE-Informix** → **Online** → **<instancename** → **Security**.

Make sure that the groups you apply to the different roles exist. You can use the verification steps described in 2.2.2, "Define role separation manually on UNIX" on page 26, to verify your settings for a local user account. This account should belong to the new applied group for the appropriate role. You should try executing the onaudit and onshowaudit. In case your settings are wrong, the execution of the auditing utilities fail with the message similar to that shown in Example 2-9.

Example 2-9 User is neither in the aao nor in the dbsso group on Windows

```
C:\Program Files\IBM\IBM Informix Dynamic Server\11.10>onaudit -l 1
Onaudit -- Audit Subsystem Configuration Utility
 Copyright IBM Corporation 1996, 2004 All rights reserved.

Must be an AAO or DBSSO to run this program.

C:\Program Files\IBM\IBM Informix Dynamic Server\11.10>onshowaudit
Must be an AAO to run this program.
```

2.3 Blocking users from local database server access

There is another security feature that might be related, on the first view, to role separation, but can also be used on a IDS installation that does not have role separation enabled. The dbsa user can restrict the database server access to a specific group. All users who are not in this particular group or other groups defined for aao, dbsso, or the dbsa trying to access any data in an existing database are rejected at connection attempt on the database server.

Example 2-10 shows the message logged in the online.log file. The client receives an error -25571.

Example 2-10 Rejected access for an excluded database user

```
13:29:45  Checkpoint loguniq 4, logpos 0x1bb018, timestamp: 0xf0da

13:29:45  Maximum server connections 0
13:29:45  On-Line Mode
13:43:54  listener-thread: err = -25571: oserr = 0: errstr = user99:
Cannot create a user thread.
```

$INFORMIXDIR/dbssodir/seccfg

The database server access is controlled by the content of the $INFORMIXDIR/dbssodir/seccfg file. Example 2-11 shows the default content of seccfg. The asterisk (*) means that there is no restriction to a specific OS group.

Example 2-11 $INFORMIXDIR/dbssodir/seccfg default settings

```
$cat $INFORMIXDIR/dbssodir/seccfg
IXUSERS=*
```

To restrict the access to a specific group, change the IXUSERS value from * to the name of the group. You need to restart the database server to have this change take effect. Example 2-12 shows a possible restriction, allowing only the informix group access to the server.

Example 2-12 Restrict the access to the IDS server to the informix group

```
$cat $INFORMIXDIR/dbssodir/seccfg
IXUSERS=informix
```

Setup restrictions at installation time

Besides changing the seccfg file, you also can apply your specification at installation time. To specify the access for a specific group, you have to enable role separation during the installation process. If you do not want to use role separation, choose the informix group for dbsso and aao to keep the default behavior. Then specify the user group to whom the server access should be granted. Continue with the installation as usual.

Example 2-13 shows a possible scenario for the interactive ASCII-based installation.

Example 2-13 How to change the content of the seccfg file at installation time

```
Do you want to enable role separation?

[ ] 1 - Yes
[X] 2 - No

To select an item enter its number, or 0 when you are finished: [0] 1

Enter 0 to continue or 1 to make another selection: [0]

Press 1 for Next, 2 for Previous, 3 to Cancel or 4 to Redisplay [1] 1

Role Separation Properties

Database System Security Officer Group [informix] informix
Auditing Analysis Officer Group [informix] informix

Users Group (Leave blank to allow all groups) [] informix
```

```
Press 1 for Next, 2 for Previous, 3 to Cancel or 4 to Redisplay [1] 4
```

If you want to use the silent bundle installation, you have to change the bundle.ini file on UNIX and set the parameter as shown in Example 2-14.

Example 2-14 Silent bundle installation with user group access

```
#cat bundle.ini| grep rolesep
-SW SERVER/IIF.jar rolesepenable.roleSep="on"
-SW SERVER/IIF.jar rolesep.dbsso_g="informix"
-SW SERVER/IIF.jar rolesep.aao_g="informix"
-SW SERVER/IIF.jar rolesep.user_g="informix"
#ids_install  -silent  -options bundle.ini
```

Blocking informix user from accessing database server

In general, the informix user is a privileged user in the IDS database server environment. His privileges include accessing the server and the DBA permissions for any existing database in the server. You can exclude the informix user from accessing any data in the database server by setting the IXUSERS to a group that the informix user does not belong to. You also need to make sure that the informix user is not in any group defined for aao, dbsso, or dbsa.

Example 2-15 shows the settings of the $INFORMIXDIR directory on UNIX that would block the informix user from accessing the database server.

Example 2-15 Settings in order to block the informix user from a local IDS instance

```
drwxrwxr-x  2 informix ifxaao   128 2007-10-24 15:39 aaodir
drwxrwxr-x  2 informix ifxdbs   104 2007-10-24 18:20 dbssodir
drwxrwxr-x  3 informix ifxdbsa 2688 2007-10-24 18:30 etc

$cat $INFORMIXDIR/dbssodir/seccfg
IXUSERS=idsusers
```

Depending on the permissions on the executables in the $INFORMIXDIR/bin directory, the informix user could remain as a dbsa and be able to execute them.

You are not able to block the IDS database server access from the root user. By default, the root user is included in the aao, dbsa, and dbsso groups. This allows the root user to attach to the server all the time. The only way to restrict the root user is by revoking the connect permissions on the local databases from the root user. Be careful of the sysutils system database. Since root is, besides the informix user, the only user allowed to run backups, revoking the connect on sysutils breaks the ability to run for root user.

Troubleshooting

The IXUSERS entry in the seccfg file can only specify one group. Be careful when changing this file. Any setting that is not supported by the database server causes the server to abort at the next startup time. Example 2-16 shows two incorrect settings in the seccfg file and the error message that comes up when the dbsa attempts to bring up the IDS with oninit.

Example 2-16 Incorrect settings in seccfg file cause oninit to abort at startup time

```
# Two user groups with blank separated
$cat $INFORMIXDIR/dbsso/seccfg
IXUSERS=informix users

or

# Two user groups with “,” separated
$cat $INFORMIXDIR/dbsso/seccfg
IXUSERS=informix,users


$> oninit
Could not read $INFORMIXDIR/dbssodir/seccfg or go through
$INFORMIXDIR/aaodir correctly
```

File permissions for the seccfg file

Set the file permission for the seccfg file properly. Since the file controls the access of particular users to the local database server, the access to this file should only be allowed for the appropriate role. It should be either dbsso or dbsa.

Example 2-17 shows the file permission of the general installation.

Example 2-17 Default permissions for the seccfg

```
drwxrwxr-x   2 informix informix       256 Aug 23 15:43 .
drwxr-xr-x  33 informix informix      4096 Aug 23 16:08 ..
-rw-r--r--   1 informix informix       846 Jul 25 10:02 adtmasks.std
-rw-rw-r--   1 informix informix        10 Jul 25 10:02 seccfg
```

In case you specified a role separation at IDS installation time, the installation process changed the group permissions of this particular file to the dbsso group. When you manually change the dbsso directory group permissions in order to switch the dbsso group, you should also change the group of seccfg file.

Example 2-18 shows the appropriate file permissions if you set the dbsso group dbsso.

Example 2-18   File permissions on the seccfg file for IDS with role separation

```
drwxrwxr-x   2 informix dbsso         256 Aug 23 15:43 .
drwxr-xr-x  33 informix informix     4096 Aug 23 16:08 ..
-rw-r--r--   1 informix dbsso         846 Jul 25 10:02 adtmasks.std
-rw-rw----   1 informix dbsso          10 Jul 25 10:02 seccfg
```

3

Auditing

This chapter describes the auditing facility available in the Informix Dynamic Server. Auditing enables the database server to log sensitive operations performed by users and administrators. The audit administrator later analyzes the audit log for unauthorized or suspicious database activity.

In this chapter, we cover the audit setup and configuration, audit masks and events, and analysis of an audit trial.

3.1 Purpose of auditing

Auditing provide enormous potential to company executives and administrators to go back in time and analyze the audit trail to discover security breaches and violations. It is usually this potential that drives IT departments to seek this capability in DBMS products.

Auditing provides the following benefits:

- Active monitoring of the audit trail by the administrator provides evidence of security breaches or violations.
- Auditing is a preventive security measure, as it prevents intruders from taking control of the system lassies-faire for fear of leaving an audit trail.
- Auditing helps in monitoring the actions of privileged users of the system and makes them accountable for their actions.
- Turning on auditing in the system helps meet regulatory compliance requirements. In some cases, generating an audit trail is the most a company needs to do to adhere to compliance requirements.
- Auditing provides a record of actions taken by users and administrators, although an audit log cannot be used for transaction recovery purposes.
- Auditing provides control to the administrator about who to audit and when to turn auditing on or off.
- By archiving the audit trail, companies can go deep into the past and uncover and provide proof of damaging activities done by users or administrators.
- Auditing provides little administrative overhead. Once the audit setup is completed, the system takes over in logging the activities of users.
- An audit trail is a permanent record of the activities done by users. Company policies stipulate how long an audit trail is safely stored.

While auditing has many benefits, it comes with a cost. Turning on full fledged auditing in the system can slow down the system. By carefully controlling which operations to audit and whom to audit, the administrator improves the performance of the system.

3.2 The audit process

Figure 3-1 captures the audit process with IDS. As shown, the **onaudit** command is used to set up audit configuration. The **onaudit** command is also used to set up audit masks. The IDS generates the audit trail depending on the audit configuration and audit mask setup. The **onshowaudit** command reads the audit trail and pipes it to a filtering and conversion utility, and loads the records into the database. The aao generates queries against the data loaded and performs analysis. We discuss each of these steps in subsequent sections.

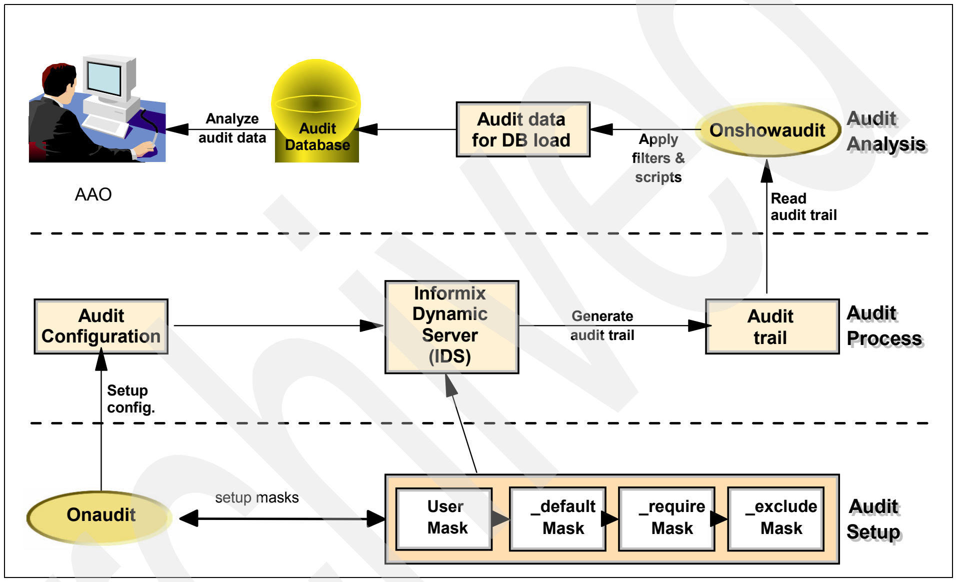


Figure 3-1 IDS audit process

3.3 Audit configuration

Audit configuration parameters are defined in the adtcfg file. The adtcfg file resides in aaodir of Informix distribution. When you change an audit configuration parameter, shared memory has to be re-initialized for the parameter to take effect. An alternative way of changing the parameter is through the **onaudit** command, where the parameter change has an immediate impact.

Example 3-1 shows audit parameters that you can configure. It shows that auditing is turned on and the audit path is set to /work/aaodir. The audit log file size is set to 5 M, and auditing continue whether there is an error or not.

Example 3-1 Audit configuration parameters

```
ADTMODE         1
ADTPATH         /work/aaodir
ADTSIZE         5000000
ADTERR          0
```

The audit parameters that you can configure are:

- ADTMODE

  This parameter can have values 0, 1, 3, 5, and 7, and their functions are:

0	Indicates that auditing is off
1	Indicates that auditing is turned on
3	Audits database system security officer (dbsso) operations
5	Audits database system administrator (dbsa) operations
7	Audits normal user operations as well as dbsso and dbsa operations

- ADTPATH

  This parameter indicates the path where audit logs are stored.

- ADTSIZE

  This parameter indicates the maximum size of the audit file. Once the file limit is exceeded, then a new file is created. The minimum size of the file is 10 K bytes.

- ADTERR

  This parameter can have the following values:

0	Indicates that auditing is to be continued in the face of errors
1	Indicates that the thread should be suspended as the audit record is written
2	Indicates that when the database server encounters an error, the thread is suspended and the system is shut down

3.4 Audit masks and events

This section describes the audit events and how to configure events in audit masks.

3.4.1 Audit events

Each database operation that needs to be audited has an event associated with it. An event has a corresponding mnemonic, and it is the mnemonic that gets logged in the audit file. For example, the *create table* database operation is auditable, and its audit mnemonic is CRTB. The audit events supported by the server and their mnemonics are listed in Appendix A, “Audit event mnemonics” on page 269.

Success and failure mnemonics

By prepending the audit mnemonic with S or F, you can log only success or failure events, respectively. For example, CRTB is the audit mnemonic to log a CREATE TABLE event whether the statement runs successfully or not. But the audit mnemonic SCRTB logs only successful CREATE TABLE events. If the audit mnemonic is FCRTB, the server logs only failed CREATE TABLE statements.

3.4.2 Audit masks

An audit mask contains a list of audit mnemonics. The database server checks the audit masks for events that need to be audited or excluded from auditing. Audit masks provide a way for a dbsso to easily add or delete events to be audited without affecting the existing configuration. The server checks the masks in a particular order to determine whether an event needs to be audited.

The Informix Dynamic Server supports the following mask types:

- Template masks

  As the name implies, template masks are templates of auditable events. It is easier for an administrator to derive other masks such as user masks for all users using template masks. As a convention, template masks should start with an underscore (_) and should not use the keywords _default, _require, or _exclude, which are system masks. One key differentiator between template masks and other masks is that the database server does not process template masks during the audit process.

- User masks

  As the name implies, user masks are specific to a particular user. The name of the user mask is the same as the user ID of the user. When a user runs a query, the database server first checks whether a user mask exists with the same name as the user making the connection. If the user mask exists, events defined in the user mask get audited first. If the user mask does not exist, then the database server starts processing the system masks (_default, _require, _exclude). User masks can be derived from other masks. Typically, an administrator defines several template masks and derives user masks for all the users from these masks. User masks are the first masks processed by the database server during the audit process.

- _default mask

  The _default mask contains the default set of events to be audited by the server. There is just one _default mask on the server. Typically, this mask contains auditable events common for all users. This mask is processed by the database server after processing the user mask.

- _require mask

  This mask contains all the events that must be audited. This mask has higher precedence than any of the masks discussed except the _exclude mask. An administrator uses this mask as a catch-all mechanism to audit events of high importance.

- _exclude mask

  The _exclude mask contains events that must not be audited. This mask has the highest precedence. An administrator uses this mask to filter out certain audit events that do not provide high value under a given circumstance. For example, if there is just one user database in the database server and thousands of users doing millions of transactions, the administrator can choose to exclude the open database (OPDB) audit event to reduce the size of the audit log. Another event that is typically included in the _exclude mask is the read row (RDRW) event, which produces enormous audit output in a high throughput transactional system. The _exclude mask applies only to normal users.

The database server checks the masks in the following order: user mask → _default mask → _require mask → _exclude mask.

The precedence order is exactly the opposite. For example, the _exclude mask has the highest precedence followed by the _require mask, and so on. This means that if an event is defined in both _require and _exclude masks, then that event is not audited, as the _exclude mask has higher precedence.

All the masks are created using the **onaudit** command.

3.5 onaudit

This section describes the **onaudit** command in more detail. We discuss various options of the **onaudit** command through examples of setting up audit configuration and audit masks. The **onaudit** command is chiefly used by the audit analysis officer (aao) and the dbsso. The aao uses this command to set up audit configuration. The dbsso uses this command to set up audit masks and events. For a more detailed discussion of role separation, refer to Chapter 2, “Role separation” on page 17.

3.5.1 Setup and configuration

The configuration parameters described in 3.3, “Audit configuration” on page 37, can also be edited using the **onaudit** command. The **onaudit** command provides an alternative way of changing the parameters without restarting the server.

Next we discuss some of the options to display and modify the audit configuration.

Changing audit mode

The audit command to change the audit mode is of the form:

```
onaudit -l <adtmode>
```

Refer to 3.3, “Audit configuration” on page 37, for various available audit modes and for a description of the ADTMODE configuration parameter. For example, in order to turn on auditing in the server, run the following command:

```
onaudit -l 1
```

In order to turn off auditing, run the following command:

```
onaudit -l 0
```

Changing audit error mode

The audit error mode configuration parameter is ADTERR. You can change the audit error mode by using the -e option. For example, to change the error mode to 2, run the following command:

```
onaudit -e 2
```

Changing audit path

The audit log files are stored under the directory specified by the audit path. You can change the audit log path dynamically using the -p option or the ADTPATH configuration parameter. For example, to change the audit path to /work/aaodir run the following command:

```
onaudit -p /work/aaodir
```

The -p option works only if auditing is enabled in the system. The directory specified by the -p option should be accessible by the aao.

Changing audit file size

The audit file size configuration parameter is ADTSIZE. You can change the audit file size using the -s option. For example, the following command changes the audit file size to 5 M bytes:

```
onaudit -s 5000000
```

The minimum size of audit file is 10 K bytes and the maximum size is 2 G. As the audit log files tend to fill up fast, use 5 M as the typical size of the file.

Starting a new audit log file

In order to start a new log file, run the following command

```
onaudit -n
```

This is useful if the administrator wants to start collecting audit logs fresh after making changes to audit configuration or audit masks. When this command is run, the server increments the sequence number used to identify the audit log file. For example, if the current audit log file that the server is writing to is <servername>.6, then executing the **onaudit -n** command enables the server to write to the new log file <servername>.7

Displaying audit configuration

In order to display current audit configuration of the server, run the following command:

```
onaudit -c
```

Example 3-2 presents the panel output of the **onaudit -c** command. The audit mode ADTMODE is set to 1. The error mode is set to 0, which means that the server will continue execution when errors are encountered. The audit file size is set to 5 M. The audit file number is set to 23, which means that the server is logging records to file *<servername>*.23. The audit file is the sequence number being used in audit log file names.

Example 3-2 Display audit configuration

```
>onaudit -c
Onaudit -- Audit Subsystem Configuration Utility
 Copyright IBM Corporation 1996, 2004 All rights reserved.


Current audit system configuration:
    ADTMODE    = 1
    ADTERR     = 0
    ADTPATH    = /work/aaodir
    ADTSIZE    = 5000000
    Audit file = 23
>
```

3.5.2 Manipulating audit masks using onaudit

In addition to setting up audit configuration, the **onaudit** command is also used to set up audit masks. You can use the **onaudit** command to create, delete, change, and display mask information.

Creating a mask

A mask has to be created before it can be changed. To create a template mask, execute the following command:

```
onaudit -a -u _templ0 -e +CRTB,RVLB
```

This command creates a template mask _templ0 with events CRTB and RVLB defined. The -u option is used to identify the mask name. The -e option is used to list the events defined in the mask. The -a option is used to create a mask.

Note the usage of a plus sign (+) before the event mnemonics. The plus sign indicates that the events are being added to the mask.

Note: A comma (,) is used as a separator if multiple events are added to the mask. There should be no space between a comma and an event mnemonic for all events to be added to the mask.

You can also create _default, _require, and _exclude masks in a similar fashion. Note that you cannot use _default, _require, and _exclude as template or user mask names, as these are keywords that the server understands and processes them in a particular order. You can create the _default mask as follows:

```
onaudit -a -u _default -e +DRTB,GRDB
```

Note: Although _default, _require, and _exclude are keywords in the system, they are not automatically defined. You need to explicitly create and add events to them before trying to change them.

Here after, we refer to _default, _require, or _exclude masks as system masks.

Modifying a mask

You can modify a mask by adding or deleting events from the mask. The -m option of onaudit is used to change a mask. For example, to modify the audit mask _templ0, run the following command:

```
onaudit -m -u _templ0 -e +GRSA,GRXM
```

To add an event to the _default mask, do the following:

```
onaudit -m -u _default -e +GRXM
```

As noted, the plus sign is used to add an event. Similarly, you can delete an event from a mask by using the a hyphen (-). For example, you can delete the event just added by doing the following:

```
onaudit -m -u _default -e -GRXM
```

Note that if no sign is specified before event mnemonics, it is implied that events are added to the mask.

Displaying masks

Now display the masks configured in the server by executing the **onaudit -o** command.

Example 3-3 displays the mask information. The -o option is used to output the masks and the -y option is used to respond *yes* to all prompts. In this example, the template mask _templ0 and the default mask _default contain the events defined.

Example 3-3 Display mask information

```
>onaudit -o -y
Onaudit -- Audit Subsystem Configuration Utility
 Copyright IBM Corporation 1996, 2004 All rights reserved.


_default                                       -       DRTB,GRDB
_templ0                                        -       CRTB,RVLB,GRSA,GRXM
```

Deriving from a base mask

You can derive a user mask or any of the system masks from a base mask. Typically, the administrator creates template masks and derives user masks from them. In such a case, the template mask acts as the base mask, However, you can also derive from any of the system masks (_default, _require, or _exclude) or a user mask.

The -r option of onaudit is used to derive from a base mask. For example, you can create an user mask usr1 from a template mask as follows:

```
onaudit -a -u usr1 -r _templ0
```

The mask usr1 now contains all events defined in template mask _templ0. Also, no trace is left if the events in the user mask are derived from some other mask.

Example 3-4 show the onaudit -o output.

Example 3-4 Output after deriving from a base mask

```
>onaudit -o -y
Onaudit -- Audit Subsystem Configuration Utility
 Copyright IBM Corporation 1996, 2004 All rights reserved.


_default                                  -       DRTB,GRDB,GRXM
_templ0                                   -       CRTB,RVLB,GRSA,GRXM
usr1                                      -       CRTB,RVLB,GRSA,GRXM
```

If you want to derive from a base mask but at the same time filter out some events from the base mask, you can specify options as follows:

```
onaudit -a -u usr2 -r _templ0 -e -CRTB
```

This command creates user mask usr2 and derives all events from template mask _templ0 except CRTB.

Example 3-5 shows the updated configuration. The **onaudit -o** command output does not list the base mask. In its place, a hyphen is output.

Example 3-5 After deriving from a base mask

```
>onaudit -o -y
Onaudit -- Audit Subsystem Configuration Utility
 Copyright IBM Corporation 1996, 2004 All rights reserved.


_default                                  -       DRTB,GRDB,GRXM
_templ0                                   -       CRTB,RVLB,GRSA,GRXM
usr1                                      -       CRTB,RVLB,GRSA,GRXM
usr2                                      -       RVLB,GRSA,GRXM
```

Be wary that if you modify a mask by deriving from a base mask. The pre-existing event mnemonics in the mask are overwritten.

Deleting a mask

You can use the **onaudit -d** command to delete a mask. For example, in order to delete the _default mask, do the following:

```
onaudit -d -u _default
```

Example 3-6 shows mask settings after dropping the _default mask.

Example 3-6 Audit mask display after deleting a mask

```
>onaudit -o -y
Onaudit -- Audit Subsystem Configuration Utility
 Copyright IBM Corporation 1996, 2004 All rights reserved.


_templ0                                     -       CRTB,RVLB,GRSA,GRXM
_templ1                                     -       CRDB
usr1                                        -       CRTB,RVLB,GRSA,GRXM
usr2                                        -       RVLB,GRSA,GRXM
```

Loading from an input file

The **onaudit** command can also be used to load an input file of masks. The audit masks can be predefined in an input file and loaded using the **onaudit -f** command. One scenario where this mechanism can be used is when an administrator wants to configure audit masks on multiple servers with similar configurations. The administrator creates the mask definitions in a file and loads the file on multiple servers. This reduces the amount of manual work that needs to be done, and also provides a copy of mask definitions for future use. The format of the input file is:

```
<mask name> <base mask > <event list>
```

The format of the file is the same as the output of the **onaudit -o** command except that for the output of the **onaudit -o** command, the base mask will not be listed. A hyphen is used in places where the base mask is not available.

Example 3-7 provides the contents of the sample input file mask_test.

Example 3-7 Input file for onaudit

```
mask001         _templ0                 DRLB,RNLC
mask002         -                       RVLB, CRTB
```

Now run the following command to load the masks defined in the input file:

```
onaudit -f mask_test
```

Example 3-8 shows that the audit masks were added. mask001 contains all events derived from _templ0 in addition to DRLB and RNLC.

Example 3-8 Mask definitions loaded

```
>onaudit -o -y
Onaudit -- Audit Subsystem Configuration Utility
 Copyright IBM Corporation 1996, 2004 All rights reserved.


_templ0                                 -       CRTB,RVLB,GRSA,GRXM
_templ1                                 -       CRDB
mask001                                 -
CRTB,RVLB,GRSA,DRLB,RNLC,GRXM
mask002                                 -       CRTB,RVLB
usr1                                    -       CRTB,RVLB,GRSA,GRXM
usr2                                    -       RVLB,GRSA,GRXM
```

Another scenario in which this mechanism is used is when the administrator wants to re-initialize the server. The administrator exports the current audit mask configuration using **onaudit -o -y** and reload it after server initialization using the **onaudit -f** command.

To save and reload to current audit mask information:

1. Save the current mask configuration in the mask_file file:

   ```
   onaudit -o -y > mask_file
   ```

2. Recycle the server:

   a. Bring the server down using the following command:

   ```
   onmode -ky
   ```

   b. Re-initialize the server:

   ```
   oninit -iy
   ```

3. Load the saved audit mask configuration from mask_file:

   ```
   onaudit -f mask_file
   ```

4. Verify that the masks are loaded correctly:

   ```
   onaudit -o -y
   ```

3.6 Analyzing the audit trail

Generating the audit trail is an important first step in the audit process. In most cases, generating and archiving the trail is the main security requirement. The trail is rarely analyzed except in cases where pilfering of information is suspected. In some cases, customers may choose to do regular analysis of the audit trail as part of the audit process. The main purpose is to actively monitor the audit trail for patterns that could reveal security violations. This also serves the secondary purpose of verifying that the audit logs are correctly generated and that they are not corrupted and can be accessed.

The information in the audit trail can be filtered by user, by server, or by time stamp. This is useful if a violation is reported at a particular time and the suspicion is on a particular user or group of users. Audit records can be made amenable to analysis by loading them into a database table and executing SQL queries on them to discover patterns of misuse. This is possible only if the audit trail is regularly monitored and normal patterns are known.

3.6.1 Audit file generation

Audit logs are generated in the directory specified by the ADTPATH configuration parameter. The name of an audit log file has the format *<servername>.<counter>,* where *<servername>* indicates the name of the server and *<counter>* is an integer that gets incremented each time a new audit log file is created.

For example, for the server ol_lx_rama, the audit log file names are of the form ol_lx_rama.1. When the current audit log file is full, a new file is created after the counter is incremented. Thus, the audit log files run in the following sequence:

```
ol_lx_rama.1 → ol_lx_rama.2 → ol_lx_rama.3 ...
```

The naming scheme ensures that different server instances can share the same ADTPATH directory without causing name conflicts. A server with multiple aliases generates the audit trail in file names based on the DBSERVERNAME parameter and not on any of the aliases.

The size of an audit log file is determined by the ADTSIZE configuration parameter or the **`onaudit -s`** command. A new audit log file is created when the current audit log file exceeds the size configured. A new audit log file is also created by using the -n option.

Collectively, the set of audit log files created under the ADTPATH directory constitutes the audit trail. The size of the audit trail can grow very quickly depending on the events audited. Certain events such as read row generate massive audit output that fills up the disk space quickly. The administrator can devise policies to archive the audit trail in a regular manner. Also, compression techniques can be used to reduce the size of the audit trail. There are utilities available that reduce the size of the audit log to about 5 –10% of the original size. Compression also potentially resets the numbering of audit trail files. Care has to be taken to set the correct number after the compression is completed.

3.6.2 Audit record format

When the database server encounters an auditable event, it generates an audit record. The audit record gets immediately logged to audit log file. The audit record contains information pertaining to the event that can later be used during the analysis phase.

Example 3-9 presents a sample audit log content. To get fine granular time, set USEOSTIME to 1.

Example 3-9 Sample audit trail

```
ONLN|2007-10-02 13:36:03.310|lx-rama|24173|ol_lx_rama|usr1|0:CRLC:db:mycompa
ONLN|2007-10-02 13:36:03.324|lx-rama|24173|ol_lx_rama|usr1|0:CRLC:db:mycomps
ONLN|2007-10-02 13:36:03.324|lx-rama|24173|ol_lx_rama|usr1|0:CRLC:db:mycompt
ONLN|2007-10-02
13:36:03.324|lx-rama|24173|ol_lx_rama|usr1|0:RNLC:db:mycompa:newlevel
ONLN|2007-10-02 13:36:03.324|lx-rama|24173|ol_lx_rama|usr1|0:DRLC:db:newlevel
ONLN|2007-10-02 13:36:03.325|lx-rama|24173|ol_lx_rama|usr1|0:CRLC:db:mycompa
ONLN|2007-10-02 13:36:03.325|lx-rama|24173|ol_lx_rama|usr1|0:CRPL:db:myplcy
ONLN|2007-10-02
13:36:03.325|lx-rama|24173|ol_lx_rama|usr1|0:CRLB:db:myplcy.mylbl
ONLN|2007-10-02
13:36:49.112|lx-rama|24173|ol_lx_rama|usr1|0:GRLB:db:myplcy.mylbl:USR2:R
ONLN|2007-10-02
13:36:49.123|lx-rama|24173|ol_lx_rama|usr1|0:RVLB:db:myplcy.mylbl:USR2:A
ONLN|2007-10-02
13:36:49.234|lx-rama|24173|ol_lx_rama|usr1|0:GRXM:db:myplcy:USR2
:IDSLBACREADARRAY
ONLN|2007-10-02
13:36:49.234|lx-rama|24173|ol_lx_rama|usr1|0:RVXM:db:myplcy:USR2
:IDSLBACREADARRAY
ONLN|2007-10-02 13:36:49.234|lx-rama|24173|ol_lx_rama|usr1|0:GRSS:db:USR3:USR2
ONLN|2007-10-02 13:36:49.234|lx-rama|24173|ol_lx_rama|usr1|0:GRSS:db:USR4:USR2
ONLN|2007-10-02
13:37:23.345|lx-rama|24173|ol_lx_rama|usr1|-8208:ALLC:db:level:A
ONLN|2007-10-02
13:37:23.345|lx-rama|24173|ol_lx_rama|usr1|-8208:ALLC:db:compart
:S
```

```
ONLN|2007-10-02
13:37:23.345|lx-rama|24173|ol_lx_rama|usr1|-8208:ALLC:db:group:T
```

Now, let us take a typical audit record and study its format:

```
ONLN|2007-10-02 13:36:49.234|lx-rama|24173|ol_lx_rama|usr1|0:GRSS:db:USR3:USR2
```

As shown, an audit record constitutes multiple fields separated by the | character. The following gives a description of each of the fields:

- ONLN - instance tag

  This signifies that this record belongs to an Informix server instance.
- 2007-10-02 13:36:03.000 - time stamp

  This is the time at which the event occurred in the server.
- lx-rama - host name

  This gives the host name on which the Informix instance is running.
- 24173 - process ID

  This gives the PID of the Informix process that encountered the audit event.
- ol_lx_rama - server name

  This is the name of the database server that encountered the audit event. The DBSERVERNAME parameter in the onconfig file defines the server name.
- usr1 - user name

  This gives the name of the user who runs the SQL statement that resulted in this audit event.
- 0:GRSS:db:USR3:USR2 - event-specific information

  This gives the event-specific information. The sub-fields pertaining to the event are separated by a colon (:).

  The first sub-field is an error code. The value 0 indicates that it is a SUCCESS event.

  The second sub-field gives the event mnemonic. The event mnemonic GRSS indicates that it is a GRANT SETSESSIONAUTH statement.

  The third sub-field gives the database name. The statement has been run on database db.

  The rest of the fields give information pertaining to the event. In this case the GRANT SETSESSIONAUTH statement is run on USR2 to USR3. These sub-fields vary for each event, and so they are different for each audit record.

3.6.3 onshowaudit

An audit log file can be opened by the aao. However, it is not advisable to open the audit log files manually, as it is possible to accidentally tamper with or corrupt the file. Once the audit log file is corrupted, it becomes unusable for further analysis.

The most secure way to read the audit records is by using the onshowaudit utility. The **`onshowaudit`** command reads the audit log files and the resultant output can be redirected to another file for further analysis. **`onshowaudit`** can only be run by an aao or user informix. The following sections give several options of **`onshowaudit`**.

Reading the audit trail

The audit trail is read from the path specified by the ADTPATH configuration parameter. The command that is run to read the complete audit trail is the following:

```
onshowaudit -I
```

This command reads all the audit log files that exist at the path specified by the ADTPATH configuration parameter.

Note: In IDS 11.10, **`onshowaudit`** reads the audit trail from the location specified by ADTPATH configuration parameter. Even when this value is changed to a different value using **`onaudit -p`**, **`onshowaudit`** still reads audit trail from the old path.

Reading a specific audit log file

You can read a specific audit log file as shown in the following:

```
onshowaudit -I -f /work/aaodir/ol_lx_rama.7
```

Filtering audit records by user

The audit administrator may not want to retrieve all the audit records. At a certain time, records pertaining to a particular user may only be relevant. This can be achieved by using the following command:

```
onshowaudit -I -f /work/aaodir/ol_lx_rama.7 -u usr1
```

Filtering audit records by server name

You can also filter records by server name using the -s option. The following command retrieves the records pertaining to a particular user and server in a specific audit log file:

```
onshowaudit -I -f /work/aaodir/ol_lx_rama.7 -u usr1 -s ol_lx_rama
```

3.6.4 Analyzing with SQL

Audit records can be loaded into a database in order to harness the expressive power of SQL and the processing power of a database server. As such, the audit record format is amenable to loading by utilities such as dbload. There are seven fields in an audit record, with the last field being event information. The event information field can be further divided into sub-fields so that queries can use these sub-fields as predicates.

Audit table schema

Readers can create their own schema based on audit record format and their own needs.

Example 3-10 gives a SQL script to create audit table schema to load audit records.

Example 3-10 Creating audit record table

```
DROP DATABASE aud_db;
CREATE DATABASE aud_db;

CREATE TABLE aud_tab (
        instance                        CHAR(5),
        timestamp                       DATETIME YEAR TO FRACTION(3),
        host                            CHAR(32),
        pid                             INT,
        server                          CHAR(32),
        username                        CHAR(32),
        errno                           INT,
        event_code                      CHAR(5),
        database                        CHAR(128),
        eventinfo                       CHAR(2048)
        );
```

There are ten columns in the event table. The first six columns map directly into the first six fields of the audit record. The final field of the audit record is divided into four columns with the first three sub-fields (errno, event mnemonic, and database) mapping into columns seven, eight, and nine, respectively, of the audit table. The remainder of the last field maps into the last column eventinfo. We propose this schema because the first three sub-fields of the final audit record field are common for all audit records. The other sub-fields are variable and are captured in the final column as eventinfo.

Readers may choose to implement their own schema where even the variable part of event information is fanned to separate columns. The audit table schema can be tailored to meet the needs of the analysis officer and the person writing analysis queries.

Audit record conversion

The audit records generated by the database server do not exactly map into the audit table schema. Conversion has to be done to split the fixed part of event information into separate fields so that utilities like dbload can directly load the records into the database.

Example 3-11 shows the format that a sample audit record should be converted to.

Example 3-11 Audit record conversion

```
Original record:

ONLN|2007-10-02 13:36:03.000|lx-rama|24173|ol_lx_rama|usr1|O:GRSS:db:USR3:USR2

Converted as the following:

ONLN|2007-10-02 13:36:03.000|lx-rama|24173|ol_lx_rama|usr1|O|GRSS|db|USR3:USR2
```

Example 3-12 gives the Perl script to convert audit records from **onshowaudit** output so that they can be loaded into the audit table.

Example 3-12 Audit record converting script

```
#!/usr/local/bin/perl
# Converts audit log records to database load format. The format matches
# the load table schema. The input to this script is the output from
# "onshowaudit" program.
#
$file="audit_input.txt";
open (INFO, $file);
@lines = <INFO>;
foreach $line (@lines)
{
    chomp $line;
    if ($line)
        {
#       print $line."\n";
        @fields = split(/\|/, $line);

        @eventinfo = split(/:/, $fields[6]);
        $eventfield1 = shift @eventinfo;
        $eventfield2 = shift @eventinfo;
        $eventfield3 = shift @eventinfo;

        $eventfield4 = join (':', @eventinfo);
        $newline =
$fields[0]."|".$fields[1]."|".$fields[2]."|".$fields[3]."|".$fields[4]."|".$fie
lds[5]."|".$eventfield1."|".$eventfield2."|".$eventfield3."|".$eventfield4;
|".$eventfield;
        print $newline."\n";
        }
}
~
```

Loading audit records into database table

The converted records are in the correct format to be loaded into the audit table. Use the **dbload** command file to load the records, as shown in Example 3-13.

Example 3-13 Load audit records

```
FILE loadinput.dbl  DELIMITER '|' 10 ;
INSERT INTO aud_tab;
```

As shown in the command file, **dbload** loads records from the input file loadinput.dbl. Each record has ten fields, and the fields are delimited by the | character. The records are loaded into the aud_tab table.

In most cases, loading serves the purpose of identifying issues with audit record generation. By routinely loading the audit trail, the administrator ensures that the audit trail is generated and is usable.

Analyzing audit data

Once the audit table is loaded, the aao starts analyzing the data. The analysis queries and methodologies are specific to company security requirements. The aao performs routine analysis to find hidden anomalies in access patterns. Queries are run to identify non-compliance with security policies. In must be stated here that analysis is an open-ended activity and any amount of time can be spent looking for security breaches and vulnerabilities. Hence, we make a distinction between open-ended analysis and analysis that ensures compliance with defined security policies by both privileged and normal users.

It is also important to realize that a discovered security breach during analysis does not immediately make the user identified in the audit record the prime suspect. Very often, discretionary access control (DAC) and label-based access control (LBAC) restrict users to the tasks that they are assigned to do. If users change their identities to a dbsa or dbsso, they are now free to execute tasks that only a privileged user can perform. This type of breach is recorded as normal activity in the audit trail, if the user who changed identity does not commit serious errors that could be caught by aao-defined queries. However, if OS auditing is turned on for operations such as su or login on UNIX systems, it is possible to identify unauthorized transgressions. DBMS auditing alone does not identify all the breaches and violations. Database auditing done in conjunction with auditing at OS and the network level helps expose some of the sophisticated breaches and violations.

We present several scenarios where each scenario highlights the need for analyzing audit information in a certain manner.

Scenario 1 - complying with defined security policies

Is the DBSECADM complying with defined security policies? The following presents a scenario:

> Company XYZ has implemented Label Based Access Control. The DBSECADM assigned labels to different users who access data. There is a table secure_tab which is protected by the security policy secret_policy and the data in this table is deemed highly confidential and only certain privileged users are granted access to this table. However, the company workflow requires that certain users at lower security levels need to be granted access to secure_tab routinely to access certain information. The company security policy requires that the DBSECADM grant exemptions to these users on security policy secret_policy for a period of no more than two hours at a time. TO comply with the security policy any exemption granted has to be revoked with in two hours. Also, such exemptions should not be granted more than once to a user in a 30-day period.

Based on the scenario, the aao checks for cases where exemptions are granted to users (GRXM) but not yet revoked (RVXM). The aao also checks for cases where exemptions are not revoked within two hours of granting. Also, the aao checks that such exemptions are not granted more than once to a particular user in a 30-day period. This is one example of active monitoring of an audit trail for non-compliance with one of the company security policies.

Scenario 2 - tracking access

Is the company payroll information secure? Here we present a scenario in which the aao tracks access to payroll information:

> Company Atin Corp wants to restrict updates to its payroll information to only 2 users, usr1 and usr2. The DBA creates a role, payroll, and grants update privilege on Salary table to this role. This role is then granted to users usr1 and usr2. The company security policy stipulates that only one user should access salary table on any given day.

Based on the scenario, the aao looks for the set role audit mnemonic (STRL) by users usr1 and usr2. Also, the aao ensures that the DBA has not granted the payroll role to any other user. Additionally, the aao verifies that DML privileges on the salary table are not granted to any other role or user.

Scenario 3 - detecting access violation

Is your informix super-user accessing sensitive information? This scenario describes where the aao detects that the informix super-user is accessing the table containing sensitive information:

> The user `informix` is all powerful in IDS. The user `informix` can access any table and assume any privilege. The company security policies stipulate that user `informix` should perform administrative responsibilities and not access any user data.

The aao checks the audit records to see whether user informix is accessing user tables.

There are other scenarios in which an aao looks for anomalous access patterns such as unusual activity on a particular table, too many errors generated, and so on.

3.7 Sysmaster tables

The tables sysaudit and sysadtinfo in the sysmaster database contain audit-related information. The sysaudit table contains audit masks created using the **`onaudit`** command and the sysadtinfo table contains the audit configuration. These tables are shown here only for administrator reference. Knowledge of these tables is not needed for setting up and maintaining the audit process.

Table 3-1 shows the schema for the sysadtinfo table. The table captures the audit configuration parameters.

Table 3-1 SYSADTINFO

Column	Type	Description
adtmode	INTEGER	Audit mode.
adterr	INTEGER	Error mode.
adtsize	INTEGER	Audit log size.
adtpath	CHAR(256)	Audit path.

Column	Type	Description
adtfile	INTEGER	Audit file number. Current sequence number used for audit log files. This number can be changed using the **`onaudit -n`** command.

Table 3-2 shows the schema of the sysaudit table. The column username represents the mask name, the columns succ1 to succ5 represent the bit mask for success events, and the columns fail1 to fail5 represent the bit mask for failure events. If an event bit is set in both these masks, that event generates an audit record whether the operation is a success or a failure.

Table 3-2 SYSAUDIT

Column	Type	Description
username	*CHAR(32)*	Name of the mask
succ1	INTEGER	Bitmask of audit mask for success
succ2	INTEGER	Bitmask of audit mask for success
succ3	INTEGER	Bitmask of audit mask for success
succ4	INTEGER	Bitmask of audit mask for success
succ5	INTEGER	Bitmask of audit mask for success
fail1	INTEGER	Bitmask of audit mask for failure
fail2	INTEGER	Bitmask of audit mask for failure
fail3	integer	Bitmask of audit mask for failure
fail4	integer	Bitmask of audit mask for failure
fail5	integer	Bitmask of audit mask for failure

3.8 Role separation in auditing

Auditing presents a case for classic role separation. Auditing tasks are divided into different categories and each category is assigned a role, as shown in Table 3-3 on page 58.

Table 3-3 Auditing role separation

Auditing tasks	Responsible role
► Enabling/disabling auditing ► Setting audit modes, paths, and size of audit files ► Managing the disk space where audit trail is getting stored ► Analyzing the audit trail	aao
► Determining what is audited ► Setting up user masks and adding events to various masks	dbsso
► Enabling auditing at OS and network level	OS administrator

A close interaction between aao and the OS administrator is useful in identifying and preventing security breaches.

Although separate roles exist, a company may have a single person play all the roles for cost or convenience reasons.

Role separation is described in more detail in Chapter 2, “Role separation” on page 17.

3.9 Performance considerations in auditing

Auditing provides an enormous security benefit. The mere knowledge that the database system is audited deters potential intruders. However, auditing comes with a cost in terms of system performance, disk space, and administrative overhead spent in managing and analyzing the audit trail.

Once the company decides to audit its database system, the database system administrator (dbsa) needs to determine the impact of auditing on a smoothly running DBMS server. The database system security officer (dbsso) determines what events are audited and sets up masks to achieve the same. The impact of auditing on system performance depends on the following:

- What events are audited?
- How many events are audited?
- The current throughput and utilization of the system.
- How many user transactions encounter auditable events?
- Is the audit trail getting periodically archived?

Certain events such as read row (RDRW) generate enormous audit output especially in a high-throughput transactional system. Having high-frequency audit events impacts the response time of queries that run on the system. Similarly, if the number of events being audited is quite high, system performance is negatively impacted. In most cases this is expected because logging an audit record is a step that happens before the query results are sent back to the user, and the response time will be slower even if it is the only query running on the system. The administrator role ensures that writing the audit trail to the disk is not adversely impacting the system throughput by using high-speed disks for storing the audit trail, by adding more capacity to the system and tuning the system for optimal performance.

Disk space in the file system for storing the audit trail and in DBMS for loading audit logs is another cost that needs to be carefully planned. The audit trail may need to be archived to a backup store for compliance reasons. The amount of space allocated for analysis data and whether a separate DBMS is needed to run analysis queries needs to be carefully addressed. In most cases, the entire audit trail need not be loaded into the database. Scripts can filter most of the data and load only relevant data that the aao is interested in to run queries.

If the system performance degrades to an unacceptable level, the dbsso revisits the audit event setup and disables certain high-frequency events to improve system performance.

4

Securing data with SQL

This chapter describes various ways of securing access to data using SQL statements. Initially, we describe various access control mechanisms available in the database server to control access to the SQL objects. This is followed by a description of the column-level encryption feature, used to encrypt data in specified columns of a table.

The access control mechanisms available in the Informix Dynamic Server (IDS) are broadly classified into discretionary access control (DAC) and label-based access control (LBAC). DAC is enforced in the database server using privileges and roles. LBAC enforcement is done by using security labels.

The database administrator (DBA) plays a critical role in ensuring that sensitive data is protected from unauthorized users. SQL provides the means in achieving data protection at different levels of granularity. The granularity varies from protecting objects at the database level to protecting individual rows and columns using security labels. Note that a DBA is different from a database system administrator (dbsa). The role of a dbsa is to perform server maintenance, whereas the role of a DBA is to create and maintain databases.

4.1 Discretionary access control

Discretionary access control is the primary access control mechanism that enables access to SQL objects using privileges and roles. The objects in the database server that are protected using DAC are databases, tables, columns, views, table types, routines, and languages. By protecting these objects and by granting privileges to access these objects to authorized users, effective access control is achieved. Users who do not have the necessary privileges will not be able to execute the queries that access these objects.

In a typical IDS installation, there are a large number of users accessing the system, executing queries, and modifying data. Granting necessary privileges to all these users can make the administrator role quite tedious. Furthermore, company policies may stipulate that users assume a different set of privileges depending on the work that they perform at any moment. This makes the administrator responsibility more onerous, as privileges have to be constantly granted and revoked from users based on the least privilege principle. The roles capability in the server alleviates the extra effort of the administrator.

Roles are created for specific responsibilities, and privileges are assigned to roles. Then the administrator grants these roles to specific users. A user can assume a role depending on the task being performed at any given time. In addition to the privileges granted to them explicitly, users derive privileges from their current role settings. These privileges dictate what operations this user can perform on SQL objects. Thus, setting a role is an application responsibility. The privileges of a application depend on what roles are set when performing an operation.

Thus, role-based access control extends privilege-based access control by grouping a set of privileges into roles and assigning these roles to users. At any point, the roles defined in the database may mimic the organizational structure in the company. Thus, while users come and go, the initial set of roles defined by the administrator changes less often and is easily maintainable.

4.1.1 Protected SQL objects

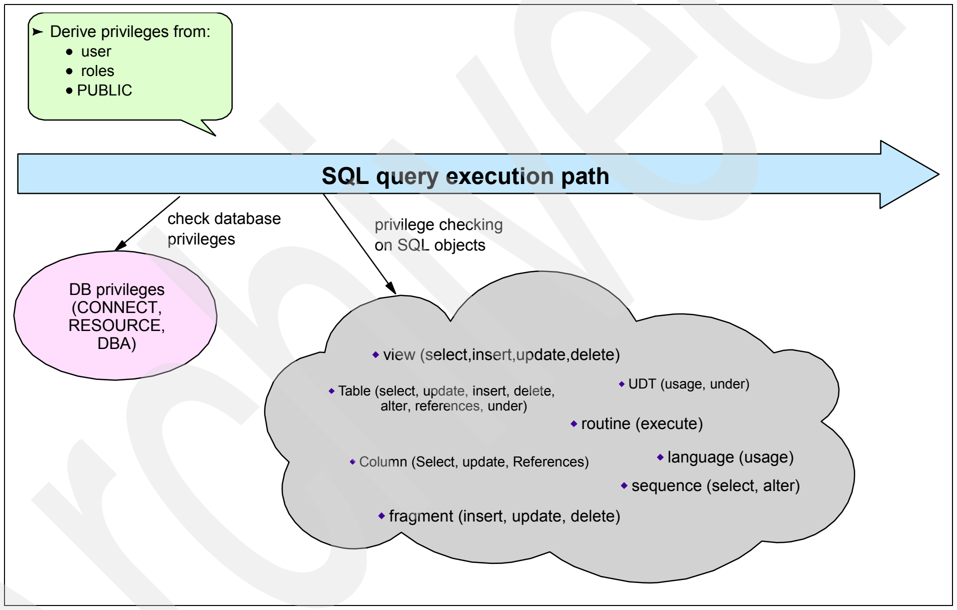
The following objects in the DBMS require users to have specific authorization or privileges in order to access them:

- Databases
- Tables
- Columns in tables
- Fragments
- Views

- User-defined types (UDT)
- Routines
- Languages

Most of the SQL queries submitted by the user interact with one or more of these objects. Privileges granted to the user on these objects dictate whether the user can perform the desired operation.

Figure 4-1 shows the privilege checking done in the course of an SQL query execution path. First, database privileges are checked. This is followed by checking privileges on SQL objects associated with the user query. User privileges are based on privileges granted to the user explicitly, privileges derived from user's current role, and privileges that belong to PUBLIC.



Derive privileges from:
• user
• roles
• PUBLIC
SQL query execution path
check database privileges
privilege checking on SQL objects
DB privileges (CONNECT, RESOURCE, DBA)
view (select,insert,update,delete)
Table (select, update, insert, delete, alter, references, under)
UDT (usage, under)
routine (execute)
Column (Select, update, References)
language (usage)
sequence (select, alter)
fragment (insert, update, delete)

Figure 4-1 Privilege checking in SQL query

Table 4-1 provides a summary of privileges available for a particular SQL object.

Table 4-1 Privileges associated with each SQL object type

SQL object	Privileges
Database	CONNECT, RESOURCE, DBA
Table	SELECT, UPDATE, INSERT, DELETE, INDEX, ALTER, REFERENCES
Column	SELECT, UPDATE, REFERENCES

SQL object	Privileges
Fragment	INSERT, UPDATE, DELETE
View	SELECT, INSERT, UPDATE, DELETE
Sequence	SELECT, ALTER
UDT	USAGE, UNDER
Routine	EXECUTE
Language	USAGE

As shown in Table 4-1 on page 63, each SQL object is associated with certain privileges and each privilege has an associated set of capabilities. When a user is granted a privilege, the user has permissions to exercise that capability on the SQL object.

For example, if a user is granted the RESOURCE privilege on a database, it means that the user has the ability to connect to the database and create objects within that database. A database object can be either a table, a view, a UDT, a routine, or some other SQL object contained within the database. If another user wants to create a view on the created table, then this user at least needs the CONNECT privilege on the database and the SELECT privilege on the table. This example provides a perspective on the privilege-based access control mechanism in IDS.

The SQL statements that are used to grant or revoke privileges to users are GRANT and REVOKE, and they are described in detail later.

4.1.2 Database privileges

The privileges associated with a database are CONNECT, RESOURCE, and DBA. Table 4-2 gives a brief description of each.

Table 4-2 Database privileges

Database privilege	Description
DBA	Creator/owner of the database. Ability to grant privileges on other objects in the database.
RESOURCE	Can connect to the database and create other objects.
CONNECT	Can connect to the database and execute queries.

The DBA privilege is granted to the user who creates the database, but can also be granted to other users later. The DBA privilege is the highest of the three database-level privileges. Hereafter, we refer to a DBA privileged user as a DBA. A DBA can perform all operations in the database. The DBA can also grant DBA, RESOURCE, and CONNECT privileges to other users of the database. Thus, more than one DBA can exist for a database. Typically, the creator of an object in a database is the owner of an object. A DBA can also create objects in the database to be owned by others. The tasks a DBA can perform are:

- Grants and revokes database-level privileges to other users
- Grants and revokes privileges on other objects in the database such as tables, views, sequences, UDT, routines, languages, and so on
- Creates tables, views, and indexes in the database to be owned by other users
- Drops or alters any object in the database regardless of who owns it
- Can perform the DROP DATABASE statement
- All operations that a RESOURCE privileged user can perform

The RESOURCE privilege has the second highest ranking in a database next to DBA. The DBA grants RESOURCE privilege to users to create other SQL objects within the database. The tasks a RESOURCE privileged user can perform are:

- Create permanent tables, indexes, and SPL routines. Tasks that require allocation of disk space.
- All operations that a CONNECT privileged user can perform.

The DBA grants the CONNECT privilege to users to connect to the database and execute queries. The CONNECT privilege has the least ranking because users are not able to create other objects within the database with this privilege. The following are the tasks that a CONNECT privileged user can perform:

- Connect to the database.
- EXECUTE the queries and DML statements on tables of a database provided that the users have the necessary table-level privileges.
- Execute an SPL routine provided that they have the necessary table-level privileges.
- Create views provided the users are permitted to query on tables on which the view is based.
- Create temporary tables and create indexes on temporary tables.

Example 4-1 provides the syntax of the GRANT and REVOKE statements on database objects. In the example, *privilege* represents one of CONNECT, RESOURCE, or DBA privileges. *user-list* represents a list of users including PUBLIC.

Example 4-1 Grant and revoke syntax

```
GRANT privilege TO user-list
REVOKE privilege FROM user-list
```

Example 4-2 provides a script to create a database and grant the RESOURCE privilege to usr1. This user can now connect to the database db and create permanent tables in the database. The CONNECT privilege on the database has been granted to PUBLIC, which means that all users of the system can connect to the database and execute queries. The success or failure of these queries depends on the objects that these queries access and the privileges that the user possesses on these objects.

Example 4-2 Demonstrates GRANT

```
CREATE DATABASE db WITH LOG MODE ANSI;

GRANT RESOURCE TO "usr1";
GRANT CONNECT TO PUBLIC;
```

Displaying database privileges

The system catalog table SYSUSERS contains the information pertaining to database privileges. The username and usertype columns give the user name and privilege, respectively. In the usertype column, D represents the DBA privilege, R represents the RESOURCE privilege, and C represents the CONNECT privilege.

Example 4-3 displays database privileges for users from system catalog table SYSUSERS.

Example 4-3 Display of database privileges from system catalog table SYSUSERS

```
6246 >dbaccess db -

Database selected.

> SELECT username, usertype FROM SYSUSERS;


username                         usertype
```

```
dbauser                          D
usr1                             C
usr2                             R
user3                            D
4 row(s) retrieved.

>
```

4.1.3 Using SQL to control access to data

Before describing privileges associated with SQL objects such as tables, routines, languages, and so on, we describe in more detail the GRANT and REVOKE statements provided by IDS.

SQL provides GRANT and REVOKE statements to grant and revoke privileges to users, respectively. GRANT and REVOKE are also used to grant roles and other security administration options such as SETSESSIONAUTH to the user.

In this section we describe the syntax and semantics of the grant and revoke statements on each of the SQL objects.

GRANT statement

A privilege is granted to a user or a list of users. PUBLIC is a keyword used to represent all the users in the database system. When a privilege is granted to PUBLIC, it means that all the users in the system acquire that privilege. Similarly, when a privilege is revoked from PUBLIC, all users in the system are stripped of the privileges granted to PUBLIC. However, they still retain the privileges granted to them individually, or granted through a role.

Example 4-4 provides some of the GRANT statements on various SQL objects. As shown, database-level privilege CONNECT is granted to usr2, table-level privilege UPDATE is granted to usr3, type-level privilege UNDER is granted on the super table tab_super to usr4, language-level privilege USAGE is granted on SPL language to usr5, and finally routine-level privilege EXECUTE is granted on procedure myproc to usr6.

Example 4-4 Sample GRANT statements

```
GRANT CONNECT TO "usr2";
GRANT UPDATE ON tab1 TO "usr3";
GRANT UNDER ON tab_super TO "usr4";
GRANT USAGE ON LANGUAGE SPL TO "usr5";
GRANT EXECUTE ON myproc TO "usr6";
```

The privileges that you grant remain in effect until you revoke the privilege. Only the grantor of a privilege or DBA can revoke it.

Example 4-5 gives the corresponding REVOKE statements for grants in Example 4-4 on page 67.

Example 4-5 Sample REVOKE statements

```
REVOKE CONNECT FROM "usr2";
REVOKE UPDATE ON tab1 FROM "usr3";
REVOKE UNDER ON tab_super FROM "usr4";
REVOKE USAGE ON LANGUAGE SPL FROM "usr5";
REVOKE EXECUTE ON myproc FROM usr6;
```

The GRANT and REVOKE statements come with additional clauses such as WITH GRANT OPTION, AS *grantor*, AS *revoker*, and CASCADE/RESTRICT clauses. Some of these are only applicable to certain types of SQL objects. The WITH GRANT OPTION and AS grantor clauses are applicable to the GRANT statement, while the AS REVOKER and CASCADE/RESTRICT clauses are applicable to the REVOKE statement.

WITH GRANT OPTION

The WITH GRANT OPTION clause in a GRANT statement is used to indicate that the grantee of the privilege can in turn grant it to other users. The WITH GRANT OPTION clause is not applicable to database-level privileges, but is applicable to privileges on all other SQL objects.

Example 4-6 shows that usr3 has been granted the UPDATE privilege on table tab1 with WITH GRANT OPTION. This means that usr3 can grant the same privilege on the same table to other users.

Example 4-6 Usage of WITH GRANT OPTION

```
GRANT UPDATE ON tab1 TO "usr3" WITH GRANT OPTION;
GRANT INSERT ON tab1 TO "usr3";
```

Example 4-7 demonstrates the effect of the WITH GRANT OPTION clause used in Example 4-6. As shown, usr3 can now grant the UPDATE privilege on table tab1 to other users, but cannot grant the INSERT privilege on table tab1 to anyone.

Example 4-7 Demonstration of a user with WITH GRANT OPTION granting privileges

```
6138 >su usr3

1 >dbaccess db -
```

```
Database selected.

> GRANT UPDATE ON tab1 TO "usr7";

Permission granted.

> GRANT INSERT ON tab1 TO "usr7";

  302: No GRANT option or illegal option on multi-table view.
Error in line 1
Near character position 29
```

AS grantor clause

The AS grantor clause in the GRANT statement allows the user with DBA privilege to grant privileges to users as another user indicated by grantor. This means only the grantor can now revoke these privileges.

Example 4-8 shows the usage of the AS grantor clause. The first statement shows that the UPDATE privilege is granted on tab1 to usr3 WITH GRANT OPTION. In the next statement, by using the AS usr3 clause, usr3 is the grantor of the UPDATE privilege on tab1 to usr7.

Example 4-8 Usage of AS grantor 1

```
GRANT UPDATE ON tab1 TO "usr3" WITH GRANT OPTION;
GRANT UPDATE ON tab1 TO "usr7" AS "usr3" ;
```

Another scenario where the AS grantor clause is used is when the owner of an object is a user not recognized by the operating system. In this scenario, the user with the DBA privilege cannot grant privileges on this object to users.

Example 4-9 demonstrates the fact that only the owner of an object can grant privileges on the object. The DBA creates a tab3 table owned by user dummy in an ANSI database. The user dummy is not a known user to the operating system. The DBA then tries to grant the INSERT privilege on tab3 to usr8. This statement fails because the DBA is not the owner of tab3. As the user dummy is not known to the operating system, the only way to grant privileges on tab3 is by using the AS grantor clause, as shown in Example 4-9.

Example 4-9 Demonstration that only the owner can do GRANT

```
CREATE TABLE "dummy".tab3 (i int);

6156 >dbaccess db -
```

```
Database selected.

> GRANT INSERT on "dummy".tab3 to "usr8";

  302: No GRANT option or illegal option on multi-table view.
Error in line 1
Near character position 37
> GRANT INSERT on "dummy".tab3 to "usr8" AS "dummy";

Permission granted.
```

REVOKE statement

This section discuss the REVOKE statement.

Example 4-10 shows how to revoke the privilege that is granted in Example 4-7 on page 68. The privilege that is granted on an object with the AS *grantor* clause can only be revoked by the grantor or by the DBA by using the AS *revoker* clause.

Example 4-10 Revoking privileges

```
6148 >dbaccess db -

Database selected.

> REVOKE UPDATE ON tab1 from "usr7";

  580: Cannot revoke permission.

  111: ISAM error:  no record found.
Error in line 1
Near character position 32

6150 >su usr3
1 >dbaccess db -

Database selected.

> REVOKE UPDATE ON tab1 FROM "usr7";

Permission revoked.
```

AS revoker clause

You cannot revoke a privilege from yourself. You cannot revoke *grantor* status from another user. If you want to revoke a privilege that is granted with the AS grantor clause of the GRANT statement, you must have the DBA privilege and you must use the AS revoker clause, where revoker is the same as grantor in a GRANT statement.

Example 4-11 shows how to revoke the privilege granted to usr3 in Example 4-8 on page 69.

Example 4-11 Revoke privilege granted with AS grantor

```
REVOKE UPDATE ON tab1 FROM "usr7" AS "usr3";
```

Effect of CASCADE and RESTRICT clause on Revoke

A privilege can be revoked with a CASCADE or RESTRICT clause. This is applicable for objects such as tables and super-types that have other objects depending on them. For example, a view is created only if the user has the SELECT privilege on the base table. If the SELECT privilege on the table is revoked after the view is created, then the view is dropped. This is equivalent to specifying the CASCADE clause in the revoke statement. Instead, if the RESTRICT clause is specified while revoking the SELECT privilege on the table, then the revoke statement returns an error because the view is dependent on the SELECT privilege of the table. Examples are provided in 4.1.9, “VIEW privileges” on page 87.

A similar argument applies for the UNDER privilege on super-types. If the UNDER privilege on a super-type is revoked from a user, then all the sub-types created by that user are dropped. You get equivalent behavior with the CASCADE option. However, with the RESTRICT option, REVOKE statement returns an error. Examples are provided in 4.1.8, “TYPE privileges” on page 84.

Other dependencies that cause the REVOKE with RESTRICT clause to fail are:

- A foreign-key constraint depends on a REFERENCES privilege that you attempt to revoke.
- You attempt to revoke a privilege from a user who subsequently granted this privilege to another user or to a role.

REVOKE does not fail if the user specified has the right to grant the privilege to others, but has not exercised that right.

Object ownership

Every SQL object has an owner. The owner of an object is typically the user who created it. Only a DBA can create an object to be owned by others. The owner of an object has all rights to the object and can alter or drop it without additional privileges.

Example 4-9 on page 69 shows a way of creating a table to be owned by another user. A table owner cannot deny themselves the privileges that come with ownership, nor can anyone else deny them that right.

4.1.4 ANSI verses non-ANSI databases

There is a difference in the way privileges are granted upon object creation between ANSI and non-ANSI databases. In a non-ANSI database, when an object is created, privileges on the object are automatically granted to PUBLIC. In order to restrict access to the object, privileges have to be first revoked from PUBLIC before granting to specific users. Alternatively, the environment variable NODEFDAC can be set so that privileges are not granted to public by default when SQL objects are created. In an ANSI database, privileges are by default not granted to anyone upon object creation. Only the owner retains privileges on the object. Privileges have to be explicitly granted to specific users.

The following examples show the steps to be taken to grant SELECT and UPDATE privileges on a table in case of non-ANSI and ANSI databases.

Example 4-12 shows how to grant the SELECT, UPDATE privilege on tab1 to usr1 in a non-ANSI database. First, privileges have to be revoked from PUBLIC before doing the grant.

Example 4-12 Granting privileges to specific users in a non-ANSI database

```
DROP DATABASE db;
CREATE DATABASE db;

CREATE TABLE tab1 (i int);
SELECT * FROM systabauth WHERE tabid > 99;

REVOKE ALL ON tab1 FROM PUBLIC;
GRANT SELECT, UPDATE ON tab1 TO "usr1";
SELECT * FROM systabauth WHERE tabid > 99;
```

Example 4-13 shows how to grant SELECT,UPDATE privileges on tab1 to usr1 in an ANSI database. Compared to Example 4-12 on page 72, the revoke step is not needed in the case of the ANSI database.

Example 4-13 Granting privileges to specific users in an ANSI database

```
DROP DATABASE db;
CREATE DATABASE db WITH LOG MODE ANSI;

CREATE TABLE tab1 (i int);
SELECT * FROM "informix".systabauth WHERE tabid>99;

GRANT SELECT, UPDATE ON tab1 TO "usr1";
SELECT * FROM "informix".systabauth WHERE tabid>99;
COMMIT;
```

NODEFDAC

When the NODEFDAC environment variable is set to *yes*, privileges by default are not granted to PUBLIC when new tables are created in non-ANSI databases. Also, this environment variable prevents the EXECUTE privilege of a newly created user-defined routine (UDR) from being granted to PUBLIC. The NODEFDAC environment variable can also be set in the $INFORMIXDIR/etc/informix.rc file so that it takes effect for anyone who connects to any of the server instances managed out of that INFORMIXDIR.

4.1.5 Table privileges

This section explains the privileges available on the table and how to display them. Table 4-3 lists the IDS privileges available.

Table 4-3 Table privileges

Table privilege	Description
SELECT	Allows users to read data from the table. They can perform SELECT queries or create views on table. This is represented by the letter s or S in the tabauth column of the SYSTABAUTH catalog table.
UPDATE	Allows users to update data in the table. This is represented by the letter u or U in the tabauth column of the SYSTABAUTH catalog table.
INSERT	Allows users to insert rows in the table. This is represented by the letter i or I in the tabauth column of the SYSTABAUTH catalog table.

Table privilege	Description
DELETE	Allows users to delete rows from the table. This is represented by the letter d or D in the tabauth column of the SYSTABAUTH catalog table of the SYSTABAUTH catalog table.
INDEX	Allows users to create indexes on tables.This is represented by the letter x or X in the tabauth column of the SYSTABAUTH catalog table.
ALTER	Allows users to alter the table. This is represented by the letter a or A in the tabauth column of the SYSTABAUTH catalog table.
REFERENCES	Allows users to create referential constraints on the table. This is represented by the letter r or R in the tabauthtabauth column of the SYSTABAUTH catalog table.
UNDER	Allows users to create sub-tables under this table. This is represented by the letter n or N in the tabauth column of the SYSTABAUTH catalog table.

Example 4-14 shows the syntax for granting table-level privileges.

Example 4-14 Syntax for granting table-level privileges

```
GRANT privilege ON table_name TO user_list
GRANT privilege ON table_name TO user_list WITH GRANT OPTION
GRANT privilege ON table_name TO user_list AS grantor
```

The *tabauth* column of catalog table SYSTABAUTH contains the privileges granted. Each position in the tabauth column is assigned a privilege. If the tabauth column shows the privilege code in uppercase, it means the grantee has an option to grant the privilege to others. If the privilege code is in lowercase, it means that the grantee cannot grant the privilege to others.

Here is an example of tabauth column with all privilege positions assigned: su-idxarn.

- The first position is s for the SELECT privilege.
- The second position is u for the UPDATE privilege.
- The third position is a hyphen (-), which represents an ungranted privilege.
- The fourth position is i for the INSERT privilege.
- The fifth position is d for the DELETE privilege.
- The sixth position is x for the INDEX privilege.
- The seventh position is a for the ALTER privilege.
- The eighth position is r for the REFERENCES privilege.
- The last position is n for the UNDER privilege.

An uppercase letter at any of these positions signifies that the same privilege can be granted by the grantee to other users.

Displaying table privileges

The SYSTABAUTH table contains the grantor, grantee, tabid, and tabauth columns. We provide a few examples of displaying the user privileges on the table.

Example 4-15 shows a GRANT statement that grants all privileges to usr1. Also, the output of the SELECT statement is given. As shown, all privilege positions are occupied. The last position is for the UNDER privilege, and it is not automatically granted as part of ALL. It must be explicitly granted. The hyphen (-) in the third position stands for an ungranted privilege.

Example 4-15 Displaying table-level privileges

```
CREATE TABLE tab1 (i int);

GRANT ALL ON tab1 TO "usr1";

SELECT grantor, grantee, tabname, tabauth FROM "informix".systabauth A,
"informix".systables T  WHERE A.tabid > 99 AND A.tabid = T.tabid;
```

Output of the SELECT statement:

```
grantor dbauser
grantee  usr1
tabname  tab1
tabauth  su-idxar-
```

Example 4-16 shows the GRANT statement, granting the UPDATE privilege to usr8 with the WITH GRANT OPTION clause. This means that usr8 can further grant this privilege to other users. The output of the SELECT statement is also given. As shown, an uppercase U is granted for UPDATE, which means that usr8 can grant this privilege to other users.

Example 4-16 Displaying table-level privileges (WITH GRANT OPTION)

```
CREATE TABLE tab2 (name char(128), address char(1024));

GRANT UPDATE ON tab2 TO "usr8" WITH GRANT OPTION;

SELECT grantor, grantee, tabname, tabauth FROM "informix".systabauth A,
"informix".systables T  WHERE A.tabid > 99 AND A.tabid = T.tabid;
```

Output of the SELECT statement:

```
grantor dbauser
grantee  usr8
tabname  tab2
tabauth  -U-------
```

Example 4-17 shows that the UNDER privilege is granted to table tab_super to user usr5. The output of the SELECT statement shows that the last privilege position is occupied with lowercase letter n.

Example 4-17 Displaying table-level privileges (UNDER privilege)

```
CREATE ROW TYPE myrow (id int, name char(128), age int);
CREATE TABLE tab_super OF TYPE myrow;
GRANT UNDER ON tab_super TO "usr5";
SELECT grantor, grantee, tabname, tabauth FROM "informix".systabauth A,
"informix".systables T  WHERE A.tabid > 99 AND A.tabid = T.tabid;
```

Output of the SELECT statement:

```
grantor dbauser
grantee  usr5
tabname  tab_super
tabauth  --------n
```

Privileges on violations table

A user can execute the START VIOLATIONS TABLE statement in order to capture unique constraint and referential constraint violations, in violations and diagnostic tables. The owner of the target table becomes the owner of the violation table. The grantor of the initial set of privileges on the violation table is the same as the grantor of the privileges on the target table. The privileges on a violation table are derived from the privileges on the target table. Table 4-4 summarizes how the initial privileges on the violation table are derived from target table.

Table 4-4 How privileges are transferred to violation tor diagnostic tables

Privilege	**How the privilege is derived from the target table**
SELECT	User has the SELECT privilege on the *informix_tupleid*, *informix_optype*, and *informix_recowner* columns of the violations table, if the user has the SELECT privilege on any column of the target table. User has the SELECT privilege on any other column of the violations table if the user has the SELECT privilege on the same column of the target table.
UPDATE	User has the UPDATE privilege on the *informix_tupleid*, *informix_optype*, and *informix_recowner* columns of the violations table if the user has the UPDATE privilege on any column of the target table. User has the UPDATE privilege on any other column of the violations table if the user has the UPDATE privilege on the same column of the target table.
INSERT	User has the INSERT privilege on violations table if the user has INSERT, DELETE, or UPDATE privileges on any column of the target table.
DELETE	User has the DELETE privilege on violations table if user has the INSERT, DELETE, or UPDATE privileges on any column of the target table.
INDEX	If user has the INDEX privilege on the target table, then the user has INDEX privilege on the violations table.
REFERENCES	The the REFERENCES privilege is not granted on the violations table, as the user cannot added referential constraints on the violation table.
ALTER	The ALTER privilege is not granted on the violations table, as users cannot alter them.

After the violations table is started, revoking a privilege on the target table from a user does not automatically revoke the same privilege on the violations table from that user. Instead, you must explicitly revoke the privilege on the violations table from the user. If you have fragment-level privileges on the target table, you have the corresponding fragment-level privileges on the violations table.

Now we give an example of starting a violations table and displaying privileges on violations and diagnostic tables.

Example 4-18 shows that violations table is started on table tab2. The second INSERT statement caused unique constraint violations, and the row will be written to the violations table.

Example 4-18 Script that generates records for violation table

```
CREATE DATABASE db WITH LOG MODE ANSI;

GRANT CONNECT TO "usr1", "usr2", "usr3", "usr4", "usr5";

CREATE TABLE tab2 (i INTEGER UNIQUE, j INTEGER);
GRANT ALL ON tab2 TO "usr4";
GRANT INSERT, INDEX ON tab2 TO "usr5";
GRANT SELECT, DELETE, UPDATE ON tab2 TO "usr6";

START VIOLATIONS TABLE FOR tab2 USING tab2_vio, tab2_diag MAX ROWS 50;
SET CONSTRAINTS FOR tab2 FILTERING;


INSERT INTO tab2 VALUES (20, 30);
INSERT INTO tab2 VALUES (20, 30);

SELECT * FROM "informix".SYSTABAUTH WHERE tabid > 99;
```

Example 4-19 displays rows from SYSTABAUTH. The tables with tabid 101 and 102 correspond to diagnostic and violation tables, respectively. As shown, privileges in the diagnostic and violation tables are derived correctly. When the target table also has ALTER and REFERENCES privileges on usr4, these privileges are not propagated to violations and diagnostic tables.

Example 4-19 Displaying how privileges are propagated to violations table

```
grantor dbauser
grantee  usr4
tabid    100
tabauth  su-idxar-

grantor dbauser
grantee  usr5
tabid    100
```

```
tabauth  ---i-x---

grantor  dbauser
grantee  usr6
tabid    100
tabauth  su--d----

grantor dbauser
grantee  usr4
tabid    102
tabauth  su-idx---

grantor dbauser
grantee  usr4
tabid    101
tabauth  su-idx---

grantor dbauser
grantee  usr5
tabid    102
tabauth  ---i-x---

grantor dbauser
grantee  usr5
tabid    101
tabauth  ---i-x---

grantor dbauser
grantee  usr6
tabid    102
tabauth  su--d----

grantor dbauser
grantee  usr6
tabid    101
tabauth  su--d----

9 row(s) retrieved.
```

4.1.6 Column privileges

When a table-level privilege is granted, it is implicitly granted on all columns of the table. The column-level privilege helps you to further restrict the scope of user activity to specific columns. The privileges that are applicable at the column level are SELECT, UPDATE, and REFERENCES only. When column-level privileges are granted to users on a table, the third privilege position in the SYSTABAUTH system catalog table (tabauth column) has the value of an asterisk (*), indicating that column privileges exist on the table.

Example 4-20 shows the syntax for granting column privileges.

Example 4-20 Syntax for granting column-level privileges

```
GRANT privilege (column_list) ON table_name TO user_list
GRANT privilege (column_list) ON table_name TO user_list WITH GRANT OPTION
GRANT privilege (column_list) ON table_name TO user_list AS grantor
```

Note: You cannot revoke privileges granted to a user on specific columns. If you want to change the columns allowed to a certain user, you must first REVOKE the privilege in full, then GRANT it on the new list of columns.

Displaying column-level privileges

The column-level privileges are stored in the SYSCOLAUTH system catalog table. The *colauth* column represents the privileges granted on a particular column to a specific user. There are three positions available in this value. The first position is for the SELECT privilege, the second position is for the UPDATE privilege, and the third position is for the REFERENCES privilege. If a colauth privilege code is in uppercase, then the grantee can grant this privilege to others. If the privilege code is in lower case then the grantee cannot grant it to others.

Example 4-21 takes a fictitious emp_data table and grants privileges on specific columns depending on whether the employee is an HR person (emp_hr), payroll person (emp_payroll), manager (emp_mgr), or an staff member (empA).

Example 4-21 Script that demonstrates granting column-level privileges

```
CREATE TABLE emp_data (
        id              INTEGER,
        firstName       CHAR(32),
        lastName        CHAR(32),
        salary          FLOAT,
        perf_grade      CHAR(3)
        );

GRANT SELECT, INSERT ON emp_data TO "emp_hr";

GRANT SELECT (firstName, lastName) ON emp_data TO "empA";
GRANT SELECT (salary), UPDATE (salary) ON emp_data TO "emp_payroll";
GRANT SELECT (perf_grade), UPDATE (perf_grade) ON emp_data TO
"emp_mgr";
GRANT REFERENCES(id) ON emp_data TO "emp_hr";

SELECT * FROM "informix".syscolauth where tabid > 99;
```

Example 4-22 shows the output of the SELECT statement in Example 4-21 on page 80.

Example 4-22 Displaying column-level privileges

```
grantor  rcpachip
grantee  empA
tabid    100
colno    2
colauth  s--

grantor  rcpachip
grantee  empA
tabid    100
colno    3
colauth  s--

grantor  rcpachip
grantee  emp_payroll
tabid    100
colno    4
colauth  su-

grantor  rcpachip
grantee  emp_mgr
tabid    100
colno    5
colauth  su-

grantor  rcpachip
grantee  emp_hr
tabid    100
colno    1
colauth  --r

5 row(s) retrieved.
```

4.1.7 Fragment privileges

You can use the GRANT FRAGMENT statement to grant INSERT, UPDATE, and DELETE privileges on individual fragments of a table. The fragment privileges are only applicable to fragments created by expression-based fragmentation.

Example 4-23 shows the syntax of the GRANT FRAGMENT statement. The *priv_list* is a comma-separated list of fragment-level privileges, which is one of INSERT, UPDATE, DELETE, or ALL. The *frag_list* is a comma-separated list of fragments. The *user_list* is a comma-separated list of users to whom privileges are being granted, including PUBLIC.

Example 4-23 Syntax for granting fragment-level privileges

```
GRANT FRAGMENT priv_list ON table_name (frag_list) TO user_list
GRANT FRAGMENT priv_list ON table_name (frag_list) TO user_list WITH GRANT
OPTION
GRANT FRAGMENT priv_list ON table_name (frag_list) TO user_list AS grantor
```

Example 4-24 shows the syntax for corresponding REVOKE statements.

Example 4-24 Syntax for revoking fragment-level privileges

```
REVOKE FRAGMENT priv_list ON table_name (frag_list) FROM user_list;
REVOKE FRAGMENT priv_list ON table_name (frag_list) FROM user_list AS
revoker;
```

Displaying fragment-level privileges

The privileges pertaining to fragments are stored in the SYSFRAGAUTH system catalog table. The fragauth column shows the privileges granted to the user on a fragment. There are six privilege positions available in this field. However, only three privileges (INSERT, UPDATE, AND DELETE) can be granted on fragments. Also, privilege code in uppercase means the grantee can grant the privilege to other users. Privilege code in lower case means that the grantee cannot grant this privilege to other users.

Example 4-25 demonstrates the use for GRANT FRAGMENT statement.

Example 4-25 Usage of GRANT FRAGMENT statement

```
CREATE TABLE emp_data (
        id              INTEGER,
        firstName       CHAR(32),
        lastName        CHAR(32),
        salary          FLOAT,
        perf_grade      CHAR(3)
        ) FRAGMENT BY EXPRESSION
        id < 50 in dbsp1,
        id >= 50 in dbsp2;

GRANT FRAGMENT INSERT, UPDATE, DELETE ON emp_data (dbsp1 ) TO “usr3”;
GRANT FRAGMENT INSERT, UPDATE, DELETE ON emp_data (dbsp2 ) TO “usr4”;

SELECT * FROM "informix".sysfragauth where tabid > 99;
```

Example 4-26 shows the output of the SELECT statement in Example 4-25 on page 82. The fragauth column shows the privileges granted. An ungranted privilege is represented by a hyphen (-).

Example 4-26 Displaying fragment-level privileges

```
grantor   rcpachip
grantee   usr3
tabid     100
fragment  dbsp1
fragauth  -uid--

grantor   rcpachip
grantee   usr4
tabid     100
fragment  dbsp2
fragauth  -uid--

2 row(s) retrieved.
```

Duration of fragment-level privileges

Fragment-level privileges that are granted to fragments of a table cease to exist under one or more of the following conditions:

- The fragmentation strategy of the table is changed from expression-based to another strategy such as round-robin.
- A fragment is dropped from the table, which results in the privileges on the fragment being dropped.
- If the expression on which fragmentation is based is changed, then also the fragment privileges are dropped and the user assumes default table privileges.

4.1.8 TYPE privileges

IDS supports user-defined data types (UDTs). This section discusses the privileges supported on IDS and how to display them. Table 4-5 lists IDS TYPE privileges.

Table 4-5 TYPE privileges

Type privilege	Description
USAGE	Authorization to use the named data type
UNDER	Authorization to use the named data type as a super-type in the type hierarchy

Example 4-27 shows the syntax for granting privileges on user-defined types. *priv_list* indicates list of privileges (USAGE, UNDER) and *user_list* indicates the list of users including PUBLIC.

Example 4-27 Syntax for granting TYPE privileges

```
GRANT priv_list ON TYPE udt TO user_list;
GRANT priv_list ON TYPE udt TO user_list WITH GRANT OPTION;
GRANT priv_list ON TYPE udt TO user_list AS grantor;
```

Example 4-28 shows the syntax for REVOKE. The treatment of the RESTRICT and CASCADE clause for revoke is done separately.

Example 4-28 Syntax for revoking TYPE privileges

```
REVOKE priv_list ON TYPE udt FROM user_list [RESTRICT | CASCASDE]
REVOKE priv_list ON TYPE udt FROM user_list [RESTRICT | CASCADE] AS
revoker
```

Example 4-29 demonstrates granting and revoking USAGE and UNDER privileges on a named ROW TYPE to usr3.

Example 4-29 Usage of GRANT on TYPEs

```
CREATE ROW TYPE myrow (id INT, name CHAR(128), age INT);

GRANT USAGE ON TYPE myrow TO "usr3";
GRANT UNDER ON TYPE myrow TO "usr3";

SELECT * FROM "informix".SYSXTDTYPEAUTH;
```

Displaying TYPE privileges

The TYPE-level privileges are stored in the system catalog table SYSXTDTYPEAUTH. The *auth* column shows the privileges granted to a TYPE. There are two privilege positions available. The first position is for the USAGE privilege. The value of n in this position signifies that the USAGE privilege is granted and the grantee cannot grant this privilege to other users. The value of N in this position indicates that the USAGE privilege is granted and the grantee in turn can grant this privilege to other users. The second position is for the UNDER privilege. The values that are possible are u or U depending on whether grantee can further grant this privilege to other users. A hyphen (-) in either of the positions indicates an ungranted privilege.

Example 4-30 provides the output of the SELECT statement in the previous example. As shown, the auth column with a value of nu indicates that both USAGE and UNDER privileges are granted to usr3 and that usr3 cannot grant these privileges to other users.

Example 4-30 Displaying TYPE privileges

```
grantor dbausr
grantee  usr3
type     2048
auth     nu
```

Revoking TYPE privileges

Example 4-31 demonstrates revoking TYPE-level privileges.

Example 4-31 Revoking TYPE-level privileges

```
REVOKE USAGE ON TYPE myrow FROM "usr3";
REVOKE UNDER ON TYPE myrow FROM "usr3";
```

CASCADE and RESTRICT options

If the UNDER privilege is revoked from a super-TYPE, then all the sub-TYPEs created are dropped. The same effect is obtained with the CASCADE option of the revoke statement. However, with RESTRICT clause, the revoke statement returns an error and the UNDER privilege is not revoked.

Example 4-32 shows that the table tab2 and row TYPE addr are created. The user usr4 is granted USAGE and UNDER privileges on TYPE addr.

Example 4-32 Script that grants USAGE and UNDER privileges

```
CREATE DATABASE db WITH LOG MODE ANSI;
GRANT RESOURCE TO "usr4";

CREATE TABLE tab2 (i INTEGER, j INTEGER);
GRANT SELECT ON tab2 to "usr4";

CREATE ROW TYPE addr (name CHAR(32), street CHAR (56), city CHAR(32));
GRANT USAGE, UNDER ON TYPE addr TO "usr4";
COMMIT;
```

Example 4-33 shows that the row TYPE sub_addr is created under addr. This script is executed by usr4.

Example 4-33 Script that creates a sub-TYPE

```
-- Execute this script as usr4
DATABASE db;

CREATE VIEW vtab2 AS SELECT i FROM "dbauser".tab2;

SELECT * FROM "informix".systabauth  WHERE  tabid IN (
        SELECT tabid FROM "informix".systables WHERE tabid>99);

CREATE TABLE mytab OF TYPE "dbauser".addr;
CREATE ROW TYPE sub_addr(zip INTEGER)  UNDER "dbauser".addr;

SELECT * FROM "informix".sysxtdtypeauth WHERE  type IN (
        SELECT extended_id FROM "informix".sysxtdtypes );
```

Example 4-34 shows the REVOKE statement with the RESTRICT clause. The revoke returns an error. If the CASCADE option is specified, then the UNDER privilege is revoked and the sub-TYPE is dropped.

Example 4-34 Usage of REVOKE statement with RESTRICT clause

```
>dbaccess db -

Database selected.

> REVOKE UNDER ON TYPE addr FROM "usr4" RESTRICT;
```

```
  887: Cannot revoke because of dependent privileges, views or
constraints.
Error in line 1
Near character position 45
>
```

4.1.9 VIEW privileges

When a VIEW is created, IDS checks the privileges on the underlying tables before granting privileges to the creator/owner of the VIEW. If the user has the SELECT privilege on the underlying table, then the VIEW is created and the SELECT privilege is granted on the VIEW to the user. If the user does not have the SELECT privilege on the underlying table, then the VIEW is not created. If the user also has INSERT, UPDATE, and DELETE privileges on the underlying table, then the same privileges are granted on the VIEW to the user.

If you are using a VIEW, then your privileges with respect to the VIEW are checked, and not the underlying tables. The creator of the VIEW typically grants privileges on the VIEW to other users. The privileges of a VIEW are stored in the SYSTABAUTH and SYSCOLAUTH catalog tables along with other table-level and column-level privileges. Users cannot use a VIEW to gain access to any privileges that they did not already have.

In case a VIEW is based on multiple tables, the privileges on a VIEW are the common privileges across all base tables.

Example 4-35 shows that usr4 is granted the SELECT privilege on tab1 and SELECT, UPDATE privileges on tab2. Example 4-35 shows that usr4 creates a VIEW using tab1 and tab2 as base tables.

Example 4-35 Script to demonstrate VIEW privileges

```
CREATE DATABASE view_db WITH LOG MODE ANSI;

GRANT CONNECT TO "usr4";

CREATE TABLE tab1 (i INT, j CHAR(30));
CREATE TABLE tab2 (k INT);

GRANT SELECT on tab1 TO "usr4";
GRANT SELECT, UPDATE ON tab2 TO "usr4";
```

Example 4-36 shows that VIEW v4_2 is created by usr4 on base tables tab1 and tab2.

Example 4-36 Script that creates a VIEW

```
DATABASE view_db;
CREATE VIEW v4_2 AS SELECT i,k FROM tab1 t1, tab2 t2 where t1.i=t2.k;
```

After the VIEW is created, usr4 only has the SELECT privilege on the VIEW. Also, usr4 has the UPDATE privilege on the second column of the VIEW. The VIEW's column-level privileges are stored in the SYSCOLAUTH catalog table.

When the SELECT privilege on the base table is revoked from the user, then the VIEW that is based on the table also gets dropped. Because you cannot alter or index a VIEW, ALTER and INDEX privileges are never granted on a VIEW. Permissions on remote tables are not automatically propagated to VIEWs on these tables.

CASCADE and RESTRICT options

If the SELECT privilege on the base table is revoked from the user with the RESTRICT clause, then revoke returns an error instead of dropping the privilege and dependent VIEW. If the REVOKE is done with the CASCADE clause or if no clause is specified, then the privilege on the base table is dropped along with the dependent VIEW.

Example 4-37 shows the effect of doing the revoke with the RESTRICT option. This example is based on Example 4-32 on page 86 and Example 4-33 on page 86.

Example 4-37 REVOKE with RESTRICT option

```
>dbaccess db -

Database selected.

> REVOKE SELECT ON tab2 FROM "usr4" RESTRICT;

  887: Cannot revoke because of dependent privileges, views or
constraints.
Error in line 1
Near character position 41
```

4.1.10 SEQUENCE privileges

SEQUENCE privileges are similar to table-level privileges. The difference is that only SELECT and ALTER privileges are granted on SEQUENCEs. Also, similar to table-level privileges, SEQUENCE privileges are stored in the system catalog table SYSTABAUTH. Only the privilege bits for SELECT and ALTER are turned on when these privileges are granted to users. The remaining bits show a hyphen (-), which stands for ungranted privilege.

A user with SELECT privilege can use the SEQUENCE in queries. A user with the ALTER privilege can alter the SEQUENCE using the ALTER SEQUENCE or RENAME SEQUENCE commands. SEQUENCEs are created only by DBA-privileged users.

Both ALTER and SELECT privileges on a SEQUENCE can be granted to a user or a role.

4.1.11 Routine privileges

The privileges needed to create a routine are different from privileges needed to use the routine. The following privileges are needed to create a routine or procedure:

- DBA privilege or RESOURCE privilege on the database
- Language-level privilege on the language in which the routine is written
- EXTEND role privilege in case the IFX_EXTEND_ROLE configuration parameter is set to 1 and the routine that is created is a Java or C UDR

A routine creator by default has the privilege to execute the routine. If another user has to execute the routine, then the creator of the routine needs to grant the EXECUTE privilege to the user.

Table 4-6 lists IDS routine privilege.

Table 4-6 Routine privileges

Routine privilege	Description
EXECUTE	Authorization to execute a procedure.

When an user-defined routine (UDR) is created in a non-ANSI database, by default the EXECUTE privilege is granted to PUBLIC. When a UDR is created in an ANSI database, privileges have to be explicitly granted to specific users.

Example 4-38 provides the syntax for granting and revoking privileges from UDRs.

Example 4-38 Syntax for granting routine privileges

```
GRANT privilege ON procname TO user_list;
GRANT privilege ON procname TO user_list WITH GRANT OPTION;
GRANT privilege ON procname TO user_list AS grantor;

REVOKE privilege ON procname FROM user_list;
REVOKE privilege ON procname FROM user_list AS revoker;
```

Displaying routine-level privileges

The system catalog table SYSPROCAUTH stores the information pertaining to grants. The *procauth* column indicates whether execute permission is available to the grantee. There is one privilege position available. The presence of a lowercase e indicates that the grantee has the EXECUTE privilege over the procedure. The presence of an uppercase E indicates that the grantee can in turn grant this privilege to others.

Example 4-39 displays rows from the SYSPROCAUTH catalog table. As shown, there is only one privilege position. The availability of the EXECUTE privilege to the grantee is indicated by a lowercase e, which means that usr6 cannot grant the privilege to other users. An ungranted privilege is represented by a hyphen.

Example 4-39 Displaying routine-level privileges

```
GRANT EXECUTE ON myproc TO "usr6";
```

Output from SYSPROCAUTH table:

```
grantor   rcpachip
grantee   USR6
procid    347
procauth  e
```

Example 4-40 shows the revoke statement for revoking the EXECUTE privilege granted to usr6 in Example 4-39.

Example 4-40 Revoking routine-level privilege

```
REVOKE EXECUTE ON myproc FROM "usr6";
```

Registering a routine

The qualifying criteria for registering a routine in the database is:

- Any user with a DBA privilege can register a routine in the database.
- A user who does not have DBA privilege needs the RESOURCE privilege to register a routine in the database.

A DBA or the routine owner can cancel the registration by executing the DROP PROCEDURE or DROP FUNCTION statements.

Using a UDR as trigger action

Let us assume that a UDR as a trigger action and the owner of the trigger has the EXECUTE privilege on the UDR WITH GRANT OPTION. In this scenario, any user can invoke the trigger action provided that the user also has the necessary privileges on the tables on which the trigger is based.

Example 4-41 shows that the EXECUTE privilege is granted on a routine *ins_tab3()* to trigger the owner, usr3, with WITH GRANT OPTION. Users who invoke the trigger action get the EXEUCUTE privilege on the routine from the trigger owner.

Example 4-41 Using UDR as trigger action

```
CREATE DATABASE db WITH LOG MODE ANSI;
GRANT CONNECT TO "usr2";
GRANT CONNECT TO "usr3";

CREATE TABLE tab2 (i INTEGER, j INTEGER);
CREATE TABLE tab3 (i INTEGER);
CREATE SEQUENCE myseq START WITH 0 INCREMENT BY 5 MAXVALUE 5000;
CREATE PROCEDURE ins_tab3 ()
   INSERT INTO tab3 VALUES (myseq.NEXTVAL);
END PROCEDURE;

GRANT INSERT ON tab2 TO "usr2";
GRANT SELECT, INSERT ON tab3 TO "usr2";
GRANT SELECT ON myseq TO "usr2" WITH GRANT OPTION;
GRANT EXECUTE ON ins_tab3 TO "usr3" WITH GRANT OPTION;

CREATE TRIGGER "usr3".tab2_trig
                INSERT ON tab2 BEFORE
                        (EXECUTE PROCEDURE "dbauser".ins_tab3())
COMMIT;
```

Example 4-42 is executed by usr3. The script inserts data into tab2, thus invoking the trigger action.

Example 4-42 Script that causes trigger invocation

```
DATABASE db;

INSERT INTO "dbauser".tab2 VALUES (30,40);

SELECT * FROM "dbauser".tab3;
```

User-defined aggregates

There are no explicit privileges associated with user-defined aggregates. The privileges on the functions used in defining the aggregate determine who can use the aggregate. Only users who have DBA and RESOUCE privileges can create functions. In an ANSI database, the owner needs to grant explicit EXECUTE privilege to other users. These users can use the function and thus the related aggregate.

4.1.12 Language privileges

In IDS, user-defined routines (UDRs) were written in SPL or in any of the external languages such as C and Java. In order to create a routine in a particular language, the user needs the USAGE privilege for that language. By default, language usage privileges on SPL are available to the user informix and the user holding the DBA privilege to that database. However, only the user informix can grant the language usage privilege to others. The user with the DBA privilege cannot grant this privilege to others. The USAGE privilege to create SPL routines is granted to PUBLIC by default.

Table 4-7 lists the IDS language privilege.

Table 4-7 Language privileges

Language privilege	Description
USAGE	Authorization to write a routine in a particular language

Syntax of grant and revoke

Example 4-43 gives the syntax of grant/revoke. This example shows how to grant and revoke the USAGE privilege on different languages. The languages supported are SPL, C, and Java.

Example 4-43 Syntax for granting language privileges

```
GRANT privilege ON LANGUAGE lang TO user_list;
GRANT privilege ON LANGUAGE lang TO user_list WITH GRANT OPTION;
GRANT privilege ON LANGUAGE lang TO user_list As grantor;

REVOKE privilege ON LANGUAGE lang FROM user_list;
REVOKE privilege ON LANGUAGE lang FROM user_list AS revoker;
```

Example 4-44 shows the languages supported from the SYSROUTINELANGS system catalog table.

Example 4-44 Displaying languages

```
> select * from "informix".sysroutinelangs;


langid       0
langname     builtin
langinitfunc udrlm_builtin_init
langpath
langclass

langid       1
langname     c
langinitfunc udrlm_clang_init
langpath
langclass

langid       2
langname     spl
langinitfunc udrlm_spl_init
langpath
langclass

langid       3
langname     java
langinitfunc udrlm_java_init
langpath     $INFORMIXDIR/extend/krakatoa/lmjava.so
langclass    jvp
```

```
langid        4
langname      client
langinitfunc  udrlm_client_init
langpath
langclass

5 row(s) retrieved.
```

Displaying language privileges

Example 4-45 displays the language privilege for Example 4-44 on page 93. As shown, the USAGE privilege (u) is granted to usr5 for langid 2, which is the SPL language.

Example 4-45 Displaying language privileges

```
grantor   rcpachip
grantee   usr5
langid    2
langauth  u
```

4.1.13 DBA-privileged routines

The UNIX operating system provides SUID and SGID bits in its file permissions. A program with the owner root and the SUID bit turned on lets any user executing this program assume the privileges of super-user or root for the duration of the execution of the program. The SUID and SGID bits can only be turned on by the owner or group of the program, respectively. This is a very powerful feature, as it lets normal users perform privileged operations defined by the administrator. The scope of the privileged operation is defined entirely by the person writing the program who is potentially an administrator. As the user has permissions to only execute the program and not modify it, there is no scope for user tampering. Thus, security is achieved while letting user programs perform privileged operations. This is done without granting privileges to the user explicitly.

In IDS, a similar concept exists in the form of DBA-privileged routines. We also refer to these routines, hereafter, as DBA procedures or DBA routines. A DBA procedure has many benefits. Some of them are:

- Allows an under-privileged user to acquire DBA privileges for the duration of the procedure execution.
- Enables and supports the least privilege principle. The privilege of a user is enhanced only for the duration of the procedure. After the procedure

execution completes, the user loses privileges that come with procedure execution, but retains old privileges.

- Enhances security by allowing the DBA to maintain tight control over how a high-privileged operation is written and executed.
- DBA controls who can execute the procedure by granting EXECUTE privilege to specific users.

Example 4-46 illustrates one benefit of DBA procedures. The users susr1, usr2, usr3, and usr4 have only the CONNECT privilege on the database. However, they are granted the EXECUTE privilege on the procedure *create_user_tables*. This procedure creates a table that the user uses to store certain user-specific data across sessions. As the user does not have the RESOURCE privilege, there is no way that the user can create a table outside of the procedure. Thus, the DBA has accomplished the twin goals of providing security and functionality. Security is obtained, as the user does not have the RESOURCE privilege to create tables inadvertently, thus saving disk space. Functionality is achieved by allowing the user to create a table using the DBA procedure. The resulting table created is owned by the user. As shown, the table created depends on the user executing the routine.

Example 4-46 DBA-privileges routine

```
CREATE DBA PROCEDURE create_user_tables ()
        RETURNING INTEGER;
    IF USER = 'usr1' THEN
       CREATE  TABLE usrtab_1 (id INT, usrdata VARCHAR(255));
    ELIF USER = 'usr2' THEN
       CREATE  TABLE usrtab_2 (id INT, usrdata VARCHAR(255));
    ELIF USER = 'usr3' THEN
       CREATE  TABLE usrtab_3 (id INT, usrdata VARCHAR(255));
    ELIF USER = 'usr4' THEN
       CREATE  TABLE usrtab_4 (id INT, usrdata VARCHAR(255));
    END IF
END PROCEDURE;
```

Example 4-47 shows the execution of the procedure by the user. The table created is specific to the user executing the procedure.

Example 4-47 Execution of a DBA privileges routine

```
EXECUTE PROCEDURE "dbauser".create_user_tables();
```

4.1.14 SETSESSIONAUTH privilege

The SETSESSIONAUTH privilege is granted by DBSECADM to users so that users can execute the SET SESSION AUTHORIZATION statement. The SET SESSION AUTHORIZATION statement is executed by a user to become another user, thus gaining both the DAC and LBAC privileges of the (new) user.

Example 4-48 gives the syntax of the GRANT SETSESSIONAUTH. The SETSESSIONAUTH privilege is granted by DBSECADM to a user on a user. This means that the user in the TO list can do SET SESSION AUTHORIZATIONL to any user in the ON list. In this example, the SETSESSIONAUTH privilege is granted to users or roles in user_role_list. These users can do SET SESSION AUTHORIZATION to any user in the ON *user_list*.

Example 4-48 Syntax for granting SETSESSIONAUTH privilege

```
GRANT SETSESSIONAUTH ON [USER] user_list TO [USER | ROLE] user_role_list;
```

Similarly, the syntax for revoke is as given in Example 4-49. Example 4-49 shows the REVOKE statement syntax, which is the converse of Example 4-48.

Example 4-49 Syntax for revoking SETSESSIONAUTH privilege

```
REVOKE SETSESSIONAUTH ON [ USER] user_list FROM [USER | ROLE] user_role_list;
```

Example 4-50 shows a few examples of granting and revoking the SETSESSIONAUTH privilege.

Example 4-50 Granting and revoking SETSESSIONAUTH to users and PUBLIC

```
CREATE DATABASE db WITH LOG MODE ANSI;

CREATE ROLE "myrole";

GRANT SETSESSIONAUTH ON PUBLIC TO "usr4";
GRANT SETSESSIONAUTH ON "usr3" TO USER "usr5";
GRANT SETSESSIONAUTH ON "usr2" TO ROLE "myrole";
GRANT SETSESSIONAUTH ON "usr4" TO "usr7";
GRANT SETSESSIONAUTH ON USER "usr8" TO "usr7";
GRANT SETSESSIONAUTH ON "usr4", "usr5" TO "usr7", "myrole";


REVOKE SETSESSIONAUTH ON PUBLIC FROM "usr4";
REVOKE SETSESSIONAUTH ON "usr3" FROM USER "usr4";
REVOKE SETSESSIONAUTH ON "usr2" FROM ROLE "myrole";
REVOKE SETSESSIONAUTH ON "usr4" FROM "usr7";
REVOKE SETSESSIONAUTH ON USER "usr8" FROM "usr7";
REVOKE SETSESSIONAUTH ON "usr4", "usr5" FROM "usr7", "myrole";
COMMIT;
```

More details on the SETSESSIONAUTH privilege are presented from the LBAC perspective in 4.2.13, “SETSESSIONAUTH privilege (in the context of LBAC)” on page 141.

4.1.15 Roles

A role can be defined as a classification of a work task. For example, you can define the manager role in a company. The manager role comes with a set of responsibilities such as hiring employees, doing performance reviews, negotiating salaries, approving bonuses, and so on. These are common responsibilities that a typical manager performs across all departments. In order to fulfill these responsibilities, managers need certain privileges such as access to employee records and other data privy to managers only. When an employee becomes a manager or when a new manager is hired into the company, the employee assumes the manager role and gets along with it the privileges associated with that role.

IDS provides the same concept in the DBMS in the form of roles. A role can only be created by a DBA privileged user. Privileges are granted to roles by the DBA and roles are assigned to users. A user can assume a role depending on the task being performed and relinquish the role when the task is completed. A user can be granted more than one role, but the user can assume only one role at any given time. Thus, depending on the operations to be performed, the user assumes a role and acquires the privileges that come with it. The user switches the role in order to perform some other operation that requires a different set of privileges. Thus, roles enable the least privilege principle, which is explained in the following section.

Least privilege principle

Roles enable the least privilege mechanism to access the data. The administrator may have granted the user certain privileges to perform some operations. The user may only need a subset of them to perform a particular SQL operation. The least privilege principle stipulates that the user assume only a minimum set of privileges to perform a particular operation. This is only possible if the administrator constantly revokes and grants new privileges before the user performs an operation. This is difficult to administer. This is where roles come into play and fill the gap. The administrator predefines the roles based on operations that need to be performed and grants privileges to roles. The admin grants roles to users depending on user responsibilities. Users assume roles depending on the activity that they are performing and can easily switch between roles. The least privilege principle enhances the security of the system because by assuming only the privileges to do a particular task, users perform only the operations that they intend to perform and will not erroneously modify other data.

The following example explains the principle and usage of the least privilege in a departmental setting. A departmental IT system contains an employee table, which has columns pertaining to employee name, address, age, salary, and perf_grade. The columns such as name, address, and age can be updated by the employee, whereas the columns salary and perf_grade can only be updated by the manager. The administrator creates the roles of *employee* and *manager* and grants the UPDATE privilege on the name, address, and age columns to the employee role, and the UPDATE privilege on all columns to the manager role. Assume that employee records are protected from each other using LBAC (LBAC is covered in a later section). User John Doe is both a manager and an employee, so the admin grants both these roles to John Doe. Users Jay Smith and Ron Wood are employees, so the admin grants the employee role to these users. When Jay Smith or Ron Wood wants to modify their data in the employee table, they assume the role of employee. If the manager John Doe wants to update the salary data of his employees, he assumes the manager role. However, if he wants to update his own information, he assumes the employee role even though he can do this with the manager role. Thus, user John Doe uses the least privilege needed to do a particular task.

The least privilege principle can be enforced in a transparent manner, reducing the burden on the user. Application segments can be written with different roles in mind. For example, a specific role can be set at the beginning of a stored procedure depending on the task being performed. When a user executes this procedure, he assumes the privileges granted to the role especially if the role is granted to the owner of the procedure WITH GRANT OPTION. The verification that if the least privilege principle is followed can be done by checking the audit records.

The following describes the syntax and semantics of role creation, granting and setting of roles, default roles, revoking roles from users, revoking privileges from roles, and dropping roles with the help of a sample scenario.

Sample scenario

Consider company XYZ Inc. There are employees, managers, and executives in the company. They are in HR, finance, sales, marketing, and engineering departments. For the purpose of illustration, consider the following roles in the company:

- payroll_admin
- employee
- manager
- executive
- finance_vp

The following tables are defined in an emp_db database:

- employee (Name, Address, perf_grade, salary)
- sales(....)
- finance(.....)

An employee role has the SELECT and UPDATE privileges over the name and address columns of employee table.

A payroll_admin role has the UPDATE privilege over the salary column of the employee table and the SELECT privilege over all columns.

A manager has the SELECT privilege over all columns of the employee table and the UPDATE privilege over the perf_grade column.

Assume that usr1 is a manager, usr2 and usr3 are employees, and usr4 is payroll_admin.

Creating roles

A role is applicable only within the database in which it is created.

Example 4-51 shows the syntax of the CREATE ROLE statement. The *role_name* is the name of the role.

Example 4-51 Syntax for creating a role

```
CREATE ROLE role_name;
```

When you create a role, the role name is added to the SYSUSERS catalog table. The role name cannot be the name of any user known to the operating system nor can it be the name of a user or role known to the database server. The role name cannot be the grantor or grantee in the system catalog tables SYSTABAUTH, SYSCOLAUTH, SYSFRAGAUTH, SYSLANGAUTH, SYSPROCAUTH, SYSXTDTYPEAUTH, or SYSROLEAUTH. The role name also cannot match the built-in role names such as EXTEND and DBSECADM.

Example 4-52 demonstrates the creation of different roles in the sample scenario.

Example 4-52 Examples statements for creating roles

```
CREATE ROLE "payroll_admin";
CREATE ROLE "employee";
CREATE ROLE "manager";
```

Example 4-53 shows the entries in the SYSUSERS system catalog table for new roles created. The username column contains the role name, and the value G in usertype column signifies that it is a role.

Example 4-53 Displaying roles

```
username  payroll_admin
usertype  G
priority  5
password
defrole

username  employee
usertype  G
priority  5
password
defrole

username  manager
usertype  G
priority  5
password
defrole
```

Granting privileges to roles

The syntax for granting privileges to roles is equivalent to the syntax of granting privileges to users. The role name has to be specified in place of the user name. The only difference is that you cannot use the WITH GRANT OPTION clause or the AS *grantor* clause while granting privileges to roles. This is understandable because a role is not a real-entity even though there are privileges associated with the role. It is the user who assumes a role, and if the WITH GRANT OPTION is used while granting a privilege or role to the user, then the user can grant it to others. A role cannot hold database-level privileges.

Only a DBA can grant a privilege to a role.

Example 4-54 shows the syntax for granting privileges to roles for various grant statements. *priv_list* indicates the list of privileges, *role_list* indicates the list of role, *table* is the name of the table, *col_list* is the list of columns, *udt* is a user-defined TYPE, *lang* is a language, *proc_name* is the name of a routine, and *frag_list* is the list of fragments.

Example 4-54 Syntax for granting privileges to roles

```
-- Granting table-level privilege to a role
GRANT priv_list ON table TO role_list;

-- Granting Column-level privileges to a role
GRANT priv_list (col_list) ON table TO role_list;

-- Granting Type-level privileges to a role
GRANT priv_list ON TYPE udt TO role_list;

-- Granting language-level privielges to a role
GRANT priv_list ON LANGUAGE lang TO role_list;

-- Granting routine-level privileges to a role
GRANT priv_list ON proc_name TO role_list;

-- Granting fragment-level privileges to a role
GRANT FRAGMENT priv_list ON table (frag_list) TO role_list;
```

Example 4-55 shows the granting of privileges to roles defined in the sample scenario.

Example 4-55 Examples for granting privileges to roles

```
GRANT SELECT ON emp_tab TO "payroll_admin", "manager";
GRANT SELECT (name, address) ON emp_tab TO "employee";
GRANT UPDATE (name, address) ON emp_tab TO "manager", "employee";
GRANT UPDATE (perf_grade) ON emp_tab TO "manager";
GRANT UPDATE (salary) ON emp_tab TO "payroll_admin";
```

As shown in Example 4-55, privileges are granted to roles. A user who is both a manager and an employee sets an appropriate role to perform activities. If the perf_grade column has to be updated, then the user assumes the manager role. If the name and address columns need to be updated, then the user assumes the employee role. Setting roles is described in a later section.

Example 4-56 gives the syntax for the corresponding REVOKE statements.

Example 4-56 Syntax for revoking privileges from roles

```
-- Granting table-level privilege to a role
REVOKE priv_list ON table FROM role_list [AS revoker];

-- Granting Type-level privileges to a role
REVOKE priv_list ON TYPE udt FROM role_list [AS revoker];

-- Granting language-level privielges to a role
REVOKE priv_list ON LANGUAGE lang FROM role_list [AS revoker];

-- Granting routine-level privileges to a role
REVOKE priv_list ON proc_name FROM role_list [AS revoker];

-- Granting fragment-level privileges to a role
REVOKE FRAGMENT priv_list ON table (frag_list) FROM role_list [AS
revoker];
```

Example 4-57 shows revokes for the grants done in Example 4-57. As column-level privileges cannot be revoked, the revoke is done using REVOKE ALL.

Example 4-57 Examples for revoking privileges to roles

```
REVOKE SELECT ON emp_tab FROM "payroll_admin", "manager";
REVOKE ALL ON emp_tab FROM "payroll_admin", "manager", "employee";
```

Granting roles to users

Granting a role to a user is similar to granting a privilege to a user. The main difference is that in the case of granting a privilege to the user, it is granted on a SQL object such as a table, column, view, routine, language, or fragment. While granting a role, there is no SQL object associated with the grant. A role's existence is within a database, and the user assumes a role for the set of privileges that come along with it. These privileges can be on any of the SQL objects, as we discussed in the previous section.

A user with the DBA privilege grants roles to users. The CREATE ROLE and GRANT *role* statements do not activate the role. A role is activated by the SET ROLE statement, which is described in a subsequent section.

Example 4-58 gives the syntax of the GRANT ROLE statement. A role can be granted to another role or to a list of users. When the role is granted to users, it can be granted with the WITH GRANT OPTION clause or AS grantor clause.

Example 4-58 Syntax for granting roles to users

```
GRANT role_name TO user_list;
GRANT role_name TO role_name;
GRANT role_name TO user_list [ WITH GRANT OPTION ] [AS grantor] ;
```

Example 4-59 shows the GRANT statements for the sample scenario that grant roles to different users.

Example 4-59 Examples for granting roles to users

```
GRANT "manager"  TO "usr1";
GRANT "employee"  TO "usr1";
GRANT "employee" TO "usr2", "usr3";
GRANT "payroll_admin" TO "usr4";
```

The system catalog table SYSROLEAUTH contains the entries corresponding to the GRANT statements.

Example 4-60 shows the entries in SYSROLEAUTH for grants done in Example 4-59.

Example 4-60 Displaying roles granted to users

```
> select * from "informix".sysroleauth;


rolename      manager
grantee       usr1
is_grantable  n

rolename      employee
grantee       usr1
is_grantable  n

rolename      employee
grantee       usr2
is_grantable  n

rolename      employee
grantee       usr3
is_grantable  n
```

```
rolename      payroll_admin
grantee       usr4
is_grantable  n
```

Table 4-61 shows the REVOKE statement syntax.

Example 4-61 Syntax for revoking roles from users

```
REVOKE role_name FROM user_list;
REVOKE role_name FROM role_name;
REVOKE role_name FROM user_list [AS revoker] ;
```

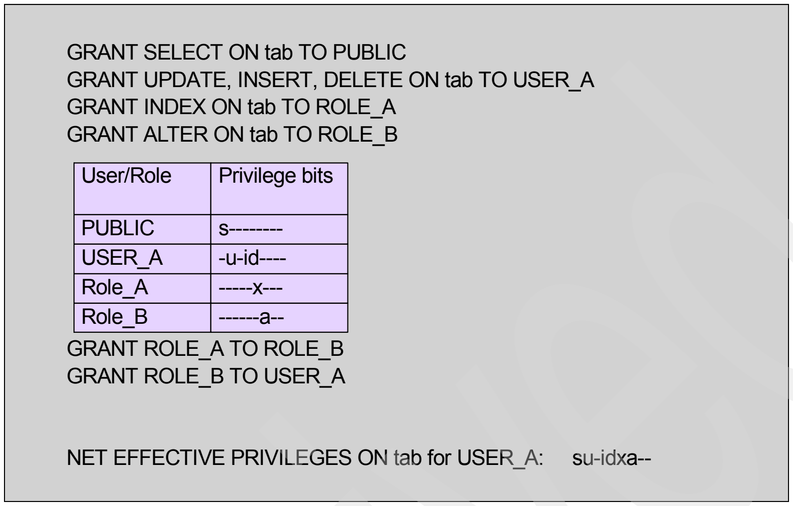
Example 4-62 shows the corresponding REVOKE statements for grants done in Example 4-59 on page 103.

Example 4-62 Example revoke statements

```
REVOKE "manager"  FROM "usr1";
REVOKE "employee"  FROM "usr1";
REVOKE "employee" FROM "usr2", "usr3";
REVOKE "payroll_admin" FROM "usr4";
```

When the REVOKE is done, the corresponding entries from SYSROLEAUTH system catalog table are deleted.

Figure 4-2 shows the privilege checking order when USER_A accesses table tab. USER_A who has been granted ROLE_B now assumes the privileges of ROLE_B, ROLE_A, and PUBLIC in addition to user's own privileges.


GRANT SELECT ON tab TO PUBLIC
GRANT UPDATE, INSERT, DELETE ON tab TO USER_A
GRANT INDEX ON tab TO ROLE_A
GRANT ALTER ON tab TO ROLE_B

User/Role	Privilege bits
PUBLIC	s--------
USER_A	-u-id----
Role_A	-----x---
Role_B	------a--

GRANT ROLE_A TO ROLE_B
GRANT ROLE_B TO USER_A

NET EFFECTIVE PRIVILEGES ON tab for USER_A: su-idxa--

Figure 4-2 Privilege checking order

Default role

An administrator can define a default role to be assigned to individual users or PUBLIC for a particular database. The default role is automatically applied when the user establishes the connection with the database.

Example 4-63 shows the syntax of the GRANT DEFAULT ROLE statement. A default role can be granted to a set of users.

Example 4-63 Syntax for granting default role

```
GRANT DEFAULT ROLE role_name TO user_list;
```

Example 4-64 shows the statements that perform GRANT DEFAULT as applied to our sample scenario. As shown, usr1 assumes the role manager when connected to the database. Similarly, usr4 assumes the role of payroll_admin when connected to the database.

Example 4-64 Examples for granting default roles

```
GRANT DEFAULT ROLE "manager" TO "usr1";
GRANT DEFAULT ROLE "payroll_admin" TO "usr4";
```

Example 4-65 shows the entries in the SYSUSERS system catalog table when default roles are granted to usr1 and usr4.

Example 4-65 Displaying default roles

```
username  usr1
usertype  C
priority  5
password
defrole   manager

username  usr4
usertype  C
priority  5
password
defrole   payroll_admin
```

Assuming a role

After a non-default role is granted to the user, the user must use the SET ROLE statement to enable it. The user assumes the default role as soon as a connection to the database is established. If no default role is granted to the user, then the user does not have any current role in the database when the connection is established. The user can change the current role or switch to a new role using the SET ROLE statement.

Example 4-66 shows the syntax of the SET ROLE statement. The role can be set to *role_name*. The role can also be set to a NULL value or NONE, in which case the current role is disabled. When the role is set to DEFAULT, the default role is enabled.

Example 4-66 Syntax for assuming a role

```
SET ROLE role_name
SET ROLE NULL
SET ROLE NONE
SET ROLE DEFAULT
```

When the user assumes a new role, the user acquires all the privileges of the role in addition to the privileges that the user already holds as an individual or as PUBLIC. If a role is granted to another role that has been assigned to the user, then the user acquires all the privileges of both roles, in addition to privileges the user already holds as an individual or as PUBLIC. A user can be granted several roles, but can assume no more than one non-default role, as specified by the SET ROLE, and can be enabled by the user at any given time. If the SET ROLE statement is repeated, then the new role replaces the old one as the current role.

The SET ROLE statement returns an error if the user does not currently hold (granted) a role and if the role being set is a built-in role.

The scope of the SET ROLE statement is until the session is terminated or when the database is closed or when the user executes another SET ROLE statement. As a role is not an entity by itself, only the user retains the ownership of database objects, such as tables, that are created during the session. When connecting the ANSI-compliant databases, SET ROLE must be the first statement. When explicit transactions are involved, the SET ROLE statement must be outside the transaction.

If the SET ROLE statement is executed as part of a trigger or stored procedure, and if the owner of the trigger or stored procedure is granted the role WITH GRANT OPTION, then the role is enabled even if you are not granted the role.

Privileges that the user has acquired by executing a procedure are not automatically relinquished after the SPL routine completes execution.

Built-in roles

The built-in roles such as EXTEND and DBSECADM are always in effect and do not require activation by the user using SET ROLE. The DBSECADM role is used in the context of label-based access control, which is described in 4.2, "Label-based access control" on page 115.

The EXTEND role is used to create and drop external routines written in C or Java, only when the IFX_EXTEND_ROLE configuration parameter is set to 1. The SET ROLE statement is not needed for the EXTEND role to take effect. It is sufficient for the user to hold the EXTEND without doing SET ROLE to enable it.

Example 4-67 shows how to grant the EXTEND role to the list of users.

Example 4-67 Syntax for granting EXTEND role

```
GRANT EXTEND TO user_list
```

Only a database system administrator (dbsa) or user *informix* can grant an EXTEND role to other users.

Example 4-68 shows the syntax for revoking the EXTEND role.

Example 4-68 Syntax for revoking EXTEND role

```
REVOKE EXTEND FROM user_list;
```

Dropping a role

Only a DBA or the user to whom the role is granted with the WITH GRANT OPTION can drop a role. After you drop a role, no user can grant or enable the dropped role. Also, users who are currently assigned the dropped role lose their privileges that came with the role, unless the same privileges were granted to public or to the user individually. If the dropped role is a default role, then the default role for the user becomes NULL.

Example 4-69 shows the syntax for the DROP ROLE statement. The name of the role is indicated by *role_name*.

Example 4-69 Syntax for dropping a role

```
DROP ROLE role_name
```

Example 4-70 shows the DROP ROLE statements for the sample scenario.

Example 4-70 Examples for dropping roles

```
DROP ROLE "payroll_admin";
DROP ROLE "manager";
DROP ROLE "employee";
```

The DROP ROLE statement cannot be used to drop a built-in role, such as EXTEND or DBSECADM.

Role-related operators

The operators related to the roles feature are DEFAULT_ROLE and CURRENT_ROLE. These operators are used anywhere in the SQL statements. The value of these operators depends on the current user and session context. If DEFAULT_ROLE is used in a query, the default role of the user who is executing the query is retrieved. If CURRENT_ROLE is used in a query, the role that is currently assigned to the user using the SET ROLE statement is shown.

Example 4-71 shows an example usage of CURRENT_ROLE and DEFAULT_ROLE operators. In our sample scenario, usr1 is granted the role of manager as a default role. As shown in the example, when usr1 connects to the database, the current role becomes the default role.

Example 4-71 Usage of CURRENT_ROLE and DEFAULT_ROLE

```
6660 >dbaccess emp_db -

Database selected.

> SELECT * FROM "informix".sysroleauth WHERE rolename=CURRENT_ROLE;
```

```
rolename      manager
grantee       usr1
is_grantable  n

1 row(s) retrieved.

> SELECT * FROM "informix".sysroleauth WHERE rolename=DEFAULT_ROLE;


rolename      manager
grantee       usr1
is_grantable  n

1 row(s) retrieved.
```

Example 4-72 is another example of the usage of operators where the current role of a user is changed upon connection. In the example, usr1 sets to employee role soon after connecting to the database. As shown, CURRENT_ROLE and DEFAULT_ROLE are different in this case.

Example 4-72 Demonstrates that CURRENT_ROLE and DEFAULT_ROLE are different

```
6661 >dbaccess emp_db -

Database selected.

> SET ROLE "employee";

Role set.

> SELECT * FROM "informix".sysroleauth WHERE rolename=CURRENT_ROLE;



rolename      employee
grantee       usr1
is_grantable  n

rolename      employee
grantee       usr2
is_grantable  n

rolename      employee
```

```
grantee       usr3
is_grantable  n

3 row(s) retrieved.


> SELECT DEFAULT_ROLE FROM "informix".systables WHERE tabid=1;


(expression)

manager

1 row(s) retrieved.

> SELECT * FROM "informix".sysroleauth WHERE rolename=DEFAULT_ROLE;


rolename      manager
grantee       usr1
is_grantable  n

1 row(s) retrieved.
```

4.1.16 Distributed context

Permissions for accessing objects in remote databases are controlled at the remote server. When a user executes a distributed query, privileges of the user on the local database objects are not applicable on the remote database. If a default role exists for the user at the remote database, that is used for access control. If a default role does not exist, then explicit privileges granted to the user and PUBLIC at the remote database are used in controlling access.

4.1.17 System tables

In this section, we provide the schema of different system catalog tables used to capture the privilege information of users on SQL objects. We have discussed all these system tables already with examples, while discussing privileges.

SYSUSERS

Table 4-8 lists the schema for the SYSUSERS system catalog table.

Table 4-8 SYSUSERS

Column	TYPE	Description
username	VARCHAR(32)	Name o of the database user or role
usertype	CHAR(1)	Code that specifies database-level privilege: C - Connect R - Resource D - DBA G - Role
priority	SMALLINT	Reserved for future use
password	CHAR(16)	Reserved for future use
defrole	VARCHAR(32)	Name of the default role

For examples of displaying values from this table, refer to 4.1.2, "Database privileges" on page 64.

SYSTABAUTH

The schema for SYSTABAUTH used to store table, view, synonym, and SEQUENCE level privileges is as given in Table 4-9.

Table 4-9 SYSTABAUTH

Column	TYPE	Description
grantor	VARCHAR(32)	Names of the user who granted the privilege
grantee	VARCHAR(32)	Name of the user to whom the privilege is granted
tabid	INTEGER	ID of the table from systables

Column	TYPE	Description

tabauth	CHAR(9)	Privilege bits that indicate which privileges are granted to the grantee:
		s = SELECT
		S = SELECT WITH GRANT OPTION
		u = UPDATE
		U = UPDATE WITH GRANT OPTION
		* = Column-level privilege
		- = Ungranted option
		i = INDEX
		I = INDEX WITH GRANT OPTION
		d = DELETE
		D = DELETE WITH GRANT OPTION
		x = INDEX
		X = INDEX WITH GRANT OPTION
		a = ALTER
		A = ALTER WITH GRANT OPTION
		r = REFERENCES
		R = REFERENCES WITH GRANT
		n = UNDER
		N = UNDER WITH GRANT OPTION

Example of table and VIEW level privileges are given in 4.1.5, “Table privileges” on page 73.

SYSCOLAUTH

Table 4-10 shows the schema of the table for storing column-level privileges.

Table 4-10 SYSCOLAUTH

Column	TYPE	Description
grantor	VARCHAR(32)	Name of the user who granted the privilege
grantee	VARCHAR(32)	User to whom the privilege is granted
tabid	INTEGER	ID of the table from SYSTABLES
colno	SMALLINT	Column number within the table

Column	TYPE	Description

colauth	CHAR(3)	Bit pattern for column-level privileges:
		s = SELECT
		S= SELECT WITH GRANT OPTION
		u = UPDATE
		U=UPDATE WITH GRANT OPTION
		r = REFERENCES
		R = REFERENCES WITH GRANT OPTION

Examples for viewing column-level privileges are provided in 4.1.6, “Column privileges” on page 79.

SYSFRAGAUTH

Table 4-11 shows the schema of the table for storing fragment-level privileges.

Table 4-11 SYSFRAGAUTH

Column	TYPE	Description
grantor	VARCHAR(32)	User who granted the privilege.
grantee	VARCHAR(32)	User to whom privilege is granted.
tabid	INTEGER	ID of the table from SYSTABLES.
fragment	VARCHAR(128)	Name of the dbspace where the fragment resides.

fragauth	CHAR(6)	A 6-byte pattern showing fragment privileges. Three of them are reserved for future use:
		u = UPDATE
		U = UPDATE WITH GRANT OPTION
		i = INSERT
		I = INSERT WITH GRANT OPTION
		d = DELETE
		D = DELETE WITH GRANT OPTION

Examples of displaying fragment-level privileges are given in 4.1.7, “Fragment privileges” on page 82.

SYSXTDTYPEAUTH

Table 4-12 shows the schema of the table for showing TYPE-level privileges.

Table 4-12 SYSXTDTYPEAUTh

Column	TYPE	Description
grantor	VARCHAR(32)	User who granted the privilege
grantee	VARCHAR(32)	User to whom privilege is granted
type	INTEGER	Code identifying the UDT

auth	CHAR(2)	Privilege bits on UDT:
		n = UNDER
		N = UNDER WITH GRANT OPTION
		u = USAGE
		U = USAGE WITH GRANT OPTION

Examples of displaying TYPE-level privileges are given in 4.1.8, “TYPE privileges” on page 84.

SYSPROCAUTH

Table 4-13 shows the schema of the table for storing routine-level privileges.

Table 4-13 SYSPROCAUTH

Column	TYPE	Description
grantor	VARCHAR(32)	User who granted the privilege
grantee	VARCHAR(32)	User to whom privilege is granted
procid	INTEGER	Unique identifying code of the routine

procauth	CHAR(1)	TYPE of the privilege granted on the routine:
		e = EXECUTE
		E = EXECUTE WITH GRANT OPTION

Examples of displaying routine-level privileges are given in 4.1.11, “Routine privileges” on page 89.

SYSLANGAUTH

Table 4-14 shows the schema of table for storing language-level privileges.

Table 4-14 SYSLANGAUTH

Column	Type	Description
grantor	VARCHAR(32)	User who granted the privilege
grantee	VARCHAR(32)	User to whom privilege is granted
langid	INTEGER	Identifying code of language in SYSROUTINELANGS table

langauth	CHAR(1)	Language authorization:
		u = USAGE
		U = USAGE WITH GRANT OPTION

Examples of displaying language-level privileges are given in 4.1.12, “Language privileges” on page 92.

4.2 Label-based access control

While discretionary access control works at the level of database objects such as databases, tables, columns, views, and so on, label-based access control works at the level of data rows and columns. Tables are protected by security policies, and individual rows and columns are protected by security labels. The database security administrator (DBSECADM) creates security policies and labels and grants labels to individual users. When individual users with security labels write data rows to a table protected by an equivalent security policy, data labels are generated in individual rows of the table.The cardinal principle by which LBAC works is that users whose security labels dominate the data labels protecting the rows are able to read data. How is the dominance calculated? Any label is composed of individual components and elements within each component. A user label dominates a row label if individual components in the user label dominate individual components of a row label, for a given security policy. These concepts are best described with sample scenarios.

4.2.1 Sample scenarios

Consider a company LBL Corp., which has a traditional organizational structure. Figure 4-3 shows the partial organizational structure of the company. This figure shows all departments of the company headed by the respective VPs.

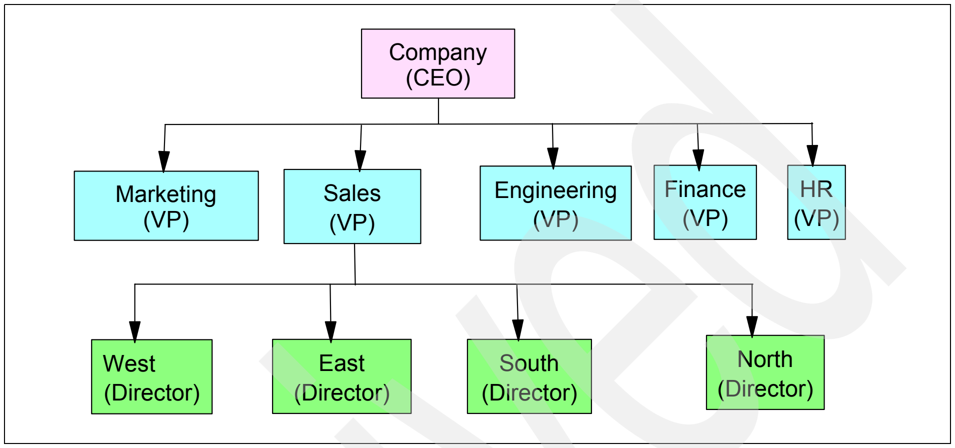


Figure 4-3 Sample scenario showing organizational structure in a company

Using LBL Corp. as background, here we describe three business requirements. We then discuss how to use LBAC to fulfill these business requirements.

- Scenario 1

  The sales department is further divided into regions headed by the respective regional directors. The hierarchy extends further into sub-regions and finally into individual sales-persons. Figure 4-4 shows an expanded view of the sales organization.

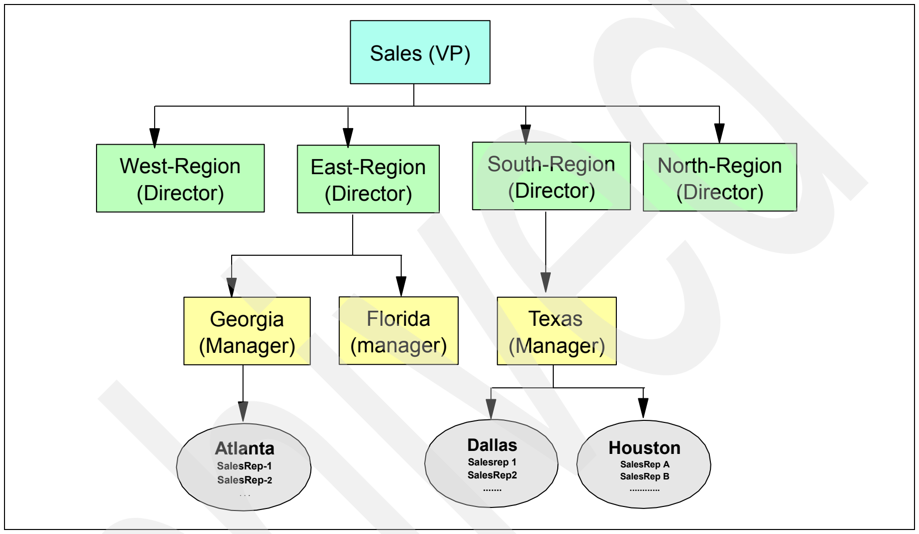


Figure 4-4 Sales organization

The prime requirement of storing sales data is that people at any level be able to see the data belonging to them and their subordinates while not seeing data that belongs to their peers. For instance, the sales vice president should be able to see sales data belonging to all the regions, whereas each regional director only sees data belonging to their region. Using a similar approach down the hierarchy, an individual sales person can only see his own data.

This can be accomplished by granting labels to users in the organization such that labels granted to people higher up in the hierarchy dominate labels granted to people who report to them. Thus, the label granted to the sales VP dominates the labels granted to any of the regional directors. The labels granted to regional directors dominate the labels granted to any of the sub-region managers that report to them. Labels that are granted to individual sales people are dominated by labels granted to their managers. This is best accomplished by defining a TREE component for the sales organization. We further discuss this scenario as an example while defining the security policies and labels for the sales data table.

- Scenario 2

  Another scenario in which LBAC is useful is where the documents are classified based on their level of sensitivity and a user with a higher sensitivity level can read documents belonging to lower sensitivity levels. Once again, consider Figure 4-3 on page 116, where marketing documents can be classified into sensitivity levels such as Top Secret, Secret, Confidential, and Unclassified. The CEO of the company could be granted a label that has Top Secret as its component element. A technical architect could be given a higher sensitivity level so that competitive information is accessible. This is best accomplished by defining the ARRAY component of sensitivity levels. We discuss this scenario further while describing examples of creating security components.

- Scenario 3

  Now we describe a third scenario, which describes the usage of the SET component type. Consider the HR department in Figure 4-3 on page 116. The VPs of individual departments can see the employee records belonging to their own departments. They should not be able to see records of employees belonging to other departments. But the human resources VP should be able to see all employee records of the company. Thus, the HR VP has a label that includes all departments. Other VPs have labels that have only their departments as part of the SET component. We demonstrate this with examples in subsequent sections.

The discussion of the sample scenarios demonstrates that LBAC provides the framework to create security policies in a variety of ways and protect data with these policies. The chief characteristic of an LBAC-protected table is that security and privacy is achieved by making data access more granular. The SQL objects that enable LBAC enforcement are security label components, security policies, and security labels. These objects are used in protecting individual rows and columns in tables. LBAC enforcement is done at a lower level than DAC enforcement. After the privileges are checked at the database level and table level, LBAC enforcement is done at row and column levels to fetch individual rows.

Only a DBSECADM can create LBAC SQL objects or apply them to tables or grant them to users. The DBSECADM is a built-in role provided by IDS, and this role is granted to individual users by the database system administrator. All LBAC SQL objects also exist in the context of a database.

4.2.2 Security label components

A security label component is a primary building block of any LBAC object. A component consists of multiple elements. It is the elements in the component that determine the dominance of a label relative to another. There are three types of security label components in LBAC. These are ARRAY, SET, and TREE component types.

ARRAY security component type

An ARRAY component type consists of an ordered set of elements, where an element occurring earlier in the ARRAY has a higher weight than the element occurring later.

Example 4-73 shows the syntax for creating an ARRAY security label component type. The component name that is created is represented by *compname*, and the elements in the component are represented by *element_list*. There can be a maximum of 64 elements in the element list.

Example 4-73 Syntax for creating ARRAY security component

```
CREATE SECURITY LABEL COMPONENT compname ARRAY [ element_list ];
```

Note: Notice the usage of brackets ([ ])to enclose the element_list while creating an ARRAY component type.

Example 4-74 shows an example for creating an ARRAY component. The component *classification* is created with elements, as shown in the example. The element Top Secret has a higher weight than Secret, and Secret has a higher weight than Confidential, and Confidential has a higher weight than Unclassified. Thus, an element occurring earlier in an ARRAY component has a higher weight than elements occurring later. A user with Top Secret classification can read data at any classification level, provided that other components of the user's label dominate corresponding components of the data label. Similarly, a user with Secret classification can only read data classified at Secret, Confidential, or Unclassified.

Example 4-74 Creating an ARRAY security component

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

Thus, the ARRAY component helps separate users into different classification levels.

Example 4-75 shows that employees are divided into broad categories based on hierarchy in an organization. In general, the employees higher up in the organizational ladder are privy to more sensitive information. Thus, using the ARRAY component type is suitable for this purpose.

Example 4-75 Creating another component of type ARRAY

```
CREATE SECURITY LABEL COMPONENT org_position ARRAY ['CEO', 'VP',
'Director', 'Manager','Staff'];
```

SET security component type

A SET component type constitutes an unordered set of elements. Unlike elements in an ARRAY component type, the elements in the SET component type do not have any weight relative to another.

Example 4-76 shows the syntax for creating the SET component type. The name of the component is represented by *compname* and the element list is represented by *element_list*. There can be a maximum of 64 elements in a SET component type.

Example 4-76 Syntax for creating SET component

```
CREATE SECURITY LABEL COMPONENT compname SET { element_list };
```

Note: Notice the usage of curly brackets ({ }) to enclose the element list while creating a SET component type.

Example 4-77 shows an example of creating the SET security component type. A SET component *department* is created with elements as individual departments such as marketing, sales, engineering, finance, and HR. Each of these departments has its own weight, which is not higher or lower than the weight of any other department. A user with a label containing marketing and sales elements is able to read data belong to the marketing department or the sales department or both. But a user with a label containing marketing is not able to read data that is labelled marketing and sales.

Example 4-77 Creating a SET component

```
CREATE SECURITY LABEL COMPONENT department SET {'Marketing', 'Sales',
'Engineering', 'Finance', 'HR'};
```

TREE security component type

A TREE component type consists of elements organized in a tree structure. There can be a maximum of 64 elements in a TREE component type. An element can have any number of children, but only one parent.

Example 4-78 shows the syntax for creating the TREE component. The component name is represented by *compname* and each element in the element list is UNDER some other element, thus forming a tree structure. The first element is always the ROOT element.

Example 4-78 Syntax for creating a TREE component

```
CREATE SECURITY LABEL COMPONENT compname TREE ( 'element' ROOT,
              element_list);
```

Example 4-79 provides an example of creating the TREE security component type. As shown, sales regions are organized into a hierarchical structure with HeadQuarters as ROOT. Under HeadQuarters, there are four regions: east, west, north, and south. Under each region there are several states. We only show a partial tree under the east region. This component type can be used to protect sales data. Someone with a label containing HeadQuarters as the element can read data labelled at any of the regions below HeadQuarters. In general, if a user label contains any of the elements of the data label or the ancestor of one such element, the user is allowed to read. This component in combination with the org_position component in Example 4-75 on page 120 can provide labels to users that help in viewing sales data based on organizational position and region.

Example 4-79 Creating a TREE component

```
CREATE SECURITY LABEL COMPONENT region TREE ( 'HeadQuarters' ROOT,
                'East' UNDER 'HeadQuarters',
                'West' UNDER 'HeadQuarters',
                'North' UNDER 'HeadQuarters',
                'South' UNDER 'HeadQuarters',
                'Georgia' UNDER 'East',
                'Florida' UNDER 'East',
                'Atlanta' UNDER 'Georgia',
                'Texas' UNDER 'South',
                'Dallas' UNDER 'Texas',
                'Houston' UNDER 'Texas'
);
```

The elements of a TREE are defined in a strict order. You can use an element as a parent element only if it has been defined before as the ROOT element or as a child of some other element.

4.2.3 Security policy

A security policy is defined using the security components that are created. Based on the type of data in a table, a suitable security policy is defined that protects and provides granular data access to users. There can be a maximum of 16 components defined in a security policy.

Example 4-80 shows the syntax for creating the security policy. The policy name is represented by *plcyname*, and the individual components are represented by *component_list*.

Example 4-80 Syntax for creating a security policy

```
CREATE SECURITY POLICY plcyname COMPONENTS component_list;
```

Example 4-81 provides an example of creating a security policy. As shown, sales_plcy is the name of the policy, and this security policy is made up of components org_position and region.

Example 4-81 Creating a security policy

```
CREATE SECURITY POLICY sales_plcy COMPONENTS org_position, region;
```

The security policy is used to protect the table. The rows and columns of a table are protected by security labels, which are derived from the table's security policy. Depending on data in the table, different security policies protect different tables. In most practical cases, companies may be using two or three security policies to secure tables in their database.

Example 4-82 demonstrates that a single policy can be defined to protect all tables. Individual security labels created can be tailor-made to secure data specific to tables. As a policy can potentially contain a maximum of 16 components, a component type can occur multiple times.

Example 4-82 Single policy for all tables

```
CREATE SECURITY POLICY company_policy COMPONENTS org_position,
                                                region,
                                                department,
                                                classification ;
```

4.2.4 Security label

A security label is derived from a security policy. The DBSECADM creates security labels and assigns them to individual users. As a table is protected by a single security policy, only users with labels belonging to the table's security policy can access and modify the table. Other users, with no labels assigned, or with labels that belong to a different security policy, are not able to access the data in the table.

A security label contains all the components in a security policy. A restriction when creating security labels is that there can be only one element specified as part of an ARRAY component.

Example 4-83 shows the example syntax for creating a security label. As shown, a security label sales_vp is created in the policy sales_plcy. The label is assigned the VP element from the component org_position and the HeadQuarter element from the component region. This label is suitable to be granted to a sales VP.

Example 4-83 Creating a security label

```
CREATE SECURITY LABEL sales_plcy.sales_vp
                     COMPONENT org_position 'VP',
                     COMPONENT region  'HeadQuarters';
```

Example 4-84 shows another example of creating a security label. Here a label called sales_rep is created in the policy sales_plcy. This label is defined using elements such as staff and Atlanta. This label is granted to sales representatives in the Atlanta region. In subsequent sections, we discuss how data rows protected by this label can be read by sales_vp, whose label is created as shown in Example 4-83.

Example 4-84 Creating another security label

```
CREATE SECURITY LABEL sales_plcy.sales_rep
                     COMPONENT org_position 'staff',
                     COMPONENT region  'Atlanta';
```

A very significant factor in creating a label is its permanence. The DBSECADM has to carefully choose labels so that they are permanent over an extended period of time. As labels get attached to data, changing any of the components or policies or labels leads to dropping entire tables and reloading data with newly created labels. Needless to say, this is a very tricky and undesirable exercise for any administrator. For this reason, LBAC objects have to be generic. For instance, if you attach the name of a user as an element of a label assigned to the user, this label is meaningless as soon as the user leaves the company. The DBSECADM needs to make reasonable assumptions about the permanence of generic element names such as the name of a city or the name of a region such as west, east, north, or south.

4.2.5 Label comparison

A label dominates another label if component elements in one label dominate the corresponding component elements in another label. How a component element in one label dominates the component element in another label depends on the component type. If all the components in label A dominate corresponding components in label B, then label A is said to dominate label B.

In the case of the ARRAY component type, the comparison depends on where the elements in the two labels are positioned within the array. If the ARRAY component element in label A is positioned earlier than the corresponding component element in the label B, then label A dominates with respect to the component that is being compared.

In the case of the SET component type, the comparison depends on whether the elements in one label are a super-set of elements in the other label. If the elements contained in label A's SET component are a super-set of the elements contained in the corresponding component in label B, then label A dominates over label B with respect to the component that is being compared.

In the case of the TREE component type, the comparison depends on whether the element in one label is one of the elements in the second label or the ancestor of one such element. If each of the TREE component elements in label A is one of the elements in the corresponding component in label B, or an ancestor of one such element in label B, label A dominates over label B with respect to the component that is being compared.

In totality, label A is said to dominate label B if each of the components in label A dominates the corresponding components in label B. We illustrate the label comparison method using some examples.

Example 4-85 on page 125 shows the CREATE SECURITY LABEL statements for labels sales_vp and sales_rep. The component org_position of type ARRAY was defined in 4.2.2, "Security label components" on page 118. In the component org_position, the element VP occurs earlier than the element staff. Thus, the component org_position in label sales_vp dominates the corresponding component in label sales_rep. Similarly, the component region is of type TREE. The element HeadQuarters is an ancestor of the element Atlanta. Thus, the component region in label sales_vp dominates the corresponding component in label sales_rep. As both the components in sales_vp label dominate corresponding components in sales_rep label, the label sales_vp dominates the label sales_rep.

Example 4-85 Demonstrates one label dominating the other label

```
CREATE SECURITY LABEL sales_plcy.sales_vp
                        COMPONENT org_position 'VP',
                        COMPONENT region  'HeadQuarters';

CREATE SECURITY LABEL sales_plcy.sales_rep
                        COMPONENT org_position 'staff',
                        COMPONENT region  'Atlanta';
```

Example 4-86 shows that a security policy hr_plcy is created with components *org_position* and *department*. Also, two labels are created within the security policy called mktg_vp and finance_director. The component org_position in label mktg_vp dominates the corresponding component in label finance_director, because the element VP occurs before the element director in the ARRAY component. The component department is of type SET. The element marketing in label mktg_vp is not present in the corresponding component in label finance_director. Thus, the component department in label mktg_vp does not dominate the corresponding component in label finance_director. As some components in label mktg_vp do not dominate corresponding components in label finance_director, the label mktg_vp does not dominate the label finance_director.

Example 4-86 Demonstrates that labels do not dominate each other

```
CREATE SECURITY POLICY hr_plcy COMPONENTS org_position, department;

CREATE SECURITY LABEL hr_plcy.mktg_vp
                        COMPONENT org_position 'VP',
                        COMPONENT department  'Marketing';

CREATE SECURITY LABEL hr_plcy.finance_director
                        COMPONENT org_position 'Director',
                        COMPONENT department  'Finance';
```

4.2.6 Grant and Revoke

The DBSECADM grants security labels to users. The labels are granted for read access or for write access or both. The user's WRITE label is used as a data label during the INSERT operation. The user's READ label is used when data is accessed with SELECT queries. For a single policy, a user can hold at most one label. If the READ and WRITE labels are different, then the user can hold at most one READ label and one WRITE label in a single policy. Also, the READ and WRITE user labels are related as follows:

- The READ label must dominate the WRITE label.
- The element specified in the ARRAY component types in both READ and WRITE labels should match.
- Also, a user cannot grant a label to itself.

Example 4-87 shows the syntax of the GRANT statement.

Example 4-87 Syntax for granting security label

```
GRANT SECURITY LABEL plcyname.labelname TO username [FOR [WRITE | READ
| ALL ] ACCESS ];
```

Example 4-88 shows some of the example GRANT statements. You can grant a security label for READ access, WRITE access, or ALL access.

Example 4-88 Some sample grant statements for granting security labels

```
GRANT SECURITY LABEL sales_plcy.sales_vp TO "usr1";
GRANT SECURITY LABEL sales_plcy.sales_vp TO "usr2";
GRANT SECURITY LABEL sales_plcy.sales_rep TO "usr3" FOR WRITE ACCESS;
GRANT SECURITY LABEL sales_plcy.sales_rep_mgr TO "usr3" FOR READ
ACCESS;
```

Example 4-89 shows the syntax of the REVOKE statement.

Example 4-89 Syntax for revoking security label

```
REVOKE SECURITY LABEL plcyname.labelname FROM username [FOR [WRITE |
READ | ALL ] ACCESS ];
```

Example 4-90 shows the REVOKE statements for the GRANTs done in Example 4-88.

Example 4-90 Some sample REVOKE SECURITY LABEL statements

```
REVOKE SECURITY LABEL sales_plcy.sales_vp FROM "usr1";
REVOKE SECURITY LABEL sales_plcy.sales_vp FROM "usr2";
REVOKE SECURITY LABEL sales_plcy.sales_rep FROM "usr3" FOR WRITE
ACCESS;
REVOKE SECURITY LABEL sales_plcy.sales_rep_mgr FROM "usr3" FOR READ
ACCESS;
```

4.2.7 Protecting tables

Individual tables are protected by means of security policies. An unprotected table is one that does not have any security policy attached. Users with no security labels granted to them can only access unprotected tables. In order to access protected tables, users should have READ and WRITE security labels that belong to the protected table's security policy. User labels get attached to rows when a user performs DML operations. Similarly, table columns can be associated with labels that restrict access to protected columns. Thus, data in the table is protected at both the granularity of the row and column level.

Example 4-91 shows the CREATE TABLE statement used to create a table with a security policy sales_plcy.

Example 4-91 Creating a table with row protection

```
CREATE TABLE sales_data (
        ulabel          IDSSECURITYLABEL,
        id              INTEGER NOT NULL,
        name            CHAR(32) NOT NULL,
        product         CHAR(128) NOT NULL,
        sale_total      FLOAT NOT NULL,
        date_sold       DATE NOT NULL
        ) SECURITY POLICY sales_plcy;
```

Row protection

A special column type IDSSECURITYLABEL is used to store security labels. A protected tables needs to have this extra column to enable row protection. Example 4-91 shows the special column *ulabel* of type IDSSECURITYLABEL used to store data labels. All rows in a row-protected table have valid labels in the IDSSECURITYLABEL column. You cannot have NULL values stored in an IDSSECURITYLABEL column.

Table 4-15 shows some of the sample rows in the sales_data table. For brevity, some of the columns are dropped in the display. The ulabel column shows the encoded label. As both usr1 and usr1 are granted the same label, sales_vp, the value in the ulabel column is the same for transactions belonging to usr1 and usr2. The user usr3 is granted the label sales_rep and the label value is different.

Table 4-15 Sample rows of sales_data table

ulabel	Name	Product	Sales	date_sold
AAAAAAAAAAAAAAA8000000000000001	usr1	prod-A	154.00	10/25/2007
AAAAAAAAAAAAAAA8000000000000001	usr1	prod-B	162.00	10/24/2007

ulabel	Name	Product	Sales	date_sold
AAAAAAAAAAAAAAA80000000000000001	usr2	prod-A	169.00	10/24/2007
AAAAAAAAAAAAAAA80000000000000001	usr2	prod-C	192.00	10/25/2007
AAAAAAAAAAAAAAA80000000000000001	usr2	prod-A	155.00	10/24/2007
2AAAAAAAAAAAAAAA0000000000000400	usr3	prod-C	144.00	10/24/2007
2AAAAAAAAAAAAAAA0000000000000400	usr3	prod-B	205.00	10/24/2007
2AAAAAAAAAAAAAAA0000000000000400	usr3	prod-D	187.00	10/24/2007

The labels granted to usr1 and usr2 are the same. If the usr1 or usr2 does a SELECT * from this table, they see all rows with label value AAAAAAAAAAAAAAA80000000000000001 and 2AAAAAAAAAAAAAAA0000000000000400. The labels granted to user1 and user 2 dominate the label granted to usr3. If usr3 does a SELECT * on this table, the user only sees the rows with label value 2AAAAAAAAAAAAAAA0000000000000400. For a protected table, the IDSSECURITYLABEL column should not be NULL or empty.

Column protection

One or more columns in table can be protected by one or mode labels, by specifying the COLUMN SECURED WITH clause in the CREATE TABLE or ALTER TABLE statements.

Example 4-92 shows the creation of additional labels for the HR department. The labels hr_vp, hr_rep_mktg, and hr_rep are created. These labels are granted to usr4, usr5, and usr6, respectively.

Example 4-92 More examples of creating and granting security labels

```
CREATE SECURITY LABEL hr_plcy.hr_vp COMPONENT org_position 'VP',
COMPONENT department  'Marketing', 'Sales', 'Engineering', 'Finance',
'HR';

CREATE SECURITY LABEL hr_plcy.hr_rep_mktg
                        COMPONENT org_position 'Staff',
                        COMPONENT department  'Marketing', 'HR';

CREATE SECURITY LABEL hr_plcy.hr_rep
                        COMPONENT org_position 'Staff',
                        COMPONENT department  'HR';
```

```
GRANT SECURITY LABEL hr_plcy.hr_vp TO "usr4";
GRANT SECURITY LABEL hr_plcy.hr_rep_mktg TO "usr5";
GRANT SECURITY LABEL hr_plcy.hr_rep TO "usr6";
```

Example 4-93 shows the schema of the hr_data table with the salary column protected. A user with a READ label that dominates the label hr_rep is able to read from this table. A user with the WRITE label set to hr_rep is able to write to this table. The label that protects the column is stored in the SYSCOLUMNS system catalog table.

Example 4-93 Create table with column protection

```
CREATE TABLE hr_data (
        name            CHAR(32),
        title           CHAR(32),
        salary          FLOAT COLUMN SECURED WITH hr_rep
        ) SECURITY POLICY hr_plcy;
```

Example 4-94 demonstrates that if a user whose READ label does not dominate hr_rep tries to read from the hr_data table, an error message is shown. This command is executed as usr3.

Example 4-94 Error - unauthorized user reads column protected table

```
7305 >dbaccess db -

Database selected.

> select * from hr_data;

 8245: User cannot perform READ access to the protected column
(salary).
Error in line 1
Near character position 20
>
```

Protecting rows and columns

Example 4-95 extends the hr_data table by adding the column of type IDSSECURITYCOLUMN so that rows are also protected.

Example 4-95 Create table with both row and column protection

```
CREATE TABLE hr_data (
        dlabel          IDSSECURITYLABEL,
        name            CHAR(32),
        title           CHAR(32),
        salary          FLOAT COLUMN SECURED WITH hr_rep
        ) SECURITY POLICY hr_plcy;
```

ALTER TABLE

The ALTER TABLE statement is used to add a security policy to an unprotected table, add row or column protection or both to a table, drop row or column protection or both, drop a security policy from a protected table, and so on.

Example 4-96 shows that the hr_data table is initially created as an unprotected table and later altered to add security policy and row and column protection. Note the usage of the default clause while adding the IDSSECURITYLABEL column. If the unprotected table already has rows in it, then the label specified by the default clause is applied to existing rows.

Example 4-96 Alter table to add both row and column protection

```
CREATE TABLE hr_data (
        name            CHAR(32),
        title           CHAR(32),
        salary          FLOAT
        );

ALTER TABLE hr_data
        ADD SECURITY POLICY hr_plcy,
        ADD     dlabel  IDSSECURITYLABEL default 'hr_rep',
        MODIFY  (salary FLOAT COLUMN SECURED WITH hr_rep);
```

Example 4-97 shows a few of the ALTER TABLE statements to drop the security policy, and row and column security. Effectively, row protection is dropped when the IDSSECURITYLABEL column is dropped.

Example 4-97 Drop security policy

```
ALTER TABLE hr_data DROP dlabel;
ALTER TABLE hr_data MODIFY (salary FLOAT DROP COLUMN SECURITY );
ALTER TABLE hr_data DROP SECURITY POLIC
```

4.2.8 LBAC access rules

There are certain inherent rules LBAC follows while accessing data in protected rows and columns. These rules are conceptually captured in 4.2.5, "Label comparison" on page 124. These rules are formally defined as IDSLBACRULES. The rules are different for each of the component types and whether the access is READ or WRITE. The rules are:

- IDSLBACREADARRAY

  Each array component of the user security label must be greater than or equal to the array component of the data row security label. That is, only data at or below the level of the user can be read.

- IDSLBACWRITEARRAY

  Each array component of the user security label must be equal to the array component of the data row security label. That is, only data at the same level as the user can be written.

- IDSLBACREADSET

  Each set component of the user security label must include the set component of the data row security label.

- IDSLBACWRITESET

  Each set component of the user security label must include the set component of the data row security label.

- IDSLBACREADTREE

  Each tree component of the user security label must include at least one of the elements in the tree component of the data row security label (or the ancestor of one such element).

- IDSLBACWRITETREE

  Each tree component of the user security label must include at least one of the elements in the tree component of the data row security label (or the ancestor of one such element).

The READ access rules such as IDSLBACREADARRAY, IDSLBACREADSET, and IDSLBACREADTREE are used when reading rows and columns from protected tables using SELECT, UPDATE, or DELETE operations.

The WRITE access rules such as IDSLBACWRITEARRAY, IDSLBACWRITESET, and IDSLBACWRITETREE are used when writing data to protected tables using INSERT, UPDATE, or DELETE operations.

4.2.9 Exemptions

User access to secured tables is dictated by rules that are defined as IDSLBACRULES. However, users might be granted exemptions to circumvent one or more of these rules. For example, if an exemption is granted to a user on IDSLBACREADARRAY, then this rule is not applied while comparing a user label with a data label. If an exemption is granted on ALL rules then LBAC checks are bypassed completely for the user. The exemptions that are granted can be revoked at a later time.

Example 4-98 provides the syntax for granting and revoking exemptions.

Example 4-98 Syntax for granting exemptions

```
GRANT EXEMPTION ON RULE rule FOR plcy TO [USER] user

REVOKE EXEMPTION ON RULE rule FOR plcy FROM [USER] user
```

Example 4-99 shows that the finance department VP (usr7) has been granted exemptions to access sales data.

Example 4-99 Example GRANT EXEMPTION statements

```
CREATE SECURITY POLICY fin_plcy COMPONENTS org_position, department;

CREATE SECURITY LABEL fin_plcy.fin_vp
                        COMPONENT org_position 'VP',
                        COMPONENT department  'Finance';

GRANT SECURITY LABEL fin_plcy.fin_vp TO "usr7";

GRANT EXEMPTION ON RULE IDSLBACREADARRAY FOR sales_plcy TO "usr7";
GRANT EXEMPTION ON RULE IDSLBACREADTREE FOR sales_plcy TO "usr7";
```

Example 4-100 shows that usr7 has been granted exemptions on all rules on sales_plcy.

Example 4-100 Granting exemption on ALL rules

```
GRANT EXEMPTION ON RULE ALL FOR sales_plcy TO "usr7";
```

Note that exemptions are granted on an exceptional basis so that users perform privileged operations. They need to be revoked as soon as the user completes the operation. Exemptions are a powerful mechanism to circumvent LBAC access rules and so they need to be used carefully.

4.2.10 Maintaining LBAC objects

We discussed the flow for enforcing LBAC security by creating security label components, security policies, and security labels. There are SQL statements available that help maintain the components, policies, and labels that are created. We briefly discuss the DROP, RENAME, and ALTER statements.

DROP statements

The DROP statements help in dropping the unused security label components, policies, or labels.

Example 4-101 gives the syntax for dropping a security label component. The component cannot be dropped if there exists a policy that depends on the component.

Example 4-101 Dropping a security label component

```
DROP SECURITY LABEL COMPONENT compname
```

Example 4-102 provides the syntax for dropping a security policy. If the RESTRICT option is specified, then the policy is not dropped if there are dependent labels defined. If the CASCADE option is specified, then dropping the policy effectively drops all the labels that are defined on the policy.

Example 4-102 Syntax for dropping a security policy

```
DROP SECURITY POLICY plcyname [ RESTRICT | CASCADE ];
```

Example 4-103 provides the syntax for dropping a security label. The security label is not dropped if it is protecting a column or if it has already been granted to a user.

Example 4-103 Syntax for dropping a security label

```
DROP SECURITY LABEL plcyname.labelname
```

Example 4-104 provides the actual DROP statements for the LBAC objects we created in the book.

Example 4-104 Dropping LBAC objects

```
DROP SECURITY LABEL COMPONENT classification;
DROP SECURITY POLICY sales_plcy;
DROP SECURITY LABEL sales_plcy.sales_rep;
```

RENAME statements

The RENAME statements have the net effect of dropping and creating the object.

Example 4-105 shows the syntax for renaming a security label component. The security component oldcompname is renamed to newcompname. As the security label component is renamed, the policies and labels that depend on it are not impacted, as the relationship between them is through identifiers.

Example 4-105 Syntax for renaming a security label component

```
RENAME SECURITY LABEL COMPONENT oldcompname TO newcompname
```

Example 4-106 shows the syntax for renaming the security policy. The policy oldplcynameis renamed to newplcyname.

Example 4-106 Syntax for renaming security policy

```
RENAME SECURITY POLICY oldplcyname TO newplcyname
```

Example 4-107 shows the syntax for renaming a security label. The security label oldlabelname is renamed to newlabelname.

Example 4-107 Syntax for renaming a security label

```
RENAME SECURITY LABEL oldlabelname TO newlabelname
```

Example 4-108 shows the actual rename statements for LBAC objects created in this book.

Example 4-108   Renaming LBAC objects

```
RENAME SECURITY LABEL COMPONENT classsification TO newclassification;
RENAME SECURITY LABEL sales_plcy.sales_rep TO sales_associate;
RENAME SECURITY POLICY sales_plcy TO sales_policy;
```

ALTER statements

The only LBAC object on which the ALTER statement is allowed is the security label component.

Example 4-109 provides the syntax of the ALTER statement for different component types. You can add a list of elements either BEFORE or AFTER a particular element in the case of the ARRAY security label component. In the case of the SET component type, the element list is added and no location information is needed. In the case of the TREE component type, the element list is added UNDER a particular element. Example 4-109 shows the syntax of an ALTER statement under different scenarios. In the example, elem_list refers to the comma-separated list of elements.

Example 4-109   Syntax of the ALTER statement for different component types

```
ALTER SECURITY LABEL COMPONENT compname ADD ARRAY [elem_list BEFORE
element]
ALTER SECURITY LABEL COMPONENT compname ADD ARRAY [elem_list AFTER
element
ALTER SECURITY LABEL COMPONENT compname ADD SET {elem_list}
ALTER SECURITY LABEL COMPONENT compname ADD TREE (elem_list UNDER
element)
```

Example 4-110 shows a few sample ALTER statements for LBAC objects created.

Example 4-110   Altering LBAC objects

```
ALTER SECURITY LABEL COMPONENT classification ADD ARRAY [ 'Classified',
'Restricted' BEFORE 'Confidential' ];
ALTER SECURITY LABEL COMPONENT classification ADD ARRAY [ 'Public'
AFTER 'Unclassified' ];
ALTER SECURITY LABEL COMPONENT department ADD SET { 'IT', 'Professional
services' };
ALTER SECURITY LABEL COMPONENT region ADD TREE ( 'California', 'Oregon'
UNDER 'West' );
```

4.2.11 Built-in functions

IDS provides certain built-in functions, which are used for converting labels from one form to another. These labels are used in queries and DML statements. Using these functions, users are able to insert data labels in tables that are not equivalent to their WRITE labels. While these functions allow specific data labels to be written to tables, user access to these tables is still controlled by IDSLBACRULES. The built-in functions provided by IDS are:

- SECLABEL_BY_COMP
- SECLABEL_BY_NAME
- SECLABEL_TO_CHAR

SECLABEL_BY_COMP

This function takes the specified component elements as input and returns the label encoded value to be inserted into the IDSSECURITYLABEL column of a table. Elements of each component belonging to the security policy are separated by a colon (:). If more than one element in a component needs to be specified, they are enclosed within parenthesis.

Example 4-111 shows the syntax for using the SECLABEL_BY_COMP function. The policy name plcyname is specified as the first argument. Assume that this policy has all three components specified in the order of ARRAY, SET, and TREE. As there is only one element allowed in an array component, no parenthesis is required. Selem_list and Telem_list are the list of elements separated by a comma (,).

Example 4-111 Syntax of SECLABEL_BY_COMP

```
SECLABEL_BY_COMP (‘plcyname’, ‘elemA1:(Selem_list):(Telem_list)’)
```

Example 4-112 shows that usr8 is granted exemption on sales_plcy on all rules. usr8 is a market sales analyst. As a market analyst, the user inserts some records into the sales_data table, and the label of the record inserted depends on who the user interacts with or which region the user visits. The example shows two INSERTs being done using the SECLABEL_BY_COMP() function.

Example 4-112 Usage of SECLABEL_BY_COMP() function

```
CREATE SECURITY LABEL sales_plcy.mktg_sales_analyst
                       COMPONENT org_position 'Staff',
                       COMPONENT region  'HeadQuarters';

GRANT SECURITY LABEL sales_plcy.mktg_sales_analyst TO "usr8";
GRANT EXEMPTION ON RULE ALL FOR sales_plcy TO "usr8";
```

```
INSERT INTO sales_data (ulabel, id, name, product, sale_total,
date_sold )  VALUES (SECLABEL_BY_COMP('sales_plcy', 'Staff:Atlanta'),
3, "John Doe", "prod-b ", 133.00, "10-26-2007");

INSERT INTO sales_data (ulabel, id, name, product, sale_total,
date_sold )  VALUES (SECLABEL_BY_COMP('sales_plcy',
'Director:(West,East)'), 3, "John Doe", "prod-b ", 133.00,
"10-26-2007");
```

SECLABEL_BY_NAME

This function takes the name of the label as an argument and returns an encoded label value to be inserted into the table. Example 4-113 shows the syntax of the SECLABEL_BY_NAME() function.

Example 4-113 Syntax of SECLABEL_BY_NAME() function

```
SECLABEL_BY_NAME('plcyname', 'labelname')
```

Example 4-114 demonstrates how to use the SECLABEL_BY_NAME() function.

Example 4-114 Usage of SECLABEL_BY_NAME() function

```
INSERT INTO sales_data (ulabel, id, name, product, sale_total,
date_sold )  VALUES (SECLABEL_BY_NAME('sales_plcy', 'sales_vp'), 3,
"John Doe", "prod-b ", 133.00, "10-26-2007");
```

SECLABEL_TO_CHAR

This function takes the security policy and the name of the IDSSECURITYLABEL column as arguments and returns the security label in a character format. This function is typically used in SELECT statements. Example 4-115 shows the syntax. The first argument *plcyname* represents the policy name. The second argument *colname* represents the IDSSECURITYLABEL column in the table.

Example 4-115 Syntax of SECLABEL_TO_CHAR() function

```
SECLABEL_TO_CHAR('plcyname', colname)
```

Example 4-116 shows the usage of the SECLABEL_TO_CHAR function in a SELECT query.

Example 4-116 Usage of SECLABEL_TO_CHAR() function

```
SELECT SECLABEL_TO_CHAR('sales_plcy', ulabel) from sales_data;

(expression)  Staff:Atlanta

(expression)  Director:(West,East)

(expression)  VP:HeadQuarters
```

4.2.12 Referential integrity (RI) scans

Referential constraints are enforced for LBAC protected tables. Referential constraints can only be enforced if parent and child tables are protected by the same security policy. No RI enforcement is done if the parent and child are protected by different security policies.

The following cases apply for LBAC enforcement with RI:

1. Parent table is unprotected, child table is unprotected.

   In this scenario, as the tables are unprotected, RI enforcement is done in a normal manner. Refer to any book on SQL to learn more on this topic.

2. Parent table is protected, child table is unprotected.

   While inserting into the parent table, LBAC enforcement is done as the table is protected. While inserting into the child table, RI enforcement is done as follows, depending on whether the parent table has row or column protection:

   – If row protection is enabled for the parent table, and if at least one row is found in the parent table that is dominated by the user’s read label, then insert into the child table will be successful, provided that RI constraints are satisfied.

Example 4-117 shows tables t1 and t2. Table t1 has row protection enabled. Table t2 is unsecured and has a referential constraint to the primary key column in t1.

Example 4-117 Shows the creation of tables t1 and t2

```
create table t1 (seclabel IDSSECURITYLABEL,
                 i int PRIMARY KEY,
                 j char (20) )
                 security policy myplcy;

create table t2 (k int REFERENCES t1 (i), p int);
```

Table 4-16 shows that the rows in table t1 are protected at different sensitivity labels. The first two rows are protected at label mylbl_1 and the next row is protected at label mylbl4. Note that mylbl_4 dominates mylbl_1.

Table 4-16 Shows rows in table t1

i	j	Label
10	20	mylbl_1
20	30	mylbl_1
30	40	mylbl_4

Example 4-118 shows that the user with security label mylbl_1 can insert rows into t2 when RI constraint is satisfied. If the user inserts the value 10 for the referencing column, the insert is successful, as the user can see the row with primary key value 10 in the parent's table. But, if the user tries to insert the value 30 in the referencing column, an error is returned because the corresponding row in the parent table is not dominated by the user's label. A user with the label mylbl_4 can insert the value 30 in the child table.

Example 4-118 Demonstrates RI checks with LBAC (insert into child)

```
> insert into t2 values (10, 20);

1 row(s) inserted.

> insert into t2 values (30, 50);

  691: Missing key in referenced table for referential constraint
(usr2.r101_3).
```

```
Error in line 1
Near character position 29
>
```

   – If column protection is enabled for the parent table, and if the protected column is the parent table's primary key, then insert into the child is successful only if the user's label dominates column's label.

3. Parent table is unprotected, child table is protected.

   While deleting from the child table, LBAC enforcement is done as a protected table. When deleting from the parent table, RI enforcement is done as follows.

   If row protection is enabled for the child table, then delete from the parent table cannot be done if at least one row exists in the child table that satisfies the RI constraint.

   Example 4-119 shows the schema creation for tables t3 and t4. The parent table t3 is unprotected. The child table t4 has row protection enabled and also has a referential constraint to the parent table on column k.

Example 4-119 Another schema to demonstrate RI check with LBAC

```
create table t3 (k int primary key);

create table t4 (sl idssecuritylabel, k int references t3(k)
                on delete cascade ,
                name char(64) primary key)
                security policy  myplcy;
```

Table 4-18 on page 141 shows two rows in table t4. The first row is protected at label mylbl_1 and the second row is protected at label mylbl_4. The label mylbl_4 dominates mylbl_1. Table 4-17 on page 141 shows the corresponding rows in parent table t3. A user with label mylbl_1 can delete a row with the k value 35 from the parent table. But when the same user tries to delete the row with the k value 30, an LBAC error is encountered, as mylbl_4 dominates the user label.

Example 4-120 demonstrates how RI is enforced with LBAC. The user with the label mylbl tries to delete rows from t3. In cases where the user label dominates the row label, the deletion is successful. Otherwise, an LBAC error is returned.

Example 4-120 Another demonstration of RI check with LBAC (delete from parent)

```
> delete from t3 where k=30;

 8250: User does not have the LBAC credentials to perform DELETE on
table (usr2.t4).
Error in line 1
Near character position 24
> delete from t3 where k=35;

1 row(s) deleted.
```

Table 4-17 Shows the rows from table t3 (parent table)

k
35
30

Table 4-18 Shows the rows from table t4 (child table)

k	Name	Label
35	“name1”	mylbl_1
30	“name”2”	mylbl_4

If column protection is enabled for the child table, and if the protected column is the child table’s primary key, then delete from the parent table cannot be done if at least one row exists in the child table that satisfies the RI constraint.

4. Both parent and child tables are protected.

   This is a combination of cases (2) and (3). For insertions into the child table, refer to (2). For deletions from the parent table, refer to (3).

4.2.13 SETSESSIONAUTH privilege (in the context of LBAC)

We discussed the SETSESSIONAUTH privilege in the context of DAC in 4.1.14, “SETSESSIONAUTH privilege” on page 96. Here we present more details from the LBAC perspective.

When a user executes the SET SESSION AUTHORIZATION statement, the user acquires LBAC privileges in addition to DAC privileges. For this reason, DBSECADM has to be prudent while granting the SETSESSIONAUTH privilege to users. If a database is migrated from a historical version (prior to Version 11 that did not support LBAC), users with the DBA privilege are automatically granted the SETSESSIONAUTH privilege in the migration process. The DBSECADM has to explicitly revoke the SETSESSIONAUTH privilege from all of these users and grant the privilege only to authorized users.

4.2.14 Loading and unloading utilities

The load and unload utilities provided by IDS work seamlessly with LBAC support. The user doing the unload of a protected table can only see the rows dominated by the user's READ label. Similarly, while doing the load operation, LBAC access rules are applied. In order make these utilities work with data labels, exemptions may have to be granted.

We portray the utilities and LBAC-related restrictions that come with using these utilities as follows:

- **`dbexport`/`dbimport`**

  You must be user informix or must have DBA privileges on a database in order to use **`dbexport`**. Furthermore, if you are exporting a protected table, exemptions have to be granted so that all rows (with different labels) are exported correctly. While doing **`dbimport`**, DBA privilege on the database is given to the user who runs **`dbimport`**.

- **`unload`/`load`/`dbload`**

  The **`unload`** command is executed by a user using dbaccess. Only the rows with the data label that the user's label dominates are unloaded. As there will be rows with different labels, loading the file back into the table may not work, as LBAC WRITE access rules are violated. The DBSECADM needs to grant exemption to the users doing load or dbload, so that they will be able to load data successfully.

- **`onunload`/`onload`**

  These are faster versions of load utilities. The **`onunload`** utility unloads a database or table in binary form in disk-page units. As the **`onload`** utility also loads binary data, no specific LBAC conditions or restrictions apply to these utilities.

- High-performance loader

  Exemptions have to be granted to the users to unload and load protected tables using the high-performance loader.

4.3 Column-level encryption

Encryption is the mechanism of transforming data from an understandable form to a non-understandable form using some key, and keeping it understandable for those who know the key. This ensures the confidentiality of data. Only a user who knows the key can decrypt and read the encrypted data. IDS provides column-level encryption that is used to encrypt specific columns in specific tables. This feature is supported in IDS Version 10 and onwards. While inserting or updating data, a user has to mention the encryption cipher (logic) and the secret password. IDS encrypts the data using the specified cipher logic and storing it in the specified column. And only after providing the correct secret password is the user able to decrypt and read the data.

IDS does not store the encryption password in any table in plain text format (however, there are exceptions, which we discussed in “Password protection” on page 161) for security reasons. Hence, there is no way to retrieve it. To circumvent this situation IDS provides an HINT option to store the hint of the password in plain text format, which can be used to recollect the password.

IDS provides ENCRYPT_AES() and ENCRYPT_TDES() built-in functions that can store the data in a column in an encrypted form. It does not set any column property to do so. That is up to you. You can store unencrypted data as well in same column. ENCRYPT_AES uses an AES cipher and ENCRYPT_TDES() does triple encryption using the DES cipher. The size of data encrypted by ENCRYPT_TDES is less than the encrypted data generated by the ENCRYPT_AES function. IBM has not changed the logic of these standard ciphers. To conform to the requirement of Federal Information Processing Standards (FIPS 140.2), the internal cryptography is changed from using the OpenSSL library to using the IBM standard cryptography library called ICC.

While encrypting the data you need to provide a secret encryption password. A secret password can be set either globally using the SET ENCRYPTION PASSWORD statement for a session, or you can mention it in an encryption function. A password must be more than six character and less than 128 characters.

The built-in functions provided to decrypt the data are DECRYPT_CHAR and DECRYPT_BUNARY for character and binary data types, respectively.

IDS supports encryption for the following data types:

- CHAR, VARCHAR, NCHAR, NVARCHAR, LVARCHAR
- SMALLINT, INTEGER, INT8, DECIMAL, SMALLFLOAT, FLOAT
- DATE, DATETIME, INTERVAL, BOOLEAN
- BLOB, CLOB
- The various synonyms for these type names

Since the length of the encrypted string varies as per the original value of the column, the encrypted data can be stored in columns with the data type as follows:

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

This mechanism does not protect data over a network. Data transferred over the network is in an unencrypted form. In order to have the data encrypted, the server and client should be configured to use the CSM module ENCCCSM.

The column-level encryption is handled by the separate virtual processor (VP) named *encrypt*. If the *encrypt* option of the VPCLASS configuration parameter is not set, the encrypt VP starts when the first request for the encrypted column is sent to the server. You can add or remove encrypt VPs using *onmode* utility.

For example, to add two more encrypt VPs:

```
onmode -p 2 encrypt
```

To drop one encrypt VP:

```
onmode -p -1 encryp
```

4.3.1 Encryption of data at column level

The built-in functions provided for encryption are ENCRYPT_AES() and ENCRYPT_TDES(), which use DES and Triple DES ciphers, respectively.

Example 4-121 shows the syntax of the built-in encryption functions.

Example 4-121 Syntax of the built-in encryption functions

```
ENCRYPT_AES('data value for a column'[,'encryption password'])
ENCRYPT_TDES('data value for a column'[,'encryption password'])
```

Example 4-122 demonstrates the usage of the ENCRYPT_AES() function. We create a table act_details and encrypt the credit card number and social security number. As the output shows, the data is encrypted.

Example 4-122 The usage of the ENCRYPT_AES() function

```
CREATE TABLE act_details
(
 fname  CHAR(15),
 lname  CHAR(15),
 card_no CHAR(16),
 ssn_no  CHAR(11),
 address CHAR(100),
 balance DECIMAL(16,2),
 act_no CHAR(12),
 enc_card_no CHAR(67),
 enc_ssn_no CHAR(43)
);


INSERT INTO act_details
VALUES ("Julia","Robert","1111234567890000", "111-12-3456",
"Milpitas California","100","003901011111",
ENCRYPT_AES("1111234567890000","redbook"),
ENCRYPT_AES("111-12-3456","redbook"));


SELECT * FROM act_details;

Output shows:
=============
fname        Julia
lname        Robert
card_no      1111234567890000
ssn_no       111-12-3456
address      Milpitas California
balance      100.00
act_no       003901011111
```

```
enc_card_no  0Jtn/AAAAIAXT5PogWllQZ4M6uISzW1nGlrjO8kiNEcJpDq9Su1wYJcoMGHvwGO3Q=
             =
enc_ssn_no   0WKf/AAAAEAq/q7GH1fuO+Q8sX9TykYSKU78iaFz1Vh
```

We can set the encryption password at the session level using the SET ENCRYPTION PASSWORD statement to have one encryption password for all the rows inserted.

Example 4-123 demonstrates the usage of the SET ENCRYPTION PASSWORD statement.

Example 4-123 The usage of SET ENCRYPTION PASSWORD statement

```
SET ENCRYPTION PASSWORD "redbook";

INSERT INTO act_details
VALUES ("Julianne","Moore","2221234567890000","222-12-3456",
"San Jose California","100","003901099999",
ENCRYPT_AES("2221234567890000"),
ENCRYPT_AES("222-12-3456"));

INSERT INTO act_details
VALUES ("Lindsay","Lohan","3331234567890000","333-12-4356",
"Santa Clara California","100","003901011111",
ENCRYPT_AES("3331234567890000"),
ENCRYPT_AES("333-12-3456"));

SELECT card_no,enc_card_no,ssn_no,enc_ssn_no FROM act_details;

Output shows:
=============
card_no      1111234567890000
enc_card_no  0Jtn/AAAAIAXT5PogWllQZ4M6uISzW1nGlrjO8kiNEcJpDq9Su1wYJcoMGHvwGO3Q=
             =
ssn_no       111-12-3456
enc_ssn_no   0WKf/AAAAEAq/q7GH1fuO+Q8sX9TykYSKU78iaFz1Vh

card_no      2221234567890000
enc_card_no  0lGv/AAAAIAcSKn8Tp1gI6XxMFjHFCMjQTL19GoHOqysy8kHq7orr8+a36Yn7EV1w=
             =
ssn_no       222-12-3456
enc_ssn_no   0o1z/AAAAEAxr+lTqV4cWciY4tqsvp+/eb6Yo2rZkm/

card_no      3331234567890000
enc_card_no  0+gX/AAAAIA3AXPTCJY2mQwuy5E6l8GSFZMlR2VMckrfZ6lapJvZu4VPgfNT+cKLg=
             =
ssn_no       333-12-4356
```

```
enc_ssn_no   0Ocb/AAAAEALmvgxmWYec/AtkyaX1N9OhE/XQ88Cz+m

3 row(s) retrieved.
```

For maintaining security, the encrypted value is different at different times for the same original value during SELECT, UPDATE, or INSERT statements. Example 4-124 demonstrates the different encrypted values for the same original value.

Example 4-124 Different encrypted values for same original value

```
SELECT ENCRYPT_AES(card_no,"redbook") FROM act_details
WHERE fname="Julia";

Output:
(expression)
00S7/AAAAIAGvnJcewp9lbHQEtlK4/jpJMTjZax6HAdh6QGcMtn4YerkB7IW/pMqw
       ==

SELECT ENCRYPT_AES(card_no,"redbook") FROM act_details
where fname="Julia"

Output:
(expression)
0yjX/AAAAIAkZDgXsS1jsnH6LtfmT+5C1LRgPBVI2BIw+qTsJUn3hAhHIDiQNff2Q
==
```

You can have different encryption password and encryption ciphers for different columns in the same row or same column in a different row. You also can use the encryption function in SELECT and UPDATE, as well as the WHERE clause.

If you wish to change the existing unencrypted data to encrypted data you can use the UPDATE statement. See Example 4-125.

Example 4-125 The usage of the encrypt function in the UPDATE statement

```
INSERT INTO act_details VALUES ("Susan","Sarandon","4441234567890000",
"444-12-3456","San Jose California","100","003901022222",
"4441234567890000","444-12-3456");

SELECT * FROM act_details
WHERE fname="Susan"

fname        Susan
lname        Sarandon
card_no      4441234567890000
ssn_no       444-12-3456
```

```
address      San Jose California
balance      100.00
act_no       003901022222
enc_card_no  4441234567890000
enc_ssn_no   444-12-3456

UPDATE act_details
SET enc_ssn_no=ENCRYPT_AES(enc_ssn_no,"redbook"),
enc_card_no=ENCRYPT_AES(enc_card_no,"redbook")
WHERE fname="Susan";

SELECT * FROM act_details
WHERE fname="Susan"

fname        Susan
lname        Sarandon
card_no      4441234567890000
ssn_no       444-12-3456
address      San Jose California
balance      100.00
act_no       003901022222
enc_card_no  0hnn/AAAAIA6LvJdb2UBfkvklghmGDOXEwRs8GpTOWI7lBWHkLUT1H+bc4mdZKmaA=
=
enc_ssn_no   0Av3/AAAAEAWhhgUOUgmdQocXTXIO7fk9K+ObsSMua6
```

4.3.2 Retrieving encrypted data

A user can read encrypted data only if he knows the encryption password. IDS provides two built-in functions, DECRYPT_CHAR() and DECRYPT_BINARY(), to read encrypted data. DECRYPT_CHAR() is used to decrypt the character data type and DECRYPT_BINARY() is used to decrypt the binary data type.

Example 4-126 shows the syntax of the decryption functions.

Example 4-126 Syntax of decryption functions

```
DECRYPT_CHAR(‘column name’[,’encryption password’])
DECRYPT_BINARY(‘column name’[,’encryption password’])
```

Example 4-127 demonstrates the usage of the decrypt function. We read the encrypted data stored in the table act_details.

Example 4-127 The usage of decrypt function

```
SELECT fname,lname,ssn_no,DECRYPT_CHAR(enc_ssn_no,"redbook"),
card_no, DECRYPT_CHAR(enc_card_no,"redbook")
```

```
FROM act_details;

fname         Julia
lname         Robert
ssn_no        111-12-3456
(expression)  111-12-3456
card_no       1111234567890000
(expression)  1111234567890000

fname         Julianne
lname         Moore
ssn_no        222-12-3456
(expression)  222-12-3456
card_no       2221234567890000
(expression)  2221234567890000

fname         Lindsay
lname         Lohan
ssn_no        333-12-4356
(expression)  333-12-3456
card_no       3331234567890000
(expression)  3331234567890000

fname         Susan
lname         Sarandon
ssn_no        444-12-3456
(expression)  444-12-3456
card_no       4441234567890000
(expression)  4441234567890000
```

If you have multiple columns encrypted with same password then you can avoid giving password to each and every decrypt function call. We can use the SET ENCRYPTION PASSWORD statement in such situations.

Example 4-128 demonstrates the usage of the SET ENCRYPTION PASSWORD statement to read the encrypted data.

Example 4-128 The usage of SET ENCRYPTION PASSWORD statement

```
SET ENCRYPTION PASSWORD "redbook";
SELECT fname,lname,ssn_no,DECRYPT_CHAR(enc_ssn_no),
card_no,DECRYPT_CHAR(enc_card_no)
FROM act_details;

fname         Julia
lname         Robert
ssn_no        111-12-3456
(expression)  111-12-3456
```

```
card_no       1111234567890000
(expression)  1111234567890000

fname         Julianne
lname         Moore
ssn_no        222-12-3456
(expression)  222-12-3456
card_no       2221234567890000
(expression)  2221234567890000

fname         Lindsay
lname         Lohan
ssn_no        333-12-4356
(expression)  333-12-3456
card_no       3331234567890000
(expression)  3331234567890000

fname         Susan
lname         Sarandon
ssn_no        444-12-3456
(expression)  444-12-3456
card_no       4441234567890000
(expression)  4441234567890000
```

If you have different ciphers or different encryption passwords used in different rows for the encrypted column, you need to separately select those rows by providing the appropriate encryption password. The decrypt function does not accept multiple encryption passwords, and neither does the SET ENCRYPTION PASSWORD statement.

You can use the decrypt function in the WHERE clause as well. Example 4-129 demonstrates the usage of the decrypt function in the WHERE clause.

Example 4-129 The usage of decrypt function in where clause

```
SELECT fname,lname,enc_ssn_no,ssn_no FROM act_details WHERE
DECRYPT_CHAR(enc_ssn_no,"redbook")="JLM-123-4567"

Output shows:
=============
fname      Julia
lname      Robert
enc_ssn_no 0WKf/AAAAEAq/q7GH1fuO+Q8sX9TykYSKU78iaFz1Vh
ssn_no     111-12-3456
```

4.3.3 Impact of column-level encryption on existing application

Encryption of data can have an impact on an application if the application is not aware of it. The application retrieving the data having no knowledge of data encryption receives data in an encrypted form and will be of no use to it. In such a situation either the application needs to be changed or you should hide the data encryption from the application.

To hide column-level data encryption from an application, you should create a view and let the application use the view instead of the table. And for insert, update, and delete, write INSTEAD OF triggers.

Encrypting data can impact the existing trusted applications. The data retrieved are encrypted and is of no use. The work-around is to create a view with data decrypted and let the application use the view instead of the base table. For insert, update, and delete, write INSTEAD OF triggers, which ensure that the data set for a column that is supposed to be encrypted is in encrypted format.

In Example 4-130 we show the view and triggers for hiding encryption data for the trusted application. The base table act_master has enc_ssn_no and enc_card_no columns encrypted.

Example 4-130 Hiding encryption data for trusted application

```
RENAME TABLE act_master AS act_master_enc;

CREATE VIEW act_master (fname,lname,card_no,ssn_no,address,balance,
act_no,enc_card_no,enc_ssn_no) AS
SELECT fname, lname, card_no, ssn_no,address, balance, act_no,
DECRYPT_CHAR(enc_card_no), DECRYPT_CHAR(enc_ssn_no)
FROM act_master_enc;
SET ENCRYPTION PASSWORD 'redbook';

CREATE TRIGGER ti_on INSTEAD OF INSERT ON act_master
REFERENCING NEW as new
FOR EACH ROW
(INSERT INTO act_master_enc
(fname, lname, card_no, ssn_no, address, balance, act_no, enc_card_no,
enc_ssn_no)
VALUES
(new.fname, new.lname, new.card_no, new.ssn_no, new.address,
new.balance, new.act_no ,ENCRYPT_AES(new.enc_card_no ),
ENCRYPT_AES(new.enc_ssn_no )));

CREATE TRIGGER tu_on INSTEAD OF UPDATE ON act_master
REFERENCING OLD as old NEW AS new
```

```
FOR EACH ROW
(UPDATE act_master_enc SET fname = new.fname ,lname = new.lname ,
card_no = new.card_no, ssn_no = new.ssn_no, address = new.address,
balance = new.balance, act_no = new.act_no,
enc_card_no = ENCRYPT_AES(new.enc_card_no),
enc_ssn_no = ENCRYPT_AES(new.enc_ssn_no)
WHERE (fname = old.fname));

CREATE TRIGGER td_on INSTEAD OF DELETE ON act_master
REFERENCING OLD AS old
FOR EACH ROW
( DELETE FROM act_master_enc  WHERE (fname = old.fname ) );

SET ENCRYPTION PASSWORD ‘redbook’;

SELECT * FROM act_master;
```

Output shows unencrypted data:

```
================================
fname        Julia
lname        Robert
card_no      1111234567890000
ssn_no       111-12-3456
address      Milpitas California
balance      100.00
act_no       003901011111
enc_card_no  1111234567890000
enc_ssn_no   111-12-3456
:
:
:
4 row(s) retrieved
```

4.3.4 Changing encryption password

While storing the encrypted data in a column, you need to mention the encryption password. For security reasons, many organizations define a security policy requiring changing all passwords periodically. If you wish to change the encryption password to conform to the security policy of your organization, you need to change it through SQL. You can change the encryption password for an individual row or can change it for multiple rows globally. For changing the password of multiple rows, the initial encryption password of those rows should be the same.

Example 4-131 shows the SQL to change the encryption password for a single row.

Example 4-131 SQLs to change the encryption password for a single row

```
SELECT fname,ssn_no,card_no,enc_ssn_no,enc_card_no,
DECRYPT_CHAR(enc_ssn_no,"redbook"),DECRYPT_CHAR(enc_card_no,"redbook")
FROM act_details
WHERE fname="Julia"

Output shows:

fname          Julia
ssn_no         111-12-3456
card_no        1111234567890000
enc_ssn_no     0WKf/AAAAEAq/q7GH1fuO+Q8sX9TykYSKU78iaFz1Vh
enc_card_no    0Jtn/AAAAIAXT5PogWllQZ4M6uISzW1nGlrjO8kiNEcJpDq9Su1wYJcoMGHvwGO3Q
               ==
(expression)   111-12-3456
(expression)   1111234567890000


UPDATE act_details SET
enc_card_no=ENCRYPT_AES(DECRYPT_CHAR(enc_card_no,"redbook"),"infocen")
,enc_ssn_no=ENCRYPT_AES(DECRYPT_CHAR(enc_ssn_no,"redbook"),"infocen")
WHERE fname="Julia"

1 row(s) updated.

SELECT fname,ssn_no,card_no,enc_ssn_no,enc_card_no,
DECRYPT_CHAR(enc_ssn_no,"redbook"),DECRYPT_CHAR(enc_card_no,"redbook")
FROM act_details
WHERE fname="Julia"

Output shows:

26008: The internal decryption function failed.

SELECT fname,ssn_no,card_no,enc_ssn_no,enc_card_no,
DECRYPT_CHAR(enc_ssn_no,"infocen"),DECRYPT_CHAR(enc_card_no,"infocen")
FROM act_details
WHERE fname="Julia"

Output shows:

fname          Julia
```

```
ssn_no         111-12-3456
card_no        1111234567890000
enc_ssn_no     08wz/AAAAEA8PPDTgtbViEQLYEiaz0iugrvhy4XN+gn
enc_card_no    0sU7/AAAAIAyVgI6rjB/d1rYp2rmZXUwht04JB+9Wz52iXLyklMOwfs+Wh6JTBh8w
               ==
(expression)   111-12-3456
(expression)   1111234567890000
```

To change the encryption password of multiple rows, you can use the SET ENCRYPTION PASSWORD statement. Example 4-132 demonstrates the SQL to change the encryption password of multiple rows.

Example 4-132 The SQL to change the encryption password of multiple rows

```
SELECT fname,ssn_no,card_no,enc_ssn_no,enc_card_no,
DECRYPT_CHAR(enc_ssn_no,"redbook"),DECRYPT_CHAR(enc_card_no,"redbook")
FROM act_details
WHERE fname!="Julia"

Output shows:
fname          Julianne
ssn_no         222-12-3456
card_no        2221234567890000
enc_ssn_no     0o1z/AAAAEAxr+lTqV4cWciY4tqsvp+/eb6Yo2rZkm/
enc_card_no    0lGv/AAAAIAcSKn8Tp1gI6XxMFjHFCMjQTL19GoHOqysy8kHq7orr8+a36Yn7EV1w
               ==
(expression)   222-12-3456
(expression)   2221234567890000

fname          Lindsay
ssn_no         333-12-4356
card_no        3331234567890000
enc_ssn_no     0Ocb/AAAAEALmvgxmWYec/AtkyaX1N9OhE/XQ88Cz+m
enc_card_no    0+gX/AAAAIA3AXPTCJY2mQwuy5E6l8GSFZMlR2VMckrfZ6lapJvZu4VPgfNT+cKLg
               ==
(expression)   333-12-3456
(expression)   3331234567890000

fname          Susan
ssn_no         444-12-3456
card_no        4441234567890000
enc_ssn_no     0Av3/AAAAEAWhhgUOUgmdQocXTXIO7fk9K+0bsSMua6
enc_card_no    0hnn/AAAAIA6LvJdb2UBfkvklghmGDOXEwRs8GpTOWI7lBWHkLUT1H+bc4mdZKmaA
               ==
(expression)   444-12-3456
(expression)   4441234567890000
```

To change the encryption password to "infocen" for all these rows.

```
SET ENCRYPTION PASSWORD "redbook";
UPDATE act_details
SET enc_card_no=ENCRYPT_AES(DECRYPT_CHAR(enc_card_no),"infocen")
,enc_ssn_no=ENCRYPT_AES(DECRYPT_CHAR(enc_ssn_no),"infocen")
WHERE fname!="Julia"

3 row(s) updated.

SELECT fname,ssn_no,card_no,enc_ssn_no,enc_card_no,
DECRYPT_CHAR(enc_ssn_no,"redbook"),DECRYPT_CHAR(enc_card_no,"redbook")
FROM act_details
WHERE fname!="Julia"

26008: The internal decryption function failed.

SELECT fname,ssn_no,card_no,enc_ssn_no,enc_card_no,
DECRYPT_CHAR(enc_ssn_no,"infocen"),DECRYPT_CHAR(enc_card_no,"infocen")
FROM act_details
WHERE fname!="Julia"

fname         Julianne
ssn_no        222-12-3456
card_no       2221234567890000
enc_ssn_no    0aZb/AAAAEAzIeaOjyrut8Mqu+/cG2dRbh+WMqXDkvc
enc_card_no   06xT/AAAAIAIJmCs26RGT4iRGFeIpas1g/aIx3kxmXEdujryOSIcyi5fMNR2xn3Xw
              ==
(expression)  222-12-3456
(expression)  2221234567890000

fname         Lindsay
ssn_no        333-12-4356
card_no       3331234567890000
enc_ssn_no    0q1T/AAAAEAFcxAcTcZspC8uYQZCbP/huteQyHBzxgx
enc_card_no   0Ms3/AAAAIACaK9CgLNLK2hIsjvq7IJxYWi9o6ztNuL4jDuu+SBYmyxEvHYlKuWhA
              ==
(expression)  333-12-3456
(expression)  3331234567890000

fname         Susan
ssn_no        444-12-3456
card_no       4441234567890000
enc_ssn_no    0mWb/AAAAEAz4MD/qGhkcTKkDeM3u8m3jl66KWvEJWV
enc_card_no   0yjX/AAAAIA0Rciz6PbtPC5ORC91BmPIQZVU+7yW11wCny2GyO1VFZRJ01kOu2YsQ
              ==
(expression)  444-12-3456
(expression)  4441234567890000
```

```
3 row(s) retrieved.
```

4.3.5 Using password hints

For a Web-based application, each row has a different encryption password. Consider the credit card database. Each credit card owner has her own password, which is treated as an encryption password so that nobody is able to access her information. For security reasons, we do not store the password in plain text format. In such a situation only a password HINT can help recollect the password if the user forget it. Example 4-133 shows the syntax of creating the password hint.

Example 4-133 Syntax of password hint

```
ENCRYPT_AES("column name","encryption password","hint")

SET ENCRYPTION PASSWORD "password" WITH HINT "hint"
```

If you forgot the password, you can use the GETHINT function to retrieve the hint, which may help you recollect your encryption password. The syntax for GETHINT function is:

```
SELECT GETHINT("encrypted column name") FROM tablename
```

Example 4-134 shows the usage of the HINT and GETHINT objects.

Example 4-134 The usage of HINT and GETHINT objects

```
CREATE TABLE states
( owner_name char(50), credit_card_no char(67) )

SET ENCRYPTION PASSWORD "redbook" WITH HINT "Book Name";
INSERT INTO states VALUES(1,ENCRYPT_AES("1234467891012345"));

1 row(s) inserted.

SELECT GETHINT(state_name) FROM states;

(expression) book name
```

4.3.6 Calculating the size of column storing encrypted data

Except for the original data of BLOB and CLOB data types, the encrypted data value is encoded in BASE64 format. The size of the encrypted data is always more than the size of the original data. If you use a password hint, it adds to the size. In our examples, you might have observed that we are storing a 12-character card number and SSN number but the size of columns enc_card_no and enc_ssn_no is 45 bytes.

The required storage size for encrypted data is sensitive to three factors:

- N - the number of bytes in the plain text
- Whether or not a hint is provided
- Which encryption function you use (ENCRYPT_TDES or ENCRYPT_TDES)

The following formula can be used to determine the required size of the column:

```
AES = B64(NGM(N, 16) + H + 8) + 11

Triple-DES = B64(NGM(N,   8) + H + 8) + 11

Where
H = 0 with no hint; H = 40 with hint.
NGM(x,y) – Next multiple of y that is greater than x.
NGM(x, y) = ((x + y) ÷ y) × y
B64(x) – Base-64 encoding size.
B64(x) = ((x + 2) ÷ 3) × 4
```

AES can be bigger than Triple-DES, but not by much.

The above formulas can be simplified depending on three factors:

- Encryption by ENCRYPT_TDES() with no hint:

  ```
  Encrypted size = (4 x ((8 x((N + 8)/8) + 10)/3) + 11
  ```

- Encryption by ENCRYPT_AES() with no hint:

  ```
  Encrypted size = (4 x ((16 x((N + 16)/16) + 10)/3) + 11)
  ```

- Encryption by ENCRYPT_TDES() with a hint:

  ```
  Encrypted size = (4 x ((8 x((N + 8)/8) + 50)/3) + 11)
  ```

- Encryption by ENCRYPT_AES() with no hint:

  ```
  Encrypted size = (4 x ((16 x((N + 16)/16) + 50)/3) + 11)
  ```

The integer division (/) returns an integer quotient and discards any remainder.

If the size of the column is less than the size of the encrypted data, the server does not give any error and stores the data by truncating it to column size. When the data is retrieved, the following error message is given:

```
26012: The internal base64 decoding function failed.
```

The size limit for columns of type VARCHAR and NVARCHAR is 255 bytes. If the data string is too long for these data types to store both the encrypted data and encryption overhead, the value returned by the encryption function is automatically changed from VARCHAR or NVARCHAR to a fixed CHAR or NCHAR value, with no trailing blanks in the encoded encrypted value.

One way to calculate the size of the column is using the function LENGTH, as shown in Example 4-135.

Example 4-135 Using LENGTH function to calculate column size

```
SELECT LENGTH(ENCRYPT_AES("1111234567890000",'redbook')) FROM
systables where tabid=1

returns:

(constant) 67
```

Table 4-19 lists the storage space requirements for the encrypted column.

Table 4-19 The storage space requirements for encrypted column

Input size (bytes)	Triple-DES (no hint)	AES (no hint)	Triple-DES (with hint)	AES (with hint)
1–7	35	43	87	99
8–15	43	43	99	99
16–23	55	67	107	119
24–31	67	67	119	119
32–39	75	87	131	139
40–47	87	87	139	139
100	163	171	215	227
200	299	299	355	355
500	605	707	747	759

The encrypted values of type BLOB or CLOB are not in BASE64 encoding format, and their size increased after encryption is independent of the original data size. The following formula can be use to calculate the encrypted data size (in bytes) for BLOB or CLOB values:

- Encryption by ENCRYPT_TDES:

  `N + H + 24 bytes`

- Encryption by ENCRYPT_AES encrypts:

  `N + H + 32 bytes`

Where N is the original size of the plain text and *H* is the size of the unencrypted hint string.

4.3.7 Encrypting complex data type

For data types other than CHAR(n), you need to store encrypted data in the CHAR(n) data type. If you wish to encrypt a simple INTEGER data type, you need to encrypt and store it in the CHAR data type.

You can encrypt the ROW data type and store it in the CHAR data type. A data type LIST is also treated in the same way.

Example 4-136 shows how you can encrypt and store the ROW data type using the encrypt function and retrieve the data back using the decrypt function.

Example 4-136 Encrypting and decrypting the ROW data type

```
-- Create Row type

CREATE ROW TYPE row_t ( x int, y char(10));

-- Create table

CREATE TABLE row_demo
(
 sr_no integer,
 row_col row_t,
 enc_row char(100)
);

-- Insert row(s)

INSERT INTO row_demo
VALUES ( 1, ROW(1,"abcd")::row_t,
ENCRYPT_AES(ROW(1,"abcd")::row_t::LVARCHAR ,'redbook'));

-- Retrieve row(s) without decryption

SELECT * FROM row_demo;

sr_no    1
row_col  ROW(1          ,'abcd      ')
enc_row
0I9z/AAAAIAE6dP4uU/+HhEO9p9f5F4GbjFm6bLloXI6jMViFlvqn2FEK23wOVBIQ==

-- Retrive row(s) with decryption

SELECT sr_no,row_col,DECRYPT_CHAR(enc_row,'redbook')::row_t FROM
row_demo;
sr_no         1
row_col       ROW(1          ,'abcd      ')
(expression)  ROW(1          ,'abcd      ')
```

4.3.8 Special considerations

Column-level encryption ensures the data confidentiality, but its implementation needs certain special considerations:

- Try to keep the encryption password the same for the column of all rows.

  This helps in selecting multiple rows with minimal WHERE clauses in a single simple query. But if you have personal data to hide, you can go for variant passwords.

- Do not create an index or a functional index on the encrypted column.

  The encryption mechanism always generates different encrypted strings for the same value. Hence, you cannot have an encrypted string for comparison, as it is always unpredictable. This totally defeats the purpose of having the index on a table. Another aspect of the problem is that you get a number of unpredictable distinct values for a column where you might be expecting two or three only.

  Let us consider a scenario in which you are storing a status of employee as present or absent (P or A). With the column encrypted you get number of different unpredictable values. Having an index on such columns is of no use.

- The encrypted data always requires more space than the unencrypted data.

  As we discussed above, the size difference is really considerable. If you are deciding to store the encrypted data in a database, you need to take care of creating dbspaces of proper size. Similarly, the utilities like dbexport and onunload consume more disk space to unload the data, as the data unloaded will be in encrypted format.

As the column storing encrypted data requires a bigger size, the size of the record also increases. This causes more space consumption in transaction logs. Hence, log size is also one of the important aspects that you need to take into special consideration while deciding on column level encryption.

Password protection

The most important aspect of column-level encryption is the encryption password. Any user who knows the encryption password can access and read the data. Hence, it is necessary that you protect the password. There are various situations in which a password can be exposed because it is stored in plain text by IDS in the database. For example, if you use an encryption password in a procedure or trigger, the password is stored in a system catalog table in plain text format and is reflected as it is in dbschema or dbexport output. Hence, you should avoid using an encryption password in the trigger and procedure. You should avoid actions that might compromise the secrecy of the password:

- Do not create a functional index using the decryption function. This stores a password in plain text format in a system catalog table.
- During communication, the client and the server exchange the password and hint. You should work with encrypted data on an insecure network.
- Do not store the password in triggers, procedures, or UDRs, as it will be stored in plain text. The dbschema or dbexport output export the password used in the trigger and procedure.
- Do not set the session password prior to creating a trigger, procedure, UDR, or view. Use the session password while accessing these elements. You can refer to the example of creating and using a trigger, which uses the decrypt function without a password in Example 4-130 on page 151.
- Avoid using the encryption password in the main query. Instead, you can create a temporary table, insert an encryption password in it, and perform a join query.

Example 4-137 illustrates the use of a temporary table to hide the encryption password from the base table act_details shown in Example 4-122 on page 145.

Example 4-137 Using temporary table to hide encryption

```
CREATE TEMP TABLE temp1
(
enc_ssn_pswd CHAR(10),
enc_card_pswd CHAR(10));

INSERT INTO temp1 VALUES("infocen","infocen");
```

```
SELECT fname,ssn_no,card_no,enc_ssn_no,enc_card_no,
DECRYPT_CHAR(enc_ssn_no,temp1.enc_ssn_pswd),
DECRYPT_CHAR(enc_card_no,temp.enc_card_pswd)
FROM act_details, temp1
```

Output from the SET EXPLAIN statement always displays the password and hint parameters as XXXXX, rather than displaying the actual password or hint values.

5

Client-server communication

This chapter provides the details about secured client-server communication. Secured communication stands for fine-grained access, data integrity, and data confidentiality. We discuss different types of communications and the corresponding configuration to ensure secured communication. The major topics we cover are:

- PAM support in IDS
- Configuring IDS to use encryption
- Configuring client to use encryption

5.1 Client-server communication overview

Knowing how client-server communication is established or handled by IDS can help you understand the configurations for secured client-server communication. IDS supports different types of protocols or mechanisms to establish client-server communication.

5.1.1 Shared memory (UNIX)

A shared memory connection uses shared memory to establish communication between server and client. A server, when started with nettype ipcshm configured in the onconfig and sqlhosts files, creates a shared memory. Whenever a client asks for a connection using a server name, it gets access to shared memory and communication starts.

Though shared memory provides fast access to the database, it has disadvantages such as data insecurity. A malicious application on the same machine can destroy or read messages of other local users.

This mechanism can be used for local connections only.

5.1.2 Stream-pipe connections (UNIX)

A stream-pipe is an inter process communication system provided by UNIX to establish communication between two process. A server, when started with nettype ipcstr configured in onconfig and sqlhosts file, allows a client to connect through a stream-pipe. Unlike shared memory connection, it is secured and also allows distributed queries between two or more database servers residing on the same machine. It is not supported by all platforms.

This mechanism can be used for local connections only.

5.1.3 Named-pipe connections (Windows)

Named pipes are application programming interfaces (APIs) for bidirectional inter process communication. It stores data in memory and retrieves it when requested. This mechanism can be used for local connections only.

5.1.4 Local loop-back connections

A connection established between a client and a server residing on the same host using network protocol and network services is called a local loop-back connection. It uses network services such as Transport Level Interface (TLI) or socket. Even though the database server uses network services, it does not put data out of network. This mechanism also can be used for network communication. Table 5-1 summarizes the communication mechanism supported by IDS.

Table 5-1 Communication mechanism supported by IDS

Connection type	Windows	UNIX	Local	Network
Sockets	X	X	X	X
TLI (TCP/IP)		X	X	X
TLI (IPX/SPX)		X	X	X
Shared memory		X	X	
Stream pipe		X	X	
Named pipe	X		X	

Figure 5-1 illustrates client-server communication. Irrespective of the mechanism used for communication, data security is of utmost importance. Access to an authorized user, data confidentiality, and data integrity ensure the data security in client server communication.

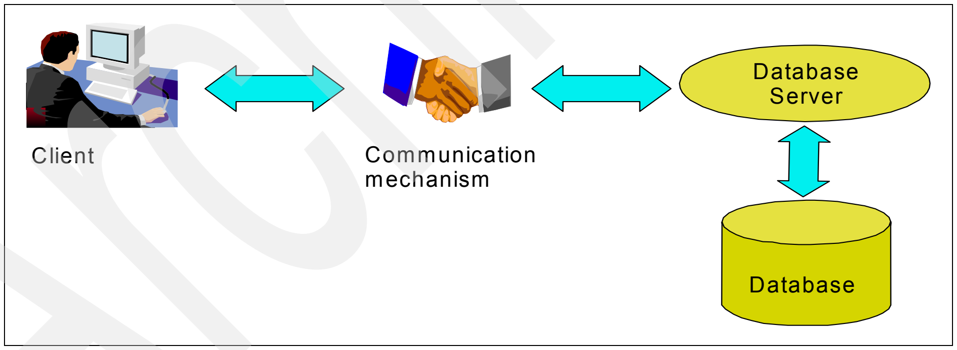


Figure 5-1 Client-server communication

Further sections in the chapter help you explore the authentication mechanism and data encryption mechanism in IDS.

5.2 The authentication mechanism in IDS

Authentication is the mechanism of verifying the identity of a user or an application. It ensures fine-grained access to the database. The database server allows only those users who confirm who they are to access data.

IDS conforms to the client-server architecture. A client can be a remote or a local user. You can configure your server for establishing a suitable authentication mechanism for fine-grained access.

Figure 5-2 is a high-level diagram demonstrating an authentication mechanism.

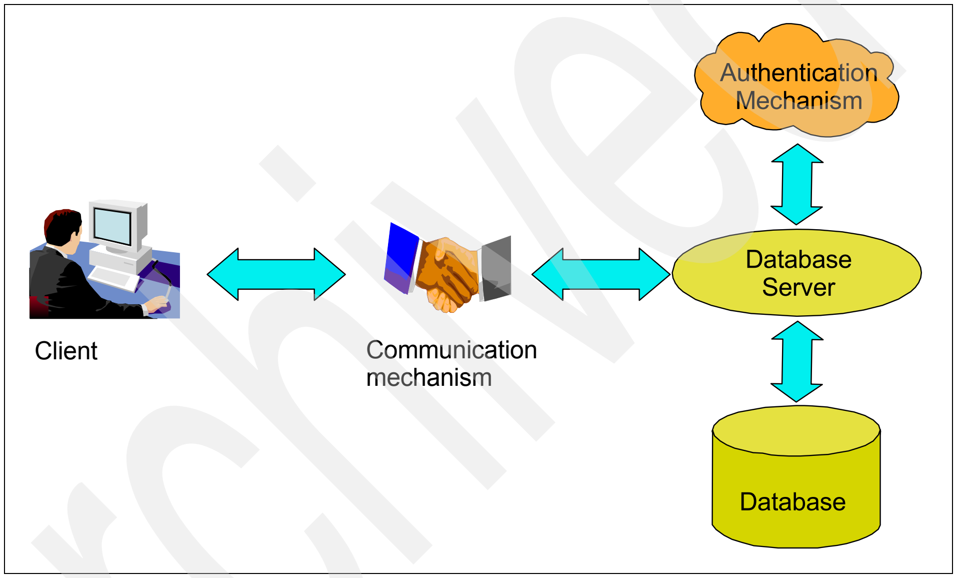


Figure 5-2 The high-level diagram demonstrating an authentication mechanism

A local connection is a connection between a client and a server residing in the same machine. A remote connection is a connection between a client and server residing in different machines. You can refer to Table 5-1 on page 165 for protocol supported for local and remote connections.

IDS supports a traditional authentication mechanism under which a user has to provide his ID and password to connect to a database. But you can configure the database server to add or modify an authentication mechanism. You can develop PAM or LDAP modules and configure a server to have a self-defined authentication mechanism. This works for local as well as remote connections.

For remote connection apart from PAM or LDAP, IDS supports default a authentication mechanism governed by network security files provided by OS.

5.2.1 Network security files

For network-based business models, IDS conforms the network authentication mechanism provided by OS. OS provides different configuration files such as /etc/hosts.equiv, $HOME/.netrc, and $HOME/.rhosts to implement trusted networks. Only trusted users have access to remote services on remote machines.

In this section we discuss systems used in our examples.

hosts.equiv file

The /etc/hosts.equiv file provides remote authentication. The users and hosts with entry to this file are treated as trusted. A trusted user can do remote login to the local host without providing a password. In the same way, a client application can connect to a server without providing a user ID and password if it is a trusted user.

Example 5-1 illustrates the usage of the hosts.equiv file. In this example, Server ol_myserver is on machine mercury.lenexa.ibm.com, and the client is on vulcan.lenexa.ibm.com. There is no entry of machine2 in machine1:/etc/hosts.equiv.

Example 5-1 Usage of hosts.equiv file

```
(command prompt$) dbaccess - -
>connect to ‘@ol_myserver’;
951: Incorrect password or user vdantale@vulcan.lenexa.ibm.com is not
known on the database server.

Interrupted system call
Error in line 1
Near character position 1
>
```

After adding entry of vulcan in mercury:/etc/hosts.equiv as :

```
vulcan
vulcan.lenexa.ibm.com
```

The connection established successfully.

```
(command prompt$) dbaccess - -
```

```
>connect to '@ol_myserver';
Connected.
>
```

This mechanism ensures that the connecting user or host is trusted. Use this file only if the user application does not pass the user ID and password. If this file does not exist, you can create it and add desired entries in it.

On Windows this file is in the %WINDIR%\system32\drivers\etc directory.

.rhosts

As an alternative for /etc/hosts.equiv, a user can add entries of hosts from which he can connect as a trusted user in the .rhosts file on server machine. This file is in a user's home directory. If not present, the user can create it and add the entries. This file can be used to individualize the trusted host or trusted user settings.

On Windows, a home directory is not assigned to a user. The administrator can add a home directory to the user's profile with user manager application.

Example 5-2 demonstrates the usage of the .rhosts file. If there is no trusted host entry in /etc/hosts.equiv and in $HOME.rhosts, a connection fails with error -951. After adding the entry of trusted hosts in $HOME/.rhosts, the connection is successful. A server instance ol_myserver is on machine mercury.lenexa.ibm.com, and we are trying to connect the server from machine vulcan.lenexa.ibm.com.

Example 5-2 Usage of .rhosts file

```
(command prompt$) dbaccess - -
>connect to '@ol_myserver';
951: Incorrect password or user vdantale@vulcan.lenexa.ibm.com is not
known on the database server.

Interrupted system call
Error in line 1
Near character position 1
>
```

After adding entry vulcan in mercury:$HOME/.rhosts as:

```
vulcan
vulcan.lenexa.ibm.com
```

The connection established successfully.

```
(command prompt$) dbaccess - -
>connect to ‘@ol_myserver’;
Connected.
>
```

.netrc

The .netrc information specifies identity data and is optional information. A user who does not have authorization to access the database server or is not on a computer that is trusted by the database server can use this file to supply a name and password that are trusted. A user who has a different user account and password on a remote computer can also provide this information.

On UNIX, the netrc information resides in the .netrc file in the user's home directory. Use any standard text editor to prepare the .netrc file. Windows maintains the netrc information in the registry keys. Use the Host Information tab of setnet32 to edit the netrc information.

If you do not explicitly provide the user password in an application for a remote server (that is, through the USER clause of the CONNECT statement or the user name and password prompts in DB–Access), the client application looks for the user name and password in the netrc information. If the user has explicitly specified the password in the application, or if the database server is not remote, the netrc information is not consulted.

The database server uses the netrc information regardless of whether it uses the default authentication policy or a communications support module.

Example 5-3 demonstrates the usage of .netrc. If we try to connect to the server instance ol_myserver present on machine mercury.lenexa.ibm from machine vulcan.lenexa.ibm.com without configuring /etc/hosts.equiv or $HOME/.rhosts or $HOME/.netrc, connection will fail with error -951. After adding user identity details and machine details in the $HOME/.netrc file, the connection is successful.

Example 5-3 Usage of .netrc

```
(command prompt$) dbaccess - -
>connect to ‘@ol_myserver’;
951: Incorrect password or user vdantale@vulcan.lenexa.ibm.com is not
known on the database server.

Interrupted system call
Error in line 1
Near character position 1
>
```

After adding entry of vulcan.lenexa.ibm.com and usr1 in mercury:$HOME/.netrc as :

```
machine vulcan login user1 password usr1pswd
```

Connection established successfully:

```
(command prompt$) dbaccess - -
>connect to '@ol_myserver';
Connected.
>
```

5.2.2 Pluggable authentication module (PAM)

A pluggable authentication module (PAM) is a framework that enables the user to develop and implement his own user authentication mechanism without making any changes in the application. This builds one more layer of authentication over the default network authentication mechanism. The PAM framework provides a set of APIs for account management, session management, authentication management, and password management.

IDS supports account management and password management APIs. The modes supported by IDS are password mode and challenge-response mode.

In password mode, the user password is sufficient to satisfy the authentication.

In the challenge response mode, a server raises a challenge and the client sends a response. A client gets access to the database only if the response is as expected.

Note: If the IDS server is configured for using PAM, the default authentication mechanism is ignored. Hence, if you wish to retain the default authentication mechanism, your PAM module should be smart enough to handle it.

The server can be configured for raising multiple challenges. Hence, a client application should be developed to respond to multiple challenges. The flow of client server communication is as follows:

1. The client sends a request to connect to the database server.
2. The database server checks a PAM service as configured in sqlhosts and the /etc/pam.conf file (/etc/pam.d/ for Linux®).

3. The database server loads a PAM module corresponding to a PAM service configured in /etc/pam.conf and executes it.
4. The database server forwards the challenge to the client.
5. The application must respond to the challenge using a callback function that is provided by the IBM Informix CSDK and the IBM Informix JDBC™ driver.
6. If the server to which the client is connecting is set up for a challenge, the application must register a callback function with CSDK or JDBC.
7. When CSDK or JDBC receives a challenge from the server, the challenge is forwarded to the application by the callback function.
8. The application must respond to the challenge.
9. CSDK or JDBC forwards the response to the database server.

Figure 5-3 shows the client-server communication when a server is configured to use PAM.

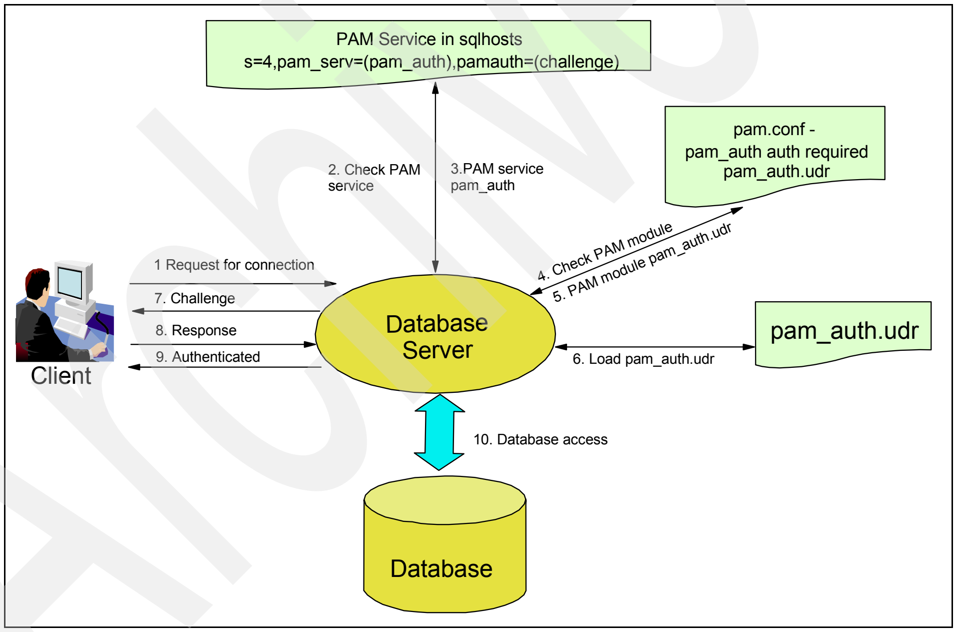


Figure 5-3 Overview of PAM used in IDS

Configuring IDS to use PAM

You can configure your IDS server to use PAM to add or modify the authentication mechanism. The following IDS clients APIs support PAM:

- ESQL/C
- ODBC
- JDBC

You can use DBSERVERALIAS to connect to a PAM configured server with other APIs.

Prerequisites

If you wish to use PAM, you need to check a few things:

- Your Informix database server must be on an operating system platform that supports PAM.
- Your client applications must be written using Client SDK (CSDK) Version 2.81 or later.
- You must have the appropriate PAM service configured in the operating system.
- You must know whether the PAM service simply accepts the given password or whether it uses a challenge-response protocol.
- If your PAM service uses a challenge-response protocol, you must modify your applications to handle the challenge and response. The application must be aware that the PAM module can raise multiple challenges.
- You must ensure that enterprise replication and high availability clusters are not affected by PAM authentication.
- You must modify the server entry in the sqlhosts file for both the client application and the database server (if they are on separate machines or in separate locations on a single machine).

Example 5-4 shows a sample program that can be used to check whether PAM is supported on your machine.

Example 5-4 A sample program to check whether PAM is supported on your machine

```
/* Program : sample_pam.c */
main()
{
   return(0);
}
```

Compile this program as:

```
command_prompt$) cc -o sample_pam.exe sample_pam.c -lpam
```

The -lpam compilation option includes a PAM library libpam.<shared object library suffix> and resolves PAM APIs.

If a program is compiled successfully, then PAM is supported on your machine.

PAM objects

For configuring IDS to use PAM, one should be aware of different objects of PAM:

- PAM library

  PAM library libpam.<shared object suffix> resides under /usr/lib/security or /lib/security. This library resolved PAM APIs.
- PAM configuration file

  PAM configuration file pam.conf, which contains the entries of services and relevant modules. You need to add entry of your service in this file. This is a little different for Windows and Linux, which we discuss later.
- IDS sqlhosts file

  An IDS sqlhosts file, INFORMIXSQLHOSTS, is used to define the PAM service and PAM mode to be used for authentication.
- PAM_STACKSIZE

  This configuration parameter is used to customize the stack size of IDS.
- PAM module

  A PAM module is a shared module that you want to use for self-defined authentication. This should be developed and copied under the directory where all your OS security libraries are kept.
- Callback function

  The callback function is a code in a client application to interact with the PAM module. This function has to be registered at the server.

Implementing PAM

In this section we demonstrate the implementation steps using an example. The following is the preliminary information for the example:

- Assumptions:

OS	Linux
PAM mode	Challenge-response
IDS server name	ol_myserver

Port number	9628
PAM module	pam_auth.udr
PAM service	pam_auth

- Authentication specifications:

  Whenever a client tries to connect to server ol_myserver, a server should raise a challenge and give database access to the client after getting a satisfactory response from the client. The challenges and responses are as follows:

  – Challenge1 - Security code: Response - RX#CODE
  – Challenge2 - Security number: Response - 9991024

Follow these steps to configure IDS to use PAM. Note that there is no specific order to these steps.

- Add PAM service entry to PAM configuration file.

  You need to add a PAM service entry for the pam_auth service. A PAM configuration file on UNIX is /etc/pam.conf, which contains entries of service and corresponding modules. The entry has the following format:

  ```
  <PAM service> <Module type> <Control flag> <Module path> <Options>
  ```

  Where module type can be auth, account, password, or session. Control flag can be required or requisite. The optional column is for specifying the options to be passed to the PAM module.

  Example 5-5 illustrates the entries in /etc/pam.conf on UNIX. At the end of the file, we add two pam_auth service entries for the user-defined PAM module, whereas pam_auth.udr is the actual user-defined module for authentication.

  Example 5-5 The entries in /etc/pam.conf on Sun™ Solaris™ OS

```
#
#ident  "@(#)pam.conf   1.20    02/01/23 SMI"
#
# Copyright 1996-2002 Sun Microsystems, Inc.  All rights reserved.
# Use is subject to license terms.
#
# PAM configuration
#

...

pam_auth auth    required    /usr/lib/security/sparcv9/pam_auth.udr
pam_auth account required    /usr/lib/security/sparcv9/pam_auth.udr
```

On Linux, each service has a configuration file under the /etc/pam.d directory, instead of one common file for all services. The entries in all the service files have the same format as follows:

```
<Module type> <Control flag> <Module path> <Options>
```

Where module type can be auth, account, password, or session. Control flag can be required or requisite. The optional column is for specifying the options to be passed of PAM module. Example 5-6 shows the login service file.

Example 5-6 Sample login service file under /etc/pam.d on Linux

```
#%PAM-1.0
auth       required     pam_securetty.so
auth       required     pam_stack.so service=system-auth
auth       required     pam_nologin.so
account    required     pam_stack.so service=system-auth
password   required     pam_stack.so service=system-auth
# pam_selinux.so close should be the first session rule
session    required     pam_selinux.so close
session    required     pam_stack.so service=system-auth
session    required     pam_loginuid.so
session    optional     pam_console.so
# pam_selinux.so open should be the last session rule
session    required     pam_selinux.so open
```

Similarly, you need to create a pam_auth file under /etc/pam.d for your PAM service pam_auth. The file for the pam_auth service should look like the one in Example 5-7.

Example 5-7 PAM service pam_auth configuration file

```
auth     required      pam_auth.so
account  required      pam_auth.so
```

On Windows, you need to create a pam.conf file under the %INFORMIXDIR%\dbssodir\etc directory, which should contain the entry of your PAM module.

Example 5-8 illustrates the entries of the pam.conf file on Windows. In this example, pam_auth.dll is the PAM module that you want to use, and pam_auth is the corresponding service.

Example 5-8 The entries of pam.conf file on WINDOWS

```
pam_auth auth required %INFORMIXDIR%\dbssodir\etc\pam_auth.dll
pam_auth account required %INFORMIXDIR\dbssodir\etc\pam_auth.dll
```

- Edit the IDS sqlhosts file.

  The server entry in the IDS sqlhosts file INFORMIXSQLHOSTS should have the PAM service and the mode mentioned in the same row. The PAM mode can be challenge or password. You can edit your sqlhosts file using any editor on a UNIX system.

  Example 5-9 illustrates the entry in INFORMIXSQLHOSTS of a PAM configured server on UNIX and Linux for challenge mode.

  Example 5-9 INFORMIXSQLHOSTS entry - challenge mode

```
ol_myserver ontlitcp vulcan tcp_serv s=4,pam_serv=(pam_chal),pamauth=(challenge)
```

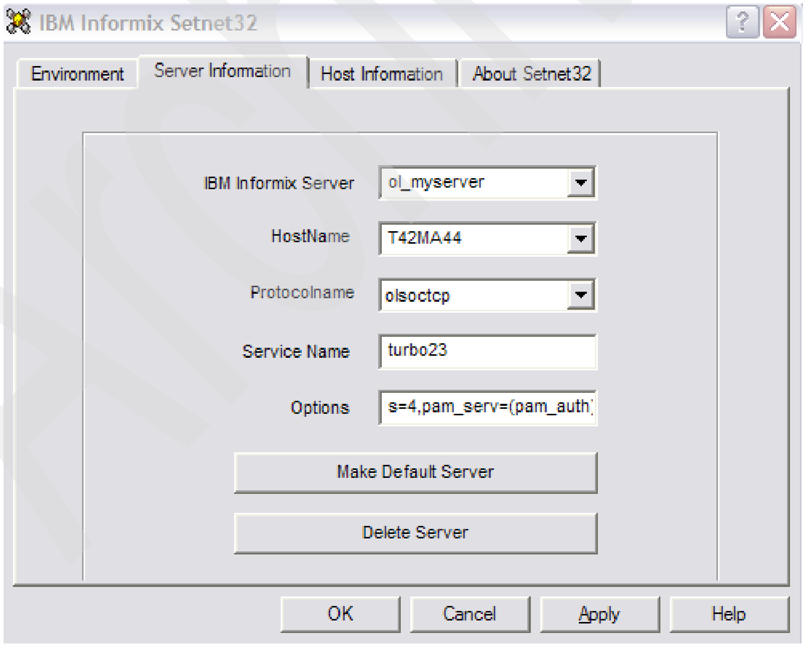
  Example 5-10 is a service entry for the password mode.

  Example 5-10 INFORMIXSQLHOSTS entry - password mode

```
ol_myserver ontlitcp vulcan tcp_serv s=4,pam_serv=(pam_pass), pamauth=(password)
```

  For Windows, you can use the setnet32 utility provided by the Informix Client Software Development Kit (CSDK) to configure PAM entries in sqlhosts.

  Figure 5-4 shows a PAM configured server on Windows. The complete entry for the Options field is `s=4,pam_serv=(pam_auth), pamauth = ( challenge)`.



IBM Informix Server: ol_myserver
HostName: T42MA44
Protocolname: olsoctcp
Service Name: turbo23
Options: s=4,pam_serv=(pam_auth)

Figure 5-4 Illustrates the entry of PAM configured server on Windows

- Configure PAM_STACKSIZE.

  A configuration parameter PAM_STACKSIZE has to be set in the onconfig file. The PAM loads OS and third-party shared libraries into the Informix thread. You cannot predict the stack size required for these modules. The default value of PAM_STACKSIZE on UNIX is 32 KB.

  You can use the PAM_STACKSIZE parameter to vary Informix stack size and accommodate unpredictable OS and third-party shared libraries. For example, on Linux some modules require 128 K as stack size. You can set PAM_STACKSIZE in such case to the appropriate value as follows:

  ```
  PAM_STACKSIZE 128
  ```

- Build PAM module library.

  The PAM module is a shared library that gets loaded and executed when you try to connect to the server. This library contains a code of rasing a challenge and validating a response. You need to build this library as per the authentication mechanism that you want and copy it under the directory where your OS security libraries are kept.

  Example 5-11 shows a PAM module pam_auth.c that gives two challenges and validate the responses.

  Compile pam_auth.c to create pam_auth.udr and copy pam_auth.udr under the /lib/security directory as follows:

  ```
  cc -c pam_auth.c
  ld -G -o pam_auth.udr pam_auth.o
  ```

  Change its permissions to 755 using the root ID:

  ```
  cp pam_auth.udr /lib/security
  chmod 755 /lib/security/pam_auth.udr
  ```

  The permission should look like the following:

  ```
  -rwxr-xr-x   1 root     other      8128 Oct 11 17:17 /lib/security/pam_auth.udr
  ```

Example 5-11 PAM module pam_auth.c

```
/* Start of pam_auth.c */
#include <stdio.h>
#include <string.h>
/* Include pam header file */
#ifdef WIN32
#include <pam_appl.h>
#else
#include <security/pam_appl.h>
#endif

#ifdef WIN32
```

```
__declspec(dllexport) int
#else
int
#endif
/**********************************************************************
 *  pam_sm_authenticate implements functionality of pam_authenticate.
 *  It verifies  the identity  of the current user
**********************************************************************/
pam_sm_authenticate(pam_handle_t *pamh, int flags,
    int argc, const char *argv[])
{
   struct pam_conv *conv;
   struct pam_message msg[3];
   struct pam_message *msgp;
   struct pam_response *resp;
   const char *pam_user;
   char *answer;
   int pam_err, retry;char  *prompt[3],*vanswer[3];
   pam_err = pam_get_item(pamh, PAM_USER, (void **)&pam_user); /* pamh is
                                                            PAM handle */
   if (pam_err != PAM_SUCCESS)
      return (PAM_SYSTEM_ERR);
   prompt[0] = (char *) strdup("Security Code(Rxcode):");
   prompt[1] = (char *) strdup("Security Number(9991024):");
   prompt[2] = (char *) strdup("PAM Text Info");
   vanswer[0] = (char *) strdup("Rxcode");
   vanswer[1] = (char *) strdup("9991024");
   vanswer[2] = (char *) NULL;
   pam_err = pam_get_item(pamh, PAM_CONV, (void **)&conv);
   if (pam_err != PAM_SUCCESS)
      return (PAM_SYSTEM_ERR);
   msg[0].msg_style = PAM_PROMPT_ECHO_OFF;
   msg[0].msg = prompt[0];msg[1].msg_style = PAM_PROMPT_ECHO_ON;
   msg[1].msg = prompt[1];msg[2].msg_style = PAM_TEXT_INFO;
   msg[2].msg = prompt[2];
   for (retry = 0; retry < 3; retry++)
   {
      msgp = &msg[retry];
      resp = NULL;
      pam_err = (*conv->conv)(1, &msgp, &resp, conv->appdata_ptr);
      if (pam_err == PAM_SUCCESS)
      {
/* No response needed for text info and error msg*/
         if ((msg[retry].msg_style == PAM_TEXT_INFO)
||(msg[retry].msg_style == PAM_ERROR_MSG))
         {
            continue;
         }
         answer = resp->resp;
         if (!answer)  /* Authorization error if response is blank */
         {
            pam_err = PAM_AUTH_ERR;break;
         }
/* Authorization error if response doesn't match the required answer */
         if (strcmp(answer, vanswer[retry]))
         {
            pam_err = PAM_AUTH_ERR;break;
         }
      }
   }

   free(prompt[0]);
   free(prompt[1]);
   free(prompt[2]);free(vanswer[0]);
   free(vanswer[1]);return (pam_err);
}

/**********************************************************************
 *  pam_sm_acct_mgmt implements functionality of pam_acct_mgmt.
 *  This checks for password aging and access-hour restrictions
 *********************************************************************/

#ifdef WIN32
__declspec(dllexport) int
#else
int
#endif
pam_sm_acct_mgmt(pam_handle_t *pamh, int flags,
    int argc, const char *argv[])
{
   const char *pam_user;
   int pam_err;

   pam_err = pam_get_item(pamh, PAM_USER, (void **)&pam_user);
   if (pam_err != PAM_SUCCESS)
      return (PAM_SYSTEM_ERR);
   else
      return (PAM_SUCCESS);
}

#ifdef WIN32
__declspec(dllexport) int
#else
int
#endif
/**********************************************************************
 *  pam_sm_open_session implements functionality of pam_open_session.
 *  It is called to initiate session management
```

```
**********************************************************************/
pam_sm_open_session(pam_handle_t *pamh, int flags,
    int argc, const char *argv[])
{
   return (PAM_SUCCESS);
}
#ifdef WIN32
__declspec(dllexport) int
#else
int
#endif
/**********************************************************************
 *  pam_sm_close_session implements functionality of pam_open_session.
 *  It is invoked when session is terminated
**********************************************************************/
pam_sm_close_session(pam_handle_t *pamh, int flags,
    int argc, const char *argv[])
{
   return (PAM_SUCCESS);
}
/* End of pam_auth.c */
```

Refer to Appendix A, "Audit event mnemonics" on page 269, for other PAM APIs and macros.

10. Register the callback function.

    IDS client applications that support PAM are ODBC, JDBC, and ESQL/C. Ensure that your application has the callback function registered for the challenge-response mode.

ESQL/C application connecting to PAM configured server

The ESQL/C provides an API ifx_pam_callback to register a callback function. Application callback function registration syntax is as follows:

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

Example 5-12 Illustrates the use of a callback API provided by ESQL/C and the flow of the client application. Compile pam_basic.ec to get pam_basic.exe:

```
esql -I$INFORMIXDIR/incl/esql pam_basic.ec -o pam_basic.exe
```

Run pam_basic.exe as follows:

```
./pam_basic.exe
```

Example 5-12 ESQL/C client application for connecting to the PAM configured server

```
/* Start of pam_basic.ec */
/* system include */
#include <stdio.h>
#include <stdlib.h>

/* ifx and pam include */
#include <ifxtypes.h>

#ifdef WIN32
#include <pam_appl.h>
#else
#include <security/pam_appl.h>
#endif

/* callback function to exchange challenges and responses */
int ifxcallback(char *challenge, char *response, int msg_style);

int error_flag=0;

int
main(int argc, char *argv[])
{
   $char   TESTNAME[]="pam_basic";
   char    *ServerName=(char *)NULL,*PamMode=(char *)NULL;
   $char   DBVAR[80]="",passwd[20]="";
   $int    retval=0;

   printf("Verify authentication when PAM is set to...\n");
   ServerName="ol_myserver";
   sprintf(DBVAR, "@%s", ServerName);

   printf("connect statement before registering should return error
-1809\n");
   $ connect to :DBVAR;
   ver_sqlcode("connect to ol_myserver",-1809);
/* register a callback function */
```

```
   printf("Register ifxcallback function\n");
   retval = ifx_pam_callback(ifxcallback);
   if(retval)
   {
      printf("Register ifxcallback function - failed\n");
      exit(1);
   }
   else
   {
      printf("Register ifxcallback function - passed\n");
   }

   printf("connecting to database, server should raise challenge\n");
   $ create database :TESTNAME;
   ver_sqlcode("create database pam_basic",0);
   $ disconnect current;
   ver_sqlcode("disconnect current",0);

   printf("connecting to database, server should raise challenge\n");
   $ database :TESTNAME;
   ver_sqlcode("connect database",0);

   $ create table pamtab (col1 int, col2 int);
   ver_sqlcode("create table pamtab",0);

   $ insert into pamtab values (1, 1);
   ver_sqlcode("insert into pamtab",0);
   $ insert into pamtab values (2, 2);
   ver_sqlcode("insert into pamtab",0);
   $ insert into pamtab values (3, 3);
   ver_sqlcode("insert into pamtab",0);

   $ close database;
   ver_sqlcode("close database pam_basic",0);
   $ drop database :TESTNAME;
   ver_sqlcode("drop database pam_basic",0);
   $ disconnect all;
   ver_sqlcode("disconnect all",0);
      printf("End of Demo\n");
   exit(0);
}

/*********************************************************************
***************
```

```
Register a callback function with CSDK. When CSDK receive a challenge
from the server, it is forwarded to the application via this callback
function.

***********************************************************************
**************/
int ifxcallback(char *challenge, char *response, int msg_style)
{
   switch (msg_style)
   {
   case PAM_PROMPT_ECHO_OFF:
   case PAM_PROMPT_ECHO_ON :
      printf("%s", challenge);
      scanf("%s", response);
      break;
   case PAM_ERROR_MSG:
   case PAM_TEXT_INFO:
   default:
      printf("%s\n", challenge);
      challenge = (char *)NULL;
   }
   return 0;
}
/**********************************************************************
***
ver_sqlcode function checks the return code of query and reports passed
or failed status.
***********************************************************************
**/
ver_sqlcode(char *stmnt,int ret_code)
{
   if (sqlca.sqlcode != ret_code)
   {
      printf("%s- failed with error code: [%d]\n",stmnt,sqlca.sqlcode);
      exit(1);
   }
      printf("%s- passed return code : [%d]\n",stmnt,sqlca.sqlcode);
}

/*End of pam_basic.ec */
```

Now we are done with PAM setup, PAM authentication module development and deployment, and client application development. Execution of the client application should look as shown in Example 5-13.

Example 5-13 Connecting to a PAM configured server from application

```
Verify authentication when PAM is set to...
connect statement before registering should return error -1809
connect to ol_myserver - passed return code : [-1809]
Register ifxcallback function
Register ifxcallback function - passed
connecting to database, server should raise challenge
Security Code(Rxcode):Rxcode
Security Number(9991024):9991024
PAM Text Info
create database pam_basic - passed return code : [0]
disconnect current - passed return code : [0]
connecting to database, server should raise challenge
Security Code(Rxcode):Rxcode
ecurity Number(9991024):9991024
PAM Text Info
connect database - passed return code : [0]
create table pamtab - passed return code : [0]
insert into pamtab - passed return code : [0]
insert into pamtab - passed return code : [0]
insert into pamtab - passed return code : [0]
close database pam_basic - passed return code : [0]
drop database pam_basic - passed return code : [0]
disconnect all - passed return code : [0]
End of Demo
```

JDBC application connecting to PAM configured server

A JDBC application communicating with PAM configured server must implement the com.informix.jdbc.IfmxPAM interface. To do so, it should have *implements IfmxPAM* in the class declaration. For example:

```
public class class_name implements IfmxPAM
```

Secondly, your application has to inform the JDBC driver what class has implemented IfmxPAM interface. There are two ways to do so:

- If you are using the DriverManager.getConnection(URL) method to connect to server, add the key-value IFX_PAM_CLASS=your_class_name.
- If you are using the DataSource.getConnection() method to connect the server, add the property IFX_PAM_CLASS with the value your_class_name in the properties list before connecting to the server.

Example 5-14 shows a JDBC class that implements the IfmxPAM interface and is similar to the callback function.

Example 5-14 The InfxPAMModule.java implementing IfmxPAM interface

```
   import com.informix.jdbc.IfmxPAM;
   import com.informix.jdbc.IfxPAMChallenge;
   import com.informix.jdbc.IfxPAMResponse;

   public class InfxPAMModule implements IfmxPAM
   {
      public IfxPAMResponse IfxPAM(IfxPAMChallenge challengeMessage)
      {
         IfxPAMResponse r = new IfxPAMResponse();
         String cm = challengeMessage.getChallenge();
         if (cm==null )
         {
            r.setResponse("UNKNOWN RESPONSE MESSAGE!");
         }
         else
         {
            if (challengeMessage.getChallengeType() ==
IfxPAMChallenge.PAM_PROMPT_ECHO_OFF)
               r.setResponse("Rxcode");
            else if (challengeMessage.getChallengeType() ==
IfxPAMChallenge.PAM_PROMPT_ECHO_ON)
               r.setResponse("9991024");
         }
      return r;
   }
```

Example 5-15 shows a JDBC application IDSConnectTest.java trying to connect the IDS instance configured to use PAM.

Example 5-15 A JDBC application program

```
import java.sql.*;
public class IDSConnectTest
{
   Connection conn = null;
   public IDSConnectTest()
   {
      super();
      try
      {
         Class.forName("com.informix.jdbc.IfxDriver");
      }
      catch (ClassNotFoundException e)
      {
         e.printStackTrace();
      }
      String url  =
"jdbc:informix-sqli://mercury.lenexa.ibm.com:9226/pam_basic:IN
FORMIXSERVER=ol_myserver;IFX_PAM_CLASS=InfxPAMModule";

      String user = "user";
      String password = "passwd";
      try
      {
         conn = DriverManager.getConnection(url,user,password);
         system.out.println("Connected.");
      }
      catch (SQLException e)
      {
         System.out.println("Problem connecting: " + e.getMessage() + "
" + e.getErrorCode());
         e.printStackTrace();
      }
   }

   public static void main(String argc[])
   {
      IDSConnectTest test;
      test = new IDSConnectTest();
   }
}
```

ODBC application connecting to PAM configured server

ODBC uses SQLSetConnectAttr API to register the callback function. Parameter attributes are passed back to the callback function exactly as they are specified to the driver.

Table 5-2 lists the attributes of the IDS-specific extensions to the ODBC standard.

Table 5-2 IDS-specific extensions to the ODBC standard for using PAM

Parameter (SQL_INFX_ATTR_PAM)	**Type**	**Description**
_FUNCTION	void *	A pointer to the callback function.
_RESPONSE_BUF	void *	A generic pointer to the buffer containing the response to a challenge.
_RESPONSE_LEN	int	Length of response buffer in bytes.
_RESPONSE_LEN_PTR	int *	The address that stores the number of bytes in the response.
_CHALLENGE_BUF	void *	A generic pointer to a buffer containing the authentication challenge. The driver stores any challenge received from the server into this buffer. If the buffer is not large enough to contain the challenge, the challenge is truncated. The callback function can detect this by comparing the buffer length with the number of bytes in the challenge. It is up to the application developer to detect this situation and handle it correctly.
CHALLENGE_BUF_LEN	int	The length of the challenge buffer in bytes.
CHALLENGE_LEN_PTR	int *	The address that stores the number of bytes in the challenge.

Example 5-16 shows the code that you should have in your ODBC application to register a callback function. A callback function can be same as written in ESQL/C code.

Example 5-16 Code to register callback function pointed to by function pointer ifxcallback

```
rc = SQLSetConnectAttr(hdbc, SQL_INFX_ATTR_PAM_FUNCTION, (SQLPOINTER)
ifxcallback, SQL_IS_POINTER);
rc = SQLSetConnectAttr(hdbc, SQL_INFX_ATTR_PAM_RESPONSE_BUF,
pamResponse, SQL_IS_POINTER);
```

```
rc = SQLSetConnectAttr(hdbc, SQL_INFX_ATTR_PAM_RESPONSE_BUF_LEN,
(SQLPOINTER) pamRespSize, SQL_IS_INTEGER);
rc = SQLSetConnectAttr(hdbc, SQL_INFX_ATTR_PAM_RESPONSE_LEN_PTR,
&pamRespRetLen, SQL_IS_POINTER);
rc = SQLSetConnectAttr(hdbc, SQL_INFX_ATTR_PAM_CHALLENGE_BUF,
(SQLPOINTER) pamChallenge, SQL_IS_POINTER);
rc = SQLSetConnectAttr(hdbc, SQL_INFX_ATTR_PAM_CHALLENGE_BUF_LEN,
(SQLPOINTER) pamChalSize, SQL_IS_INTEGER);
rc = SQLSetConnectAttr(hdbc, SQL_INFX_ATTR_PAM_CHALLENGE_LEN_PTR,
&pamChalRetLen, SQL_IS_POINTER);
```

5.2.3 Authentication mechanism in distributed query environment

IDS supports communication between two or more servers on the same machine, as well as over the network. Distributed queries, enterprise replication, high data availability replication, and shared disk are few of the features that uses this functionality.

Distributed queries

An IDS server allows you to query more than one database of different versions of IDS. Different databases can be on the same host or on remote hosts. This feature is called Distributed Queries (ISTAR). Figure 5-5 Illustrates the basic concept of the distributed query.

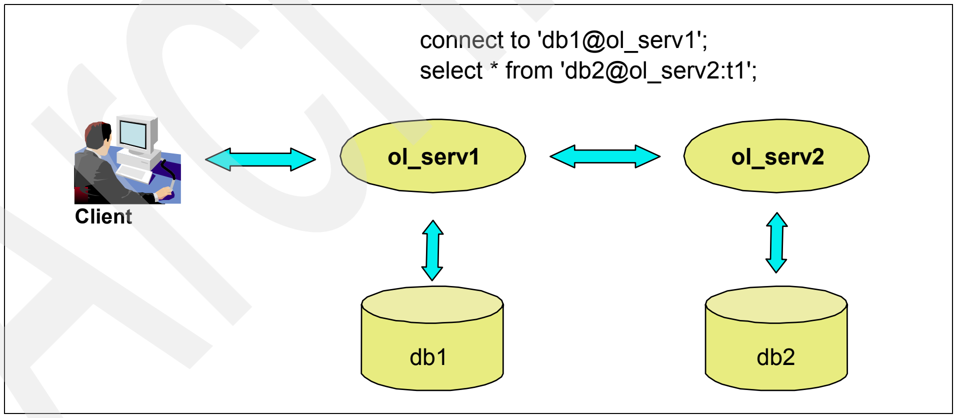


Figure 5-5 The distributed query

Remote connections are authenticated using an r-command system devised by Berkeley System Division (BSD). Whenever there is request for a connection from the remote host, a daemon running on the local system checks the /etc/hosts.equiv file or .rhosts file in user’s home directory. If the entry of the trusted host is present, the requesting remote host is treated as the trusted host and a database connection is allowed.

If PAM is set for the servers, distributed queries do not work. A challenge raised by the remote server cannot be responded to, as the challenge is raised on the local server and not on the client. In such a situation authentication on the remote servers must be done within the database or you can connect to the database server alias, which is not be configured for PAM.

Example 5-17 demonstrates the distributed query execution when one of the servers is PAM enabled. There are two hosts, vulcan.lenexa.ibm.com and mercury.lenexa.ibm.com, that have the IDS instances ol_serv1 and ol_serv2 running. Instance ol_serv2 on mercury is PAM enabled. Create database db1 on vulcan and db2 on mercury.

Example 5-17 Distributed query execution when one of the servers is PAM enabled

```
(vulcan$) cat $INFORMIXSQLHOSTS
   ol_myserver onsoctcp vulcan tcp_serv s=4, pam_serv=(pam_chal),
   pamauth=(challenge)
   ol_serv2      onsoctcp mercury tcpservice2
   ol_serv2_al   onsoctcp mercury tcpservice3

(mercury$) cat $INFORMIXSQLHOSTS
   ol_myserver ontlitcp vulcan tcpservice1
   ol_serv2    ontlitcp mercury tcpservice2
   ol_serv2_al ontlitcp mercury tcpservice3

(vulcan$)dbaccess - -
> connect to ‘@ol_myserv’;
Security Code(Rxcode):
Security Number(9991024):: 9991024
PAM Text Info:
Connected.

> create database db1;

Database created.

> create table t1 (col1 int, col2 char(20));

Table created.

> insert into t1 values (1,"redbook");

1 row(s) inserted.
```

```
(mercury$)dbaccess - -
>connect to ‘@ol_serv2’;

Connected.

>create database db2;

Database created.

> select * from db1@ol_myserv1:t1;
950: User user1@mercury.lenexa.ibm.com is not known on the database
server.

  111: ISAM error:  no record found.
Error in line 1
Near character position 32
```

IDS provides a database called sysuser to handle the authentication within the database that is used in distributed queries. To have authentication within the database, the system administrator has to enter authorized users in the sysauth table in the sysuser database.

Table 5-3 shows the sysauth table schema.

Table 5-3 The sysauth table schema

Column	Type	Description
username	char(32)	User ID of authorized user
groupname	char(32)	Group ID of authorized user
servers	char(128)	IDS server name through which connection should be allowed
hosts	char(128)	Host name from which a connection should be allowed

For the example discussed in Example 5-17 on page 189, if you want to allow user1 to perform distributed queries from vulcan.lenexa.ibm.com through ol_serv1, you need to add the information into the sysauth table in the sysuser database on machine2, as shown in Example 5-18. Information about the server, user, group, and hosts is inserted in the sysuser:sysauth table on machine2.

Example 5-18 insert a row in sysauth table in sysuser database

```
INSERT INTO sysauth
VALUES ("user1","group1","ol_serv2",”mercury.lenexa.ibm.com");
```

After adding above entry, a SELECT statement is successfully executed.

It is not always possible to insert details of each and every user into the sysuser:sysauth table. Hence, to circumvent this situation, you can use DBSERVERALIAS in distributed queries and can continue using PAM for client server communication. This does not require any entry in the sysuser:sysauth table.

Example 5-19 shows a distributed query using DBSERVERALIAS instead of DBSERVERNAME, to avoid PAM authentication mechanism.

Example 5-19 A distributed query using DBSERVERALIAS

```
(mercury$)dbaccess - -
> connect to ‘db2@ol_serv1’;
Connected.
>select * from ‘db1@ol_myserv:t1;
950: User user1@vulcan.lenexa.ibm.com is not known on the database
server.
Error in line 1
Near character position 31
>select * from ‘db1@ol_serv2_al:t1;

col1 col2

          1 redbook

1 row(s) retrieved.
```

5.3 Network encryption

Authentication ensures the authorized access to the data. However, in network-centric business models, it is important to ensure the confidentiality and integrity of data flowing on a network. Hacking and tampering are the major threats to the data flowing in the network. In client-server communication, data will be the input provided by the client to the server, and the data provided by the server to the client. Input provided by the client mostly comprises of the user ID, user password, and data request (SQL statement). Data encryption is the well-accepted mechanism to maintain the security of such data. Encryption is the transformation of data from a commonly understandable form to a non-understandable form using some algorithm. One cannot transform data back to an intelligent form unless she knows the encryption key.

Encryption keys are managed in two ways:

- Symmetric cryptography - One key is used for both encryption and decryption.
- Public key cryptography - A key used for encryption is different from the key used for decryption.

5.3.1 Encryption mechanism in IDS

IDS supports a symmetric cryptography technique for encryption.

In client-server communication, both client and server first negotiate on the secret key using the Diffie-Hallman (DH) algorithm. A DH algorithm allows it to negotiate a key over the network secretly. Once the secret key is decided, the client and server can start data transfer in encrypted form over the network.

Figure 5-6 illustrates that the server and client negotiate a secret key over a network using the DH algorithm before starting actual communication. Once the key is decided, client and server starts encryption and conveys the encrypted data to each other over the network.

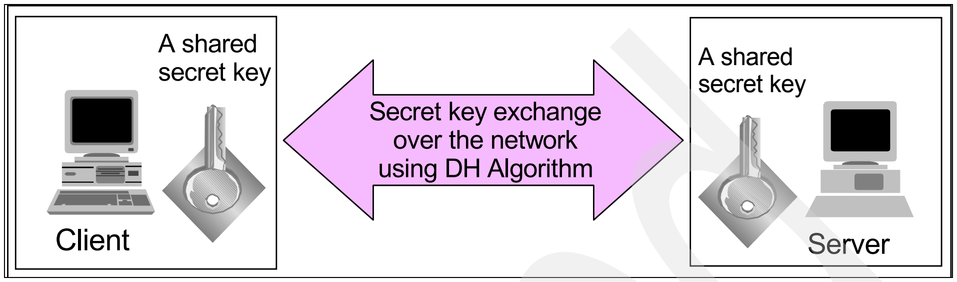


Figure 5-6 Server and client negotiate a secret key over the network using DH algorithm

A data packet to be sent over the network is encrypted using different industry standard encryption logic, usually called ciphers, supported by IDS.

With the help of the shared secret key and cipher, you can start encryption to ensure the secrecy or confidentiality of data. Encryption can ensure that the data transferred over the network is not read or used by anybody. But it does not ensure that the data sent is the data received. It may happen that the data is tampered with in between while transferred through an unsecured network. Hence, it is equally important to ensure the data integrity along with the data confidentiality.

IDS ensures data integrity by using Message Authentication Code (MAC) mechanism. Before sending a data packet, IDS calculates the MAC value and sends it along with the data packet. It is similar to message length checksum. For security, a key used for MAC calculation is different from the shared secret key negotiated between the server and the client.

At the receiving end, the encrypted data is decrypted, the MAC value is recalculated using the original data, and it is compared with the MAC value sent along with the data. Matching of the MAC value came with the data packet, and the MAC value calculated at the receiving end confirms the data integrity.

Figure 5-7 shows the encryption and decryption process in client-server communication.

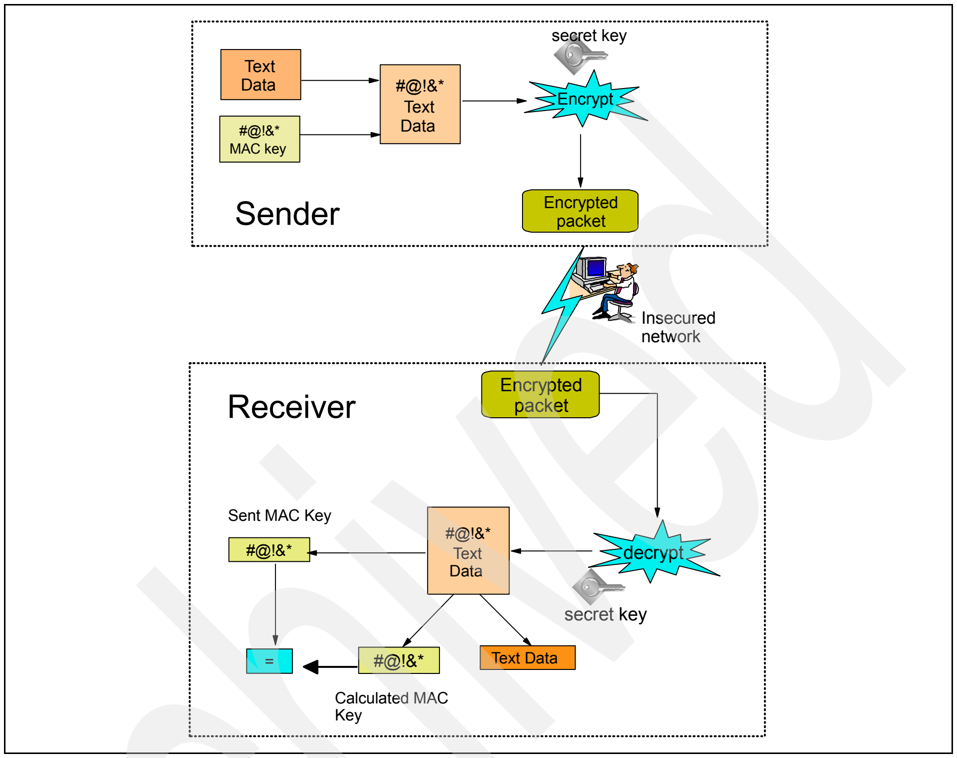


Figure 5-7 The encryption and decryption mechanism.

Few important things that you should know about encryption in IDS are:

- You can fine-tune various parameters involved in the encryption mechanism.
- The shared secret key is not the same for the entire time that the client server session is active.
- The shared secret key is renegotiated at regular intervals, which can be specified by the user.
- You can determine which encryption algorithms (ciphers) you want to use.
- IDS selects an algorithm from the specified cipher list periodically. You can control the periodicity of the cipher switch as well.
- IDS lets you control the MAC generation, depending on the size of the message packet being sent.
- You may also turn off the MAC authentication completely.

Encryption technology is integrated with IDS as a pluggable communication supports module (CSM). IDS provides the network encryption communication support module (ENCCSM), which enables you to encrypt complete client-server communication and the simple password communication support module (SPWDCSM), which enables you to encrypt the user password. CSM is a pluggable component of the communication support system (CSS) built into IDS. It facilitates users to write their own encryption logic for encryption using CSM APIs. IDS has its own proprietary CSS APIs, whereas CSM APIs are public.

In Figure 5-7 on page 194, the encryption and decryption of the packet are performed by the CSM module.

Table 5-4 lists the location of the encryption modules supported by IDS.

Table 5-4 Location of encryption modules supported by IDS

Encryption	OS	Server shared library	Client shared library
ENCCSM	UNIX	$INFORMIXDIR/lib/csm/iencs11a.so	$INFORMIXIDR/lib/client/csm/iencs09a.so
	Windows	INFORMIXDIR/bin/iencs11a.dll	INFORMIXDIR/lib/client/csm/iencs09a.dll
SPWDCSM	UNIX	$INFORMIXDIR/lib/csm/libixspw.so	$INFORMIXDIR/lib/client/csm/libixspw.so
	Windows	INFORMIXDIR/lib/csm/libixspw.dll	INFORMIXDIR/lib/client/csm/libixspw.dll

For JDBC clients, java class com.informix.jdbc.Crypto is the Java implementation of encryption CSM. This class is packaged along with JDBC driver jar ifxjdbc.jar.

5.3.2 Configuring IDS to use encryption

In order to configure IDS to use ENCCSM and SPWDCSM, you need to mention the correct objects at the correct places. The configuration procedure is:

1. Define an encryption CSM.

   The objects involved in this step are:

   – Encryption mechanism (ENCCSM/SPWDCSM)
   – Encryption logic, often called ciphers
   – CSM shared library
   – MAC key
   – Switching time between ciphers

2. Associate CMS to the IDS instance.

Define an encryption CSM

A configuration file concsm.cfg is used to define details of the encryption CSM. It is a plain text file and can be edited using any editor. By default, this configuration file should be under the $INFORMIXDIR/etc directory. If you want to keep it at different location, use the INFORMIXCONCSMCFG environment variable and mention the absolute path of the configuration file.

The syntax of the entry in concsm.cfg is as follows:

```
ENCCSMNAME("path","cipher[options],mac[ options], switch[ options]",
w"")
```

Where the options are:

ENCCSMNAME	Name of the CSM module
path	Absolute path of the CSM shared library
cipher	Name of the cipher that you want to use for encryption logic
mac	Absolute path of the MAC key file
switch	Time interval for switching between ciphers if multiple ciphers are defined

Example 5-20 shows how the entry for ENCCSM in the concsm.cfg file should look. iencs11a.so is a CSM shared library. The ciphers used are des and ede with mode ecb. The level of MAC is set to medium, and the location of the mackey file is /usr/informix/etc/mackey.dat. Ciphers are switched with interval of two seconds.

Example 5-20 ENCCSM entry in concsm.cfg file

```
ENCCSM("/usr/informix/lib/csm/iencs11a.so","cipher[des:ecb,ede:ecb],mac
[levels:<medium>,files:</usr/informix/etc/mackey.dat>],switch[cipher:2,
key:1]","")
```

For SPWDCSM, the syntax is as follows:

```
CSMNAME("path","global option","password mode")
```

Where the options are:

path	Absolute path of the shared library
Global option	Not supported currently, so should be null
Password mode	p=0 password not mandatory; p=1 password mandatory for authentication

Example 5-21 shows how the entry for SPWDCSM in the concsm.cfg file should look. libixspw.so is a CSM library and p=1 mandating the password verification is needed.

Example 5-21 SPWDCSM entry in concsm.cfg file

```
SPWDCSM("/usr/informix/lib/csm/libixspw.so","","p=1")
```

You can define multiple ciphers or all ciphers. For using all ciphers, entry in concsm.cfg should look like cipher[all,mac...]. You can use the allbut option to eliminate one or a group of ciphers and consider remaining all. If you define cipher[allbut<des>...] then except des ciphers all ciphers will be used. But if you define cipher[allbut<de>...], then, along with des, ede and desx ciphers also aree eliminated.

To ensure higher security, we recommend that you use all the ciphers and keep renegotiating the keys. IDS uses different ciphers with regular intervals specified in the switch. Table 5-5 lists the ciphers supported by IDS.

Table 5-5 IDS supports des, ede and desx ciphers

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

Table 5-6 lists AES encryption ciphers supported by IDS.

Table 5-6 AES encryption ciphers supported by ID

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

Table 5-7 Modes supported by IDS

Mode	Explanation
ecb	Electronic code book
cbc	Cipher block chaining

Mode	Explanation
cfb	Cipher feedback
ofb	Output feedback

Associating CSM to IDS instance

Once concsm.cfg is set, you need to associate a CSM to IDS. For associating CSM to IDS, update the sqlhosts file.

If using ENCCSM, the entry in sqlhosts should look like:

```
ol_myserv onsoctcp machine1 serv_name csm=(ENCCSM)
```

If using SPWDCSM, the entry in sqlhosts should look like:

```
ol_myserv onsoctcp machine1 serv_name csm=(SPWDCSM)
```

Verifying the setup

Though you have done the setup correctly, you should ensure that it works properly. It should not affect the client application that is running remotely.

For verifying your setup, bring up the engine as **`oninit -vy`**. You should see the following message on the panel:

```
Initialization of Encryption...succeeded.
```

Note that a csm vp that handles the csm requests is not started during engine start up. It is started after the first connect request only. Hence, even if your concsm.cfg is wrong, you will not see any error in engine start up. Secondly, it is also necessary to know that the concsm.cfg file is loaded only once during startup of csm vp. Hence, if you do any changes in concsm.cfg after csm cp started, it will not be taken into effect. To have the effects of the changes, you need to restart the engine (oninit -v).

To ensure that the setup is correct, make an attempt to connect to the server using the dbaccess utility. If you have configured everything correctly, you should be able to connect to the server successfully.

However, if there is any error while connecting, you should get the details of error in your online log file. For example, if you see error `5006: CSM: invalid global options` while connecting, check your online log file for details. It should have something like `listener-thread: err = -5006: oserr = 0: errstr = Unknown cipher/mode requested: CSM: invalid global options`, which directs you to correct the cipher or cipher mode details in your concsm.cfg file. In order for the changes to take effect, you need to restart the engine. Otherwise, you see an error like:

```
listener-thread: err = -5006: oserr = 0: errstr = : CSM: invalid global
options
```

5.3.3 Configuring IDS client to use encryption

In order to connect to the IDS server configured to use encryption, a client should also be configured for it. A client that is not configured to use encryption cannot connect to the server port that is configured to use encryption, and vice versa.

The steps to configure the client to use encryption are the same as for the server except the CSM shared library path defined in concsm.cfg file. Refer to Table 5-4 on page 195 to check the path of client library.

Example 5-22 shows the configuration of sqlhosts and concsm.cfg on the IDS client side.

Example 5-22 The setting of sqlhosts and concsm.cfg on IDS client side

```
sqlhosts:
ol_myserv onsoctcp vulcan serv_name csm=(ENCCSM)

concsm.cfg:
ENCCSM("/usr/informix/lib/client/csm/iencs11a.so","cipher[des:ecb,ede:e
cb],mac[levels:<medium>,files:</usr/informix/etc/mackey.dat>],switch[ci
pher:2,key:1]","")

SPWDCSM(“/usr/informix/lib/client/csm/libixspw.so”,””,”p=1”)
```

As far as JDBC clients are concerned, a new data source property CSM has been added to make a connection using encryption CSM. Here is a Java code example that sets the CSM property:

```
connProperties.put("CSM","classname=com.informix.jdbc.Crypto,cipher[all
],mac[levels:<high>,files:</usr/informix/etc/mackey.dat>],switch[cipher
:2,key:1]");
```

The Java encryption CSM is available with JDBC 2.21.JC5 and later.

When an encrypted connection is being established, the client and server exchange the list of values for each of CSM attribute. Then they determine a negotiated list of values for the CSM attribute. The negotiated values are used during the session. For the cipher attribute, the negotiated list is the list of ciphers common to server and client encryption CSM. Ciphers from the negotiated list are used during the session. Unlike the cipher, which is changed periodically during the session, only one Mac key is used for the entire life of the session. If the client and server CSMs have more than one common MAC key, the one that is generated most recently is selected. The Mac level to be used for the session is the highest common denominator value between the client and the server Mac level lists.

The connection attempt fails if the server is not able to decide upon a negotiated value for any of the encryption attributes.

Table 5-8 describes the negotiation of the encryption values between the server and the client before establishing the connection. The table assumes that encryption parameters other than the ones listed in the table are set correctly.

Table 5-8 Negotiation of encryption values

Parameter	Server value	Client value	Connection	Explanation
cipher	allbut(bf)	bf-1:cbc,bf-2:cbc	NO	No common cipher
MAC Level	low, medium	high	NO	No common denominator found
MAC Key file	/usr/informix/etc/mackey.dat,/usr/informix/etc/macvai.dat	/usr/informix/macvi.dat	Yes	MAC key from /usr/informix/macvai.dat used

In summary, the communication between the server and client when the server is configured to use encryption happens as follows:

- Between client and server
  - Negotiation between client and server for values of CSM attribute
  - Negotiation between client and server for secret session key
- Client side
  - Data encryption at client side using session key and MAC key
  - Encrypted data packet transfer over the network
- Server side
  - Decryption of data packet
  - Calculation of MAC key value

- Comparing calculated MAC key value with value came from client
- Database access
- Send response in encrypted form

Figure 5-8 is a pictorial representation of the steps involved in client-server communication when the server is configured to use encryption.

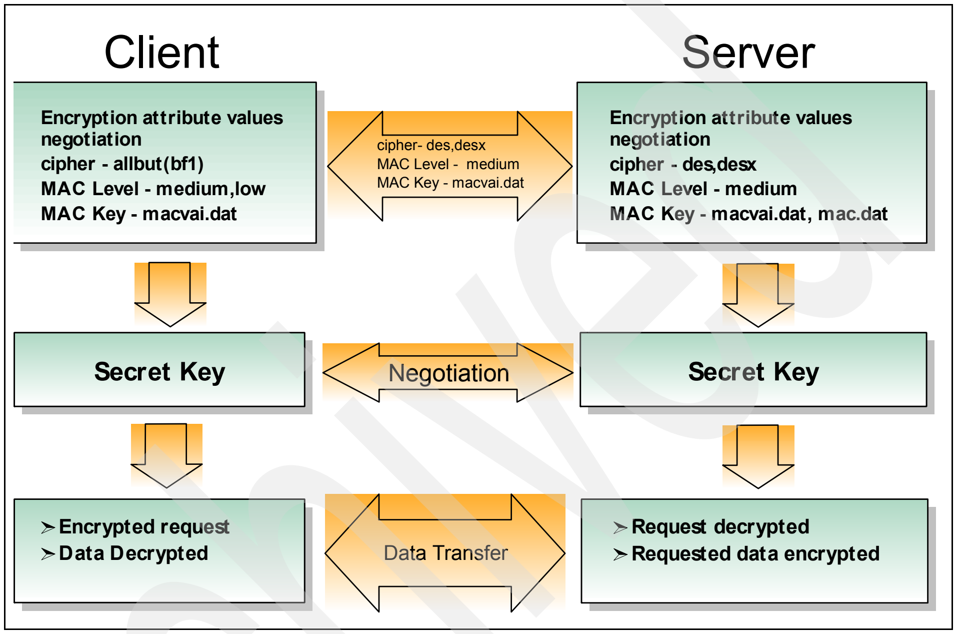


Figure 5-8 Overview of client-server communication when configured to use encryption

Note: You can use only one encryption mechanism at a time. You can use PAM in conjunction with any of the IDS-supported encryption mechanisms.

5.4 Secure local connection to a host

Securing a local connection to a host is an authentication mechanism built into IDS and can be enabled by setting configuration parameter SECURITY_LOCALCONNECTION. This parameter can have any of the three values, 0, 1, or 2:

- 0 - No security checking occurs.
- 1 - The dynamic server checks 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.

- 2 - Same as 1, plus the dynamic server retrieves the peer port number from the network API and verifies that the connection is coming from the client program. You can only specify 2 if your system has SOCTCP or IPCSTR network protocols.

By setting this parameter you can check that the ID of the user running an application is same as the ID of the user who is trying to connect to the database.

5.5 Limiting denial-of-service flood attacks

IDS has two types of threads to the server-client request: a listener thread that listens to the client request and forks a new thread for authentication. An authentication thread authenticates the request and allows or denies access to the database. For better connection performance you should have a moderate number of authentication threads, which ensures that the maximum amount of requests are handled at a time and the server is not overloaded. Also, the incomplete connection requests should be cleaned up.

If you use an external connectivity tool or utility like Telnet to connect to the port reserved for IDS but do not send data, and the second session is trying to connect to the server through an application, a listener thread is blocked waiting for data from the Telnet session and holds back the connection request from an application. If during the waiting period multiple Telnet requests are launched in a loop, you can receive a flood attack on the connection resulting in poor connection performance.

IDS provides two configuration parameters to handle such a situation:

- The MAX_INCOMPLETE_CONNECTIONS configuration parameter limits the number of incomplete connections or authentication threads running at a time. If the requests are going beyond this number, a service is denied instead of being putting on hold. The default value is 1024.
- The LISTEN_TIMEOUT configuration parameter cleans up the incomplete requests waiting for the period more than the value set for this parameter. The default value is 10 seconds.

6

Server-server communication

The requirement of communication between database servers becomes more and more important. Today there is a strong need for connecting regionally based databases to a global company network. Otherwise, the use of the IDS high-availability features that strongly rely on server-to-server communication in order to minimize production system downtime and provide load balancing is the key factor providing 24/7 operations.

Increased requirements for connecting database servers together also require exchanging sensitive, secured-by-law or business-relevant data that has to be carefully protected from unattended access. We discuss how to achieve this in this chapter.

From the security viewpoint, there are three major areas in the server-server communication that we cover here:

- Server authentication
- Firewall access
- Encrypting the data exchange between the server over the network

6.1 Introduction

Exchanging data and authenticating users not only apply to a pure client server communication, they also apply to the IDS 11 features that require inter server communication. The major features depending on inter server communication are distributed queries (referred as ISTAR functionality in this chapter) and all major high availability (HA) features.

Security tasks for the inter server communications can be considered for the setup of a new server-to-server communication and for the protection of an existing infrastructure. In this chapter we discuss the security tasks concerned with these two areas:

- Communication setup
  - Set up authentication with a trusted user environment.
  - Allow communication between the source and the target server on separate networks secured by firewalls.
- Protecting your data in a running HA environment

  Avoid raw data transmission over the network by encrypting the entire data stream exchanged between the different IDS database servers.

You may notice that the security tasks for the communication setup can also be applied for the client server communication. However, there are some differences in the way that users should be authenticated when a database server opens the connection to a remote IDS instance.

Raw data transmission over the network in a client server architecture is another serious topic for the configuration of the backup and restore infrastructure of the IDS database server. For more information about how to encrypt the data stream for the backup solutions, refer to Chapter 7, "Security issues with backup and restore" on page 239**.**

6.1.1 Whom requires server-server communication

The ISTAR feature allows SQL statements to be executed between a local coordinating database server and a number of remote participating servers. The statements can be executed within a transaction or standalone. The databases involved in the ISTAR statements do not require logging, but they must have the same log mode. The SQL statements that support remote participants include SELECT, INSERT, UPDATE, and DELETES. Most of the time these require a data exchange between the involved database servers over the network.

Example 6-1 shows a few ISTART queries.

Example 6-1 Sample ISTAR statement

```
dbaccess << EOF
INSERT INTO stores_demo@lech10uc6w5soc:customer
   SELECT customer_num+1000, fname, lname, company, address1,city,
          state, zipcode, phone
   FROM customer;

SELECT *
FROM   stores_demo@lech10uc6w5shm:customer remote,
       customer local
WHERE  local.customer_num=remote.customer_num

UPDATE stores_demo@lech10uc6w5soc:customer
SET    fname="Smith"
WHERE  customer_num=110;
EOF
```

In the HA topologies provided by the IDS, most of the time the standby servers maintaining a copy or using the same dbspace set are attached to a primary database server. In order to coordinate server log replay, heartbeat functionality, and failover, communication is required between the database servers involved in the HA infrastructure. The amount of traffic over the network depends on the daily workload on the primary database server.

Figure 6-1 illustrates the communication flow between the server instances in a shared disk secondary server (SDS) environment, which is a simple HA solution. There are three secondary servers attached to the primary database sever. The servers share the dbspaces, and the log buffer exchange is not required. The only data exchange is the current log position (LSN or log sequence number) of the primary server and the secondary servers acknowledge their progress to the primary.

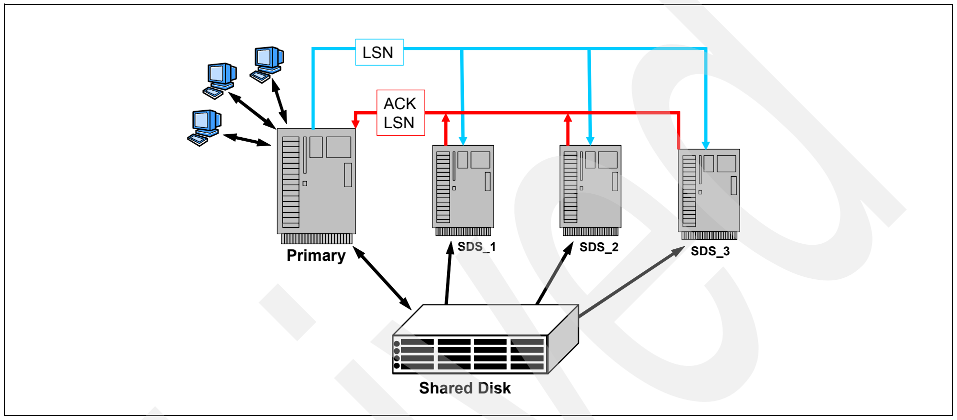


Figure 6-1 Message flow between the database servers in a SDS environment

6.1.2 Overview of high availability features provided by the IDS 11

The primary high availability usually refers to the ability to switch, in a disaster situation, from one production server to another standby server or parallel available server with at least the complete major data set required for running the production. Measuring this in seconds is one of the key features of each database server. For global Web-based business, world-wide integrated financial, and retail-based companies, a 24-hour a day availability requirement is crucial in running a production database environment. Integrated failure detection of cluster solutions in combination with database and application switch-over are necessary to fulfill the requirements of an undisturbed database operation. The additional requirement is the combination of different HA solutions in one infrastructure to provide a multi-stage insurance of 100% reliability. IDS 11 supports a set of reliable HA features that can be used in combination or by itself to fulfil these requirements.

HA solutions provided by the IDS can be classified in two different major basic technologies:

- Restore-based HA simple solutions such as continuous log restore
- Standby database server solutions based on log or log position shipping

All the solutions require server-server communication except the continuous log replay feature, which we introduce here for completeness.

The following sections introduce the HA solutions available in IDS 11 and how they can be differentiated.

Restore-based HA solution

This HA solution utilizes the continuous log replay:

- The standby database server was physically restored from a backup of the primary database server.
- The standby database server is in a perpetual log apply mode for the log backups taken on the primary database server, until the disaster happened and the database server has to be switched into the operational read write mode.
- This does not require server-server communication.

HA solution based on log or log position shipping

This type of HA solution includes the following:

- High Availability Data Replication (HDR)
  - HDR defines a pair of database servers with the same IDS server version situated on similar hardware and OS.
  - One of the database servers can handle read and write as the primary server. The other one can handle only read requests.
  - The primary ships the currently produced logs, in a time frame set on ONCONFIG parameters, to the secondary, which applies the logs to ensure data consistency between the pair.
  - Log shipping is related to the entire data set of the primary server.
  - In case of a failover, the secondary can, depending on the configuration, take over the role of the primary.
- Enterprise Replication (ER)
  - ER provides the ability to set up a heterogeneous environment with various IDS versions and OS.
  - In difference to HDR, ER has no restriction in the configuration of the paired IDS database server.
  - Different from other HA solutions, all database servers participating in the ER infrastructure are fully operational. This means that they are able to maintain their data with read and write operations.

  - The DBA is completely free to define which changes on which table have to be shipped to which target (in terms of projection and selection of the changed data in the table).
  - You are flexible in scheduling replication times for the table changes, defining conflict resolution strategies in case of parallel changes of the same data at the same time on a different database server.
  - The changes on tables participating in the ER that have to be distributed to other ER target servers are investigated by snooping the log on the source server.
- Remote Secondary Server (RSS)
  - RSS supports (differently from the HDR feature) multiple secondary servers attached to a defined primary server.
  - Each participating server maintains its own copy of the data, created by a restore from the primary database server.
  - RSS introduces a new communication layer in the database server called trusted server group multiplexer connections (SMX) to enable a multiplexed exchange of messages between the servers.
  - RSS does not have a sync mode in order to support low bandwidth networks and long-distance replications without blocking the primary server.
- Shared Disk Secondary Server (SDS)
  - SDS defines one primary server and a flexible number of read-only secondary servers, sharing the same subset of dbspaces either using the file system on the same machine or using a shared file system on separate machines.
  - Similar to RSS, SDS uses the new SMX communication protocol.
  - SDS is intended to be used for significant disk space reduction by using the shared-all approach and for load balancing across the participating and currently available database server.

For more information about the complete HA functionality provided by IDS 11, their requirements, and steps for setup and configuration, refer to *Informix Dynamic Server 11 Extending Availability and Replication*, SG24-7488.

6.2 Security tasks during setup of communication

There are different starting points for setting up the communication between IDS database servers, such as to link the standalone database servers without data consolidation, or to set up a new HA topology by attaching a new standby database server to an existing primary database server. No matter which starting point you have, there are two major security tasks that you have to consider in order to successfully set up a secure network around database server communication:

- Authenticate the database server with trusted connections.
- Allow the communication of the involved database servers on the existing firewall in your network.

In this section, we give you an overview of the steps each particular task contains. We also show you alternative ways to reach the goal with the focus of achieving the securest solution.

6.2.1 Setting up trusted server connections

For the authentication between a client and a database server, the client either has to provide a password at connection time or it has to be a trusted client. The authentication is done by the target database server. In the case of the communication between two database servers, both servers can adopt the roles of the client and the server. You may name the communication requestor the client and the communication target database server the server. Different from the client server communication, database servers do not specify passwords in the communication attempt. They require a trusted connection.

There is a difference in the requirements for which users' trusted connections are needed.

In all HA environments where an HA solution without ISTAR is used, only the informix user requires the trusted connection on the participating machines. In the ISTAR environments, the database server would allow the use of the feature to everyone who has appropriate database and table permissions on both sides. This results in the appropriate changes required in the OS-related access control file for each user who has the company-defined business needs.

UNIX

Trusted connections on UNIX-based systems are maintained either on a machine level or on a user level. The machine level is maintained by the entries in the /etc/hosts.equiv file. The user access control is maintained by the .rhosts file, which resides in the home directory of the target machine for each user who needs trusted access.

.rhosts file

In this discussion, we focus on the steps to set up the access between the machines for the HA environments. You have to enable especially the remote access for the informix user. In case you want to set up ISTAR environments and use .rhosts files for the access control, you have to apply similar changes to the user's .rhost file on the participating database machines.

The .rhosts file is located in the informix users home directory on the remote machine. The content of this file specifies which users from which machines are allowed to log into this machine using the informix account without specifying a password. Maintaining this file carelessly causes potential security vulnerabilities on this machine. To keep the security weakness to a minimum, only add to the file the IP addresses of the machines that really need to talk with the local machine as a target. Ideally in an HDR environment, on the secondary machine, specify only the primary database server, and on the primary system, specify only the secondary system. Similar requirements are in the SDS/RSS environment on the secondary machine. The file on the primary machine has to specify all potential secondary servers.

Example 6-2 illustrates the required setup for an HDR pair on two separate machines.

Example 6-2 Necessary settings in the sqlhosts and .rhosts files for an HDR environment

```
#sqlhosts file on both sides
hdr_prim_11 onsoctcp primary.ibm.com 23487
hdr_sec_11  onsoctcp secondary.ibm.com 23485

#informix users .rhosts file on the primary
secondary.ibm.com informix

#informix users .rhosts file on the secondary
primary.ibm.com informix
```

Do not specify only a plus sign (+) in this file, which allows the users from all remote machines to access this machine. We mean the informix user, which is the administrator account associated with all possible database permissions.

/etc/hosts.equiv

If you have a machine where all the users are to be trusted, you can set up the trusted connection at the machine level using the /etc/hosts.equiv file. This approach saves the overhead of adding all users the appropriate entries in the .rhosts file and maintaining them. To set up the trusted connection, add the source machine host name to the file /etc/hosts.equiv file. Pay closer attention when working on the content of this file. The security risk of having incorrectly specified entries in this file is much higher than in .rhosts. Check whether you really trust all the users from the remote host before adding the remote machine to this file. You also can exclude specific users in the hosts.equiv file.

Add the user to be excluded before the entry allowing access to the machine. Example 6-3 shows sample hosts.equiv content on both the primary and the secondary machine. Here we allow all the users except user1 from machine102 to access the primary machine. Similar entries have to be specified on the secondary.

Example 6-3 Excluding users from trust in the /etc/hosts.equiv file

```
#sqlhosts file on both sides
hdr_prim_11 onsoctcp primary.ibm.com 23487
hdr_sec_11  onsoctcp secondary.ibm.com 23485

#informix users /etc/hosts.equiv file on the primary
machine100.ibm.com informix
machine101.ibm.com informix
machine102.ibm.com  -user1
machine102.ibm.com
secondary.ibm.com informix

#informix users /etc/hosts.equiv file on the secondary
machine1.ibm.com informix
machine2.ibm.com informix
machine3.ibm.com  -user1
machine3.ibm.com  -user2
machine3.ibm.com
primary.ibm.com informix
```

Windows

The same as on UNIX systems, on Windows you are required to set up a trusted connection in order to enable the communication between the servers. You have to maintain the hosts.equiv file, which is located in the directory %SystemRoot%\system32\drivers\etc. The content of the file is similar to the file on UNIX.

Verifying your settings

If a trusted connection is not set up properly, a connection attempt between the newly started servers fails. Depending on the implementation, either the communication is discarded or the server tries to reconnect a number of times. Look into the online.log of the target machine in case your database server is not able to connect. Missing entries in the .rhosts or /etc/host.equiv files results in the entries shown in Example 6-4.

Example 6-4   Not trusted access with error -956 reported on the target machine

```
#Errors in the logfile of the communication target
18:51:43  DR: Cannot connect to secondary server
18:51:43  DR: Turned off on primary server
18:53:53  listener-thread: err = -956: oserr = 0: errstr =
informix@secondary.ibm.com: Client host or user informix@secondary.ibm.com is
not trusted by the server.

18:56:13  listener-thread: err = -956: oserr = 0: errstr =
informix@secondary.ibm.com: Client host or user informix@secondary.ibm.com is
not trusted by the server.

#error in the logfile of the newly started server initiating the
communication for a SDS secondary
17:32:50  DR: Trying to connect to server, hdr_prim_11. If the specified server
           is not in Online mode, bring it to Online mode. The
           secondary server will attempt to connect to the
           primary for up to 7 minutes.
17:32:55  DR: Trying to connect to server, hdr_prim_11. If the specified server
           is not in Online mode, bring it to Online mode. The
           secondary server will attempt to connect to the
           primary for up to 7 minutes.
17:33:00  DR: Trying to connect to server, hdr_prim_11. If the specified server
           is not in Online mode, bring it to Online mode. The
           secondary server will attempt to connect to the
           primary for up to 7 minutes.
```

sqlhosts security options and trusted connections

If your company's security requirements permit you to use either the .rhost or hosts.equiv file, you can specify your database server communication directions in the *s* option field of the definition entry in the sqlhosts file. This security option field specifies, when trusted connections are allowed, which files are considered by IDS for the authentication.

The s option field has the following settings:

s=0	No trusted connections are allowed. Only connection requests with passwords are authenticated.
s=1	Trusted connections are authenticated using the /etc/hosts.equiv file.
s=2	Trusted connections are authenticated using the .rhosts file of the owner of the incoming session.

s=3 Both .rhosts and /etc/hosts.equiv are used for authentication. This is the default setting.

s=6 This is the special option used for HA environments. The target server only allows connection requests for the source database server that is specified in the $INFORMIXDIR/etc/hosts.equiv file. The requestor for the connection must be an HA communication thread. All other connection requests are rejected with an error -25539 by the target server. The database server that you intend to use for client server communication must have at least one additional server definition that does not specify the -s=6 option.

Example 6-5 shows the s=0 setting. Note that this setting disables any kind of server-server communication regardless of the ISTAR or HA functionality. Messages in the log files reported from the failed communication request by the database server are similar to that shown in Example 6-4 on page 212.

Example 6-5 sqlhosts entry on the primary using the s option field

```
hdr_prim_11 onsoctcp   9.156.134.174 23487 s=0
hdr_sec_11  onsoctcp   9.156.176.139 23485
```

6.2.2 Database server communication and firewall setups

Firewall is another approach for preventing denial-of-service attacks like sync flooding, port scanning for potential vulnerabilities, IP spoofing, or bombarding productively used database ports with IP SPAM packages. Firewall solutions, regardless of whether they are distributed with the operating system by default or developed by security specialized IT providers, allow the administrator to block all incoming, outgoing, and forwarding IP packages (from or to) as insecurely defined sources.

Firewall solutions are rule based. The defined rules control the access of the available local network resources such as ports provided by the server applications. The ports can be maintained by OS resources like daemons or third-party server software like a database server or a mail server. You can define which port can be accessed from which network, interface, or machine. Firewall also allows you to control the outgoing messages and which program is allowed to access the ports.

Setting up HA solutions in global infrastructures in order to separate the production environment from the standby server or realizing bunker backups most likely requires you to split the database server machines into different networks and secure them independently by firewall. If a commonly used firewall is set up to drop or reject insecure-classified IP packages and the attempts of reaching not assigned ports, you have to adjust your settings to enable every newly defined server-to-server communication. Enabling the communication between the database servers through the firewall should be a non-recurring task for each newly attached database server.

In this section we provide an introductory overview of settings that you have to do in order to enable the communication through the firewall. We focus our discussion only on the iptables utility provided with most Linux distributions and the Windows firewall application.

iptables on Linux

The *iptables* utility provides the Linux administrator with a complex interface to define the incoming and outgoing rules. If the IP forwarding is enabled, additional rules for the forward direction can be defined. In general, the default settings define that all incoming messages, regardless from which interface, are accepted. That means that the firewall is turned off. Example 6-6 shows the sample predefined rules.

Example 6-6 Default rules in the firewall settings with iptables on Linux

```
# iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
Chain FORWARD (policy ACCEPT)
target     prot opt source               destination
Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
```

A recommended approach to define secure settings for the firewall is to disallow everything first and then step-by-step allowing all necessary access to the local resources. As shown in Example 6-7, we disallowed on the SDS primary database server all incoming messages and added the rules to allow the ssh access. You have to add all other necessary services for this machine to work properly for DNS, SMTP, IPP, and ICMP messages, and so on.

Example 6-7 Clean up the default, disallow everything but ssh

```
iptables -X
iptables -F
iptables -P INPUT DROP
iptables -P OUTPUT DROP
iptables -P FORWARD DROP

iptables -A INPUT -i lo -j ACCEPT
iptables -A OUTPUT -o lo -j ACCEPT
iptables -A OUTPUT -o eth4  -p udp --sport 22 -j ACCEPT
iptables -A OUTPUT -o eth4  -p tcp --sport 22 -j ACCEPT
iptables -A INPUT -i eth4  -p udp --dport 22 -j ACCEPT
iptables -A INPUT -i eth4  -p tcp --dport 22 -j ACCEPT

#allow everything with
#iptables -A INPUT -i eth4 -j ACCEPT
#iptables -A OUTPUT -o eth4 -j ACCEPT
```

The first iptables call drops all existing rules in the firewall. We than define that all messages, regardless of which way they want to go, are dropped by default. We allow the local loopback communication and set up the communication for port 22 (used by ssh) for incoming and outgoing TCP packages. This particular machine has defined the Ethernet interface with the name eth4.

Example 6-8 shows the sqlhosts file settings that we want to have on both machines of a SDS environment. With this particular firewall configuration, we are not able to set up a connection between our two database servers. The access of the database ports is blocked.

Example 6-8 sqlhosts file entries for a SDS environment with two servers

```
sds_prim onsoctcp   9.156.134.174 10000
sds_sec  onsoctcp   9.156.176.139 10001
```

An attempt to bring up the secondary database server with our current settings causes the IDS to hang. If you look into the online.log file, you will see the messages shown in Example 6-9. You may notice that these messages are the same messages as shown in Example 6-4 on page 212. As opposed to the authentication problems, the primary SDS server does not report any errors in the online.log.

Example 6-9 Failed SDS secondary server startup because of firewall rules

```
Thu Oct 18 11:13:39 2007

11:13:39  Warning: ONCONFIG dump directory (DUMPDIR) '/tmp' has
insecure permissions
11:13:39  Event alarms enabled.  ALARMPROG =
'/vobs/IDS11/etc/alarmprogram.sh'
11:13:39  Booting Language <c> from module <>
11:13:39  Loading Module <CNULL>
11:13:39  Booting Language <builtin> from module <>
11:13:39  Loading Module <BUILTINNULL>
11:13:44  DR: DRAUTO is 0 (Off)
11:13:44  DR: ENCRYPT_HDR is 1 (HDR encryption Enabled)
11:13:44  Event notification facility epoll enabled.
```

```
11:13:44  IBM Informix Dynamic Server Version 11.10.UC1     Software
Serial Number AAA#B000000
11:14:45  DR: Trying to connect to server, sds_prim_11. If the
specified server
           is not in Online mode, bring it to Online mode. The
           secondary server will attempt to connect to the
           primary for up to 7 minutes.
```

In this particular case, the best way to differentiate the reasons for the failed connection attempt is to check further the status of the threads on the secondary database server. The main_loop thread on a secondary SDS server is waiting on a socket OS system call during the startup of the SMX communication layer, as shown in Example 6-10. Running **netstat -a** on the shell most likely will show you a connection requested by the local port 10001 waiting in a SYNC.

Example 6-10   main_loop thread waiting on the firewall for a connection

```
Stack for thread: 9 main_loop()
 base: 0x113db000
  len:   69632
   pc: 0x0888746b
  tos: 0x113e9d80
state: sleeping
   vp: 5

0x0888746b (oninit)yield_processor_mvp(0x10bca098, 0x11549d68, 0x20, 0x0, 0x0)
0x0888859e (oninit)mt_yield(0x1, 0x17, 0x11549d68, 0x6, 0x113ea098, 0xb71ba0)
0x088fb754 (oninit)net_wait_for_connect(0x11549d68, 0x4, 0x1154980, 0x10dbfb18)
0x088fe722 (oninit)connsocket(0x11549800, 0x113ea1d0, 0x10bbe020, 0x1b)
0x0893879a (oninit)tlConnect(0x114de29c, 0x113ea1d0, 0x10bbe020, 0x158)
0x089376c6 (oninit)slSQIreq(0x114de29c, 0x204, 0x113ea1d0, 0x113ea188)
0x0892ead0 (oninit)pfConReq(0x114de29c, 0x1154101c, 0x114de264, 0x0)
0x08928e6b (oninit)cmReqSync(0x114de29c, 0x113ea604, 0x3c, 0x113ea4e0)
0x089298a9 (oninit)cmConReq(0x114de29c, 0x113ea604, 0x113ea4a8, 0x1139913a)
0x088ebb1a (oninit)ASF_Call(0x113ea660, 0x113ea600, 0x11303894, 0x11217ba8)
0x0873842f (oninit)rsasf_connect(0x8b76ecc, 0x113ea888, 0x113ea88c, 0x113ea930)
0x0890e10c (oninit)smx_create_pipes(0x113eb2e0, 0x1, 0x111d9bc4, 0x113eb2e0)
0x08ada754 (oninit)cloneCreatePipes(0x113eb2e0, 0xbde344, 0xbd0794)
0x08ae8241 (oninit)sdcloneSetup_Int(0x0, 0x8b76ecc, 0x0, 0x0, 0x113eb478, 0x1)
0x08ad383e (oninit)sdcloneSetup(0x0, 0x113eb470, 0x1, 0x8bd9952, 0xcf33f3)
0x0813288c (oninit)initseg (0x2, 0x0, 0x113ebee0, 0x8c96100, 0x10090bc4, 0x0)
0x0813a5be (oninit)main_loop(0x0, 0x0, 0x0, 0x0, 0x0, 0x0)
0x0888f679 (oninit)startup (0x3e, 0x113ec9d0, 0x113d9fd0, 0x9d0, 0x227c7371)
0x00000000 (*nosymtab*)0x0
```

In order to allow the connection in both ways, you have to add to your firewall rules (or policies) to these two settings, as shown in Example 6-11.

Example 6-11 Enable the communication at the firewall

```
#iptables -A INPUT -i eth4 -p tcp -s 9.156.176.139  --dport 10000 -j ACCEPT
#iptables -A OUTPUT -o eth4 -p tcp -d 9.156.176.139  --dport 10001 -j ACCEPT
```

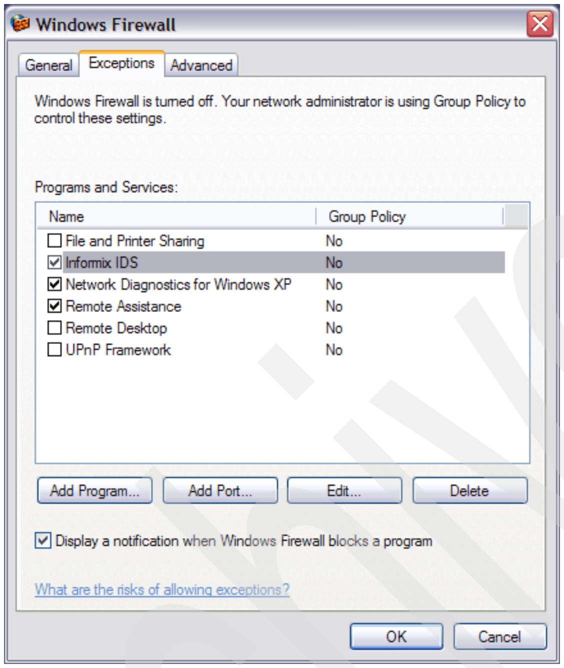
Windows Security Center

On Microsoft® Windows XP SP2, a firewall application is distributed with the operating system. It is part of the security center. This application can be accessed through **Start** → **Accessories** → **SystemTools** → **Security Center**. This application provides three functions: virus protection, automatic updates, and firewall. We particularly want to look at the firewall functionality.

You also can access the firewall application and the settings from the Control Panel Windows Firewall icon. In the General tab, you can switch the firewall on and off. You also can set the programs or ports that are excepted from blocking by the firewall. Other options include logging of access attempts, the control of ICMP, and so on.

Our goal is to enable communication for the IDS through the firewall. Under the Exceptions tab, add either the oninit.exe program to the exception program list, or add the source and target ports of the involved database server in the port list.

When using the program exception list, you have to identify all the programs using this database server, such as dbaccess.exe, onspaces.exe, and so on, and separately add them to the list. From an administration perspective, using the port exception list is the easier way. Figure 6-2 shows the Windows Firewall utility for maintaining the access.


Windows Firewall

General | Exceptions | Advanced

Windows Firewall is turned off. Your network administrator is using Group Policy to control these settings.

Programs and Services:

Name	Group Policy
☐ File and Printer Sharing	No
☑ Informix IDS	No
☑ Network Diagnostics for Windows XP	No
☑ Remote Assistance	No
☐ Remote Desktop	No
☐ UPnP Framework	No

☑ Display a notification when Windows Firewall blocks a program

Figure 6-2 Windows XP - configure your firewall exceptions

6.3 Avoid raw data transmission between IDS servers

The major security concern in the data exchange between the IDS servers across the network is the possibility of the raw data being read by an undesirable third-party. Network sniffers can go around the SQL-based permission shield to catch content of the production data. Another possible scenario could be the hack of a particular control message in an HA environment and spoof of the source of the message to the target, causing unexpected activities on the target system such as failover or shutdown.

Different from the client server communication, where passwords may be exchanged, IDS servers do not exchange passwords. They exchange log entries, database content, or control messages. To prevent from sending these data in raw format, most communication layers used by the IDS11 HA solutions support data encryption in the network flow. Since most of the HA features require setting their own ONCONFIG parameter, in this section, we provide a complete overview of how to enable data encryption and verify the parameter settings as a basic check of success.

6.3.1 Define the basic encryption settings

Before looking at the special setup requirements for the particular features to enable the encryption, we want to look first at the general base settings applicable for all features that influence which messages, and how and with which keys are encrypted. IDS 11 supports a set of ONCONFIG parameters that influence the network data flow encryption regardless the type of data exchanged. The encryption interface used by the IDS for the specified settings is provided by the IBM Crypto for C (ICC) product. This product is CMVP 140-2 Overall Level 1 evaluated.

Basic ONCONFIG parameters

The ONCONFIG parameters that influence the encryption of the messages sent over the network are:

- ENCRYPT_CIPHERS
  - Define the set of ciphers (the encryption algorithm), such as AES, DES, or advanced DES, and their encryption modes, which can be used for the encryption for the data exchanged between the database servers.
  - Our recommendation is to have at least a certain set of AES and DES ciphers so that the database source server can periodically alternate the encryption mode to make the decrypting more difficult.
  - The currently applied cipher is randomly changed on the source, depending on the ENCRYPT_SCHEDULE settings.
- ENCRYPT_MAC
  - The Security Hash Algorithm (SHA1) or XOR algorithm is used for generating an unique hash signature on the real data messages sent over network in order to ensure a data integrity check on the target.
  - Level definition between high and off is possible.
  - The configured level along with the size of the message to be send defines whether a signature is to be generated. If a signature is to be generated, the algorithm, SHA1 or XOR, is also decided.
- ENCRYPT_MACFILE
  - The signature generated depending on the ENCRYPT_MAC value can be additionally secured with a MAC key.

  - ENCRYPT_MACFILE specifies the location of one or more files that contain keys for securing the signature.
  - The server maintains a default MAC key internally in case no additional macfile parameter is configured.
  - The specified macfiles are generated with the GenMacKey executable as follows:

    ```
    GenMacKey -o <filename>
    ```

  - Make sure that the macfiles are the same on all the participating database servers and more than one macfile is specified for periodically switching the key.
- ENCRYPT_SCHEDULE
  - Defines the switch time for the macfiles and the ciphers if there are multiple options defined.
  - By default, the change happens every hour.

In summary, the communication between the IDS database servers with enabled encryption works as follows. When an IDS database server has to send an encrypted message to a target machine, the current private key and the current cipher for encrypting the raw message are determined first. In case one of the time frames specified by the ENCRYPT_SCHEDULE ONCONFIG parameter is expired, either cipher on the source machine is changed, or a new private key is generated using the Diffie-Hellman algorithm. Before the raw message is encrypted, a signature is generated based on the ENCRYPT_MAC security level setting. If the parameter is set to a value other than none, the signature is generated. The signature value is secured with the current key specified by the current ENCRYPT_MACFILE. After the message is encrypted using the current cipher and the current private key, the message is send to the target. All necessary changes such as the change of the cipher, current MAC, and private key are also added to the message sent to the target.

Sample settings

Example 6-12 illustrates a possible setting on an HDR primary database server. The encryption for HDR is switched on. The SHA1 algorithm for the hash key generation on the send message will be used all the time. The server alternatively uses the MAC key files hdrkey1 and hdrkey2 every 20 minutes. We use all available ciphers provided in the encryptions libraries besides the ECB mode. The electronic codebook (ECB) is a simple, weak encryption mode that can be cracked easy. The current cipher that is used by the database server is changed every 10 minutes.

Example 6-12 Sample encryption settings for database server with HDR

```
ENCRYPT_HDR 1
ENCRYPT_MAC high
ENCRYPT_MACFILE /usr/local/security/hdrkey1,/usr/local/security/hdrkey2
ENCRYPT_SCHEDULE 10,20
ENCRYPT_CIPHERS all
```

Setting value validation

All the ENCRYPT* ONCONFIG parameter settings are validated by IDS at database server startup time. If you specified an incorrect value, the server does not come up, but will reports an error message on the startup panel. Example 6-13 shows the case of an invalid value set in ENCRYPT_MAC. Keep in mind that only the setting values are verified, not the parameter names. For example, if ENCRYPT_CIPHER is used instead of ENCRYPT_CIPHERS, the database server does not notify you. This may result in unexpected behavior.

Example 6-13 Verification of the encryption values at startup time

```
$ oninit
The onconfig parameter [ENCRYPT_MAC higher] is invalid.
```

The ENCRYPT_MACFILE parameter specifies the location of a local file. The validation for this parameter checks the existence of the file, not the content of the file. In case you specified a file that does not exist, the server does not start. Example 6-14 shows an invalid ENCRYPT_MACFILE setting and the error message IDS gives.

Example 6-14 Special verification of the ENCRYPT_MACFILE parameter

```
$cat $INFORMIXDIR/etc/$ONCONFIG | grep ENCR
ENCRYPT_HDR 1
ENCRYPT_MAC high
ENCRYPT_MACFILE /tmp/notextists

$ oninit
The onconfig parameter [ENCRYPT_MACFILE /tmp/notexists] is invalid.
```

Any changes to the encryption-related ONCONFIG parameters, except the ENCRYPT_HDR setting, are not reported in the online.log at the next database restart.

Considerations

Once you have successfully set up the communication with encryption in the database server, you may notice that the database server allocates additional resources for the work. They are related to virtual processors (VPs) and memory.

Example 6-15 shows the VP list in the a database server from **onstat -g glo**. Depending on the number of CPU VPs, the same amount of crypto VPs are started. Be aware that using encryption significantly increases the security of your data, but it also requires additional resources in the database server. This should be considered as expectations in the network throughput and database performance.

Example 6-15 onstat -g glo and crypto VPs

```
Individual virtual processors:
 vp    pid      class       usercpu   syscpu    total
 1     11666    cpu         0.06      0.04      0.10
 2     11667    adm         0.00      0.00      0.00
 3     11668    crypto      0.24      0.00      0.24
 4     11669    crypto      0.00      0.00      0.00
 5     11670    crypto      0.00      0.00      0.00
 6     11671    cpu         0.00      0.00      0.00
 7     11672    cpu         0.00      0.00      0.00
 8     11673    lio         0.00      0.00      0.00
 9     11677    pio         0.00      0.00      0.00
 10    11678    aio         0.00      0.00      0.00
```

```
11    11679     msc         0.00      0.00      0.00
12    11680     aio         0.00      0.00      0.00
13    11681     aio         0.00      0.00      0.00
14    11682     aio         0.00      0.00      0.00
15    11683     soc         0.00      0.00      0.00
16    11684     soc         0.00      0.00      0.00
                tot         0.30      0.04      0.34
```

You also can see an additional small memory pool named Crypto in the output of **onstat -g mem**, as shown in Example 6-16.

Example 6-16 Memory pools and encryption

```
$ onstat -g mem | grep -i cry
sqcrypto     V     11399020 4096      2256     2          1
Crypto       V     11391020 81920     3736     1692       11
```

For a complete description of the encryption-related ONCONFIG parameters and their settings, refer to the *IBM Informix Security Guide*, G229-6389-00, and *IBM Informix Dynamic Server Administrator's Reference*, G229-6360-00.

6.3.2 RSS and SDS

With the implementation of the new HA features RSS and SDS in IDS11, a new communication protocol trusted server group multiplexer connections (SMX) was implemented also. SMX supports a multiplexing of sending messages that provides faster and more independent data exchange between the primary and secondaries. Here, multiplexing means that both primary and secondary database servers are able to send messages independently in parallel without waiting. In addition, the message sending requestor is not the same local thread every time. There are several sources that could place a request into a free slot and wake up the send thread to send a message.

Communication and threads

SMX communication is automatically used when RSS or SDS is set up and the first secondary server comes up. After the communication between the new secondary and the primary server is successfully established, you can see on the primary a new pair of threads, as shown in Example 6-17. The same pair of threads is also created on the secondary database server. The name of the threads depends on the name of the remote database server. In the example we used the name *sec* for the secondary.

Example 6-17 SMX threads on a SDS primary server

```
$onstat -g ath | grep smx
71      10f8453d8    10e508118 3    cond wait  netnorm      4cpu smxrcv sec
72      10f7ee1d8    10e509960 1    cond wait  smx pipe1    1cpu smxsnd sec
```

Parameter settings

SMX supports encryption of the data exchange between the primary and secondary SDS or RSS server. In order to enable the encryption, you have to add to your ONCONFIG file a new parameter ENCRYPT_SMX with the value set to 1. The default value is 0. Do not forget to set the value on both database servers in the communication pair. If you only want the encryption on one server pair in your SMX environment, specify the ENCRYPT_SMX as 1 on the primary and the paried secondary. We leave the value of ENCRYPT_SMX as 0 on the other secondaries that do not need the encrypted communication.

Setting verification

You can verify whether encryption is used for the data transmission in a database server in two different ways:

- onstat -g smx

  You can obtain the current status of the encryption by using the **`onstat -g smx`** command. This is the easiest way to obtain the information. In Example 6-18, the current infrastructure of an SDS environment involves one primary server and two attached secondary servers. The communication between the primary and the secondary server sds_sec_11_1 uses encryption. The database server sds_sec_11_0 does not use encryption for the data transfer.

  Example 6-18 Check the status of the SMX encryption with onstat -g smx

  ```
  $> onstat -g smx

  IBM Informix Dynamic Server Version 11.10.UC1     -- On-Line -- Up
  00:00:49 -- 31904 Kbytes
  ```

```
SMX connection statistics:
SMX control block: 0x111d8018

  Peer server name: sds_sec_11_1
  SMX connection address: 0x11cc4c30
  Encryption status: Enabled
  Total bytes sent: 5465
  Total bytes received: 2084
  Total buffers sent: 22
  Total buffers received: 50
  Total write calls: 22
  Total read calls: 50
  Total retries for write call: 0

  Peer server name: sds_sec_11_0
  SMX connection address: 0x11c4f2e0
  Encryption status: Disabled
  Total bytes sent: 3925
  Total bytes received: 585
  Total buffers sent: 14
  Total buffers received: 30
  Total write calls: 14
  Total read calls: 30
  Total retries for write call: 0
```

- Alarm control and remote administration

  Using the GUI-based alarm control and remote administration, you can obtain similar information from the sysmaster database. The new database table sysmx introduced in IDS 11 contains detailed SMX information. Example 6-19 shows output from the syssmx table.

Example 6-19 Check the status of the SMX encryption with the sysmaster interface

```
dbaccess -e sysmaster << EOF
select * from syssmx;
EOF

address            298601520
name               sds_sec_11_1
encryption_status  Enabled
buffers_sent       60
buffers_recv       180
bytes_sent         8993
bytes_recv         8056
reads              180
```

```
writes            60
retries           0


address           298119904
name              sds_sec_11_0
encryption_status Disabled
buffers_sent      51
buffers_recv      158
bytes_sent        6341
bytes_recv        4065
reads             158
writes            51
retries           0
```

Note that in IDS 11, you are not able to see the current setting of the ENCRYPT_SMX parameter on the database server by checking the sysconfigtab table in the sysmaster database. An unsuccessful attempt to select the ENCRYPT_SMX parameter from the sysmaster:sysconfigtab table is shown in Example 6-20.

Example 6-20 Checking the ENCRYPT_SMX value with the sysmaster interface

```
$dbaccess -e sysmaster << EOF
select * from sysconfigtab where cfname matches ”*SMX*”
EOF

No rows found
```

6.3.3 HDR

In an HDR environment, the content of the log buffer and heart beat status messages is exchanged between the primary and the secondary database servers. The value of the DRINTERVAL setting rules the frequency of shipping the logs from the primary to the secondary. In the case of creating indexes, all the necessary index information is also shipped from the primary to the secondary server, either with the log file records or under the invocation of a specific thread, depending on the ONCONFIG parameter settings.

Communication and threads

The communication between the HDR pair is provided by a certain set of threads. Example 6-21 on page 227 shows the threads on the primary and secondary systems. On the primary, the major part of the communication is done by the dr_prsend thread, which is responsible for shipping the log buffer to the secondary. On the secondary, the dr_secrcv thread receives the messages and distributes the messages to the threads attached with a working chain for applying the log entries. These threads are also responsible for message encryption and decryption.

Example 6-21 Threads providing the communication for HDR

```
#communication threads on the primary HDR server
 43      11dca2f0 10d6d420 1    cond wait  drcb_bqf     1cpu        dr_prsend
 76      116bd328 10d6dfb0 1    sleeping secs: 1        1cpu        dr_prping
 77      116bd018 10d6b738 1    sleeping secs: 4        1cpu        dr_idx_send

#communcation threads on the secondary HDR server
 17      1156ec88 10d6b738 2    cond wait  netnorm      1cpu        dr_secrcv
 20      116b98f0 10d6c890 2    sleeping secs: 22       1cpu        dr_secping
 21      116b9c80 10d6ce58 2    sleeping secs: 5        1cpu        dr_idx_recv
```

Setting value

In addition to the specification of the general encryption-related ONCONFIG parameters, you need to add the ENCRYPT_HDR parameter to the ONCONFIG file on both servers. This parameter has to be set to 1.

Example 6-22 presents a possible setting for the encryption parameters. Here we enabled the encryption for the HDR. The settings of the ENCRYPT_MAC parameter define that the SHA1 algorithm is used to generate the hash signature for all messages larger than 20 bytes. For all others, the XOR algorithm is used. The internal IDS macfile is used for securing the hash information since no other macfile is specified in the ONCONFIG file. All ciphers are allowed for encrypting the messages, and the cipher is changed every 120 minutes.

Example 6-22 Sample encryption settings for database server with HDR

```
ENCRYPT_HDR 1
ENCRYPT_MAC medium
ENCRYPT_SCHEDULE 120,120
ENCRYPT_CIPHERS all
```

Setting verification

In order to set up the encryption successfully in the HDR environment, the communication settings on the primary and secondary have to match. If you use incompatible settings, the database server does not come up. The messages logged in the online.log vary depending on which setting value is incompatible. Example 6-23 shows the attempt to start the secondary HDR server without matching encryption settings. The log file shows the error message clearly.

Example 6-23 Incompatible encryption parameter between primary and secondary

```
22:25:18  Physical Recovery Started at Page (1:5734).
22:25:18  Physical Recovery Complete: 0 Pages Examined, 0 Pages 
Restored.
22:25:18  DR: Owner of the disk is set as hdr_sec_11.
22:25:18  DR: Trying to connect to primary server = hdr_prim_11
22:25:19  Dataskip is now OFF for all dbspaces
22:25:19  Restartable Restore has been ENABLED
22:25:19  Recovery Mode
22:25:23  DR: Secondary server connected
22:25:24  DR: Configuration parameter values of the paired servers do 
not match:
              Parameter: ENCRYPT_HDR
              Current server's value: 1
              Paired  server's value: 0
              The parameter values must match.
22:25:24  IBM Informix Dynamic Server Stopped.
```

Example 6-24 shows incompatible settings on the ENCRYPT_MACFILE parameter and the log message. IDS did not report the setting values in the error message.

Example 6-24 Different settings for the ENCRYPT_MACFILE parameter

```
#settings on the primary
$cat $INFORMIXDIR/etc/$ONCONFIG | grep ENC
ENCRYPT_HDR 1
ENCRYPT_MAC high
ENCRYPT_MACFILE /tmp/holgerk,/tmp/holgerk1

#settings on the secondary
$cat $INFORMIXDIR/etc/$ONCONFIG | grep ENC
ENCRYPT_HDR 1
ENCRYPT_MAC high

Online.log
17:53:59  Restartable Restore has been ENABLED
```

```
17:53:59  Recovery Mode
17:54:01  DR: Secondary server connected
17:54:02  DR: Error accepting encryption connection
17:54:03  DR: Log Record Apply Thread Exited.
17:54:04  DR: Turned off on secondary server
17:54:16  DR: Secondary server connected
17:54:17  DR: Error accepting encryption connection
```

In an HDR environment, the current setting of the ENCRYPT_HDR parameter can be verified by looking in the online.log file. After the database server is started, the ENCRYPT_HDR setting is logged. Example 6-25 shows sample output if the encryption is switched on.

Example 6-25   Online.log output for HDR with encryption enabled

```
22:10:40  Event alarms enabled.  ALARMPROG =
'/vobs/IDS11/etc/alarmprogram.sh'
22:10:40  Booting Language <c> from module <>
22:10:40  Loading Module <CNULL>
22:10:40  Booting Language <builtin> from module <>
22:10:40  Loading Module <BUILTINNULL>
22:10:45  DR: DRAUTO is 0 (Off)
22:10:45  DR: ENCRYPT_HDR is 1 (HDR encryption Enabled)
```

Similar to the SDS and RSS environment, the current setting of the ENCRYPT_HDR cannot be obtained by selecting the sysconfigtab table in the sysmaster database. But you can use the **onstat -g dri** to check the current parameter setting. Example 6-26 shows where you can find the value in the output.

Example 6-26   onstat -g dri to verify encryption settings within a HDR environment

```
$> onstat -g dri

IBM Informix Dynamic Server Version 11.10.UC1     -- Fast Recovery
(Sec) -- Up 00:01:37 -- 31904 Kbytes

Data Replication:
  Type           State        Paired server        Last DR CKPT (id/pg)
  HDR Secondary  off          hdr_prim_11                  -1 / -1

  DRINTERVAL   30
  DRTIMEOUT    30
  DRAUTO       0
  DRLOSTFOUND  /usr/informix/etc/dr.lostfound
  DRIDXAUTO    0
  ENCRYPT_HDR  1

informix@Linux-009156134174:/vobs/IDS11/hdr11_prim> onstat -g dri

IBM Informix Dynamic Server Version 11.10.UC1     -- On-Line (Prim) --
Up 00:02:23 -- 31904 Kbytes

Data Replication:
  Type          State       Paired server       Last DR CKPT (id/pg)
  primary       off         hdr_sec_11                  -1 / -1

  DRINTERVAL   30
  DRTIMEOUT    30
  DRAUTO       0
  DRLOSTFOUND  /usr/informix/etc/dr.lostfound
  DRIDXAUTO    0
  ENCRYPT_HDR  1
```

Another way to check the success of your changes is by using a network traffic trace tool such as tcpdump on Linux. If encryption changes are successfully applied, you should not be able to read the data. Perform the following steps to verify:

1. Check the communication port of the secondary server in the sqlhosts or /etc/services file, as shown in Example 6-27.

Example 6-27 Obtain the communication port

```
$cat $INFORMIXDIR/etc/sqlhosts | grep sec
hdr_sec_11  onsoctcp       Localhost    23485
```

2. Using ifconfig to verify the interface name where the secondary database server is listening. The output in Example 6-28 shows that HDR was set up on the same machine, so local loopback is used.

Example 6-28 Obtain the interface name

```
# ifconfig
eth0      Link encap:Ethernet  HWaddr 00:0D:60:0F:30:70
...

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
...
```

Run **tcpdump** listening on the interface with the message filter to only print the incoming TCP packages on the secondary's port number.

tcpdump -XX -s 8192 -i lo dst port 23485

3. Run any log producing activity on the primary and issue a checkpoint.
4. Monitor the output produced by **tcpdump**. Example 6-29 shows output without encryption.

Example 6-29   Dump the message

```
tcpdump -XX  -s 8192 -i lo  dst port 23485

22:23:54.453118 IP6 localhost.58604 > localhost.23485: P 4506:8602(4096)
....
   0x03c0:  8000 0000 0474 6573 7469 6e66 6f72 6d69  .....testinformi
   0x03d0:  7820 2020 2020 2020 2020 2020 2020 2020  x...............
   0x03e0:  2020 2020 2020 2020 2000 1001 de00 0000  ................
   0x03f0:  7100 0400 0100 0000 0000 0000 0000 0000  q...............
   0x0400:  0099 c800 7100 0154 5000 0000 0000 0000  ....q..TP.......
   0x0410:  0000 0000 1000 0000 1000 0001 0001 0000  ................
   0x0420:  0000 0000 0000 0000 0008 0000 0000 0000  ................
   0x0430:  0000 0000 0000 0000 0000 0000 1c02 0000  ................
   0x0440:  0000 3a00 1000 0000 0000 0000 0f00 0000  ..:.............
   0x0450:  3473 0100 12fa 0300 3401 1000 3401 1000  4s......4...4...
   0x0460:  0000 0000 070b 0000 0100 0000 0700 0000  ................
   0x0470:  0d00 0000 0000 0000 0000 0000 0000 0100  ................
   0x0480:  a100 0000 a400 e800 0474 6573 7469 6e66  .........testinf
   0x0490:  6f72 6d69 7820 2020 2020 2020 2020 2020  ormix...........
```

5. Enable the encryption setting, restart both servers, run the same activity on the primary again, and check the **tcpdump** output. Example 6-30 shows that the content pass through the next work is encrypted.

Example 6-30   Content encrypted

```
#After the encryption is switched on

   0x03c0:  e0fa a16b 18b8 b05b e00f 4ce2 721b 280b  ...k...[..L.r.(.
   0x03d0:  6d68 b61f 09ef 89a8 7bc2 6c5c 6104 aef3  mh......{.l\a...
   0x03e0:  6994 59b2 9c45 0f68 f271 baf9 57d7 d02e  i.Y..E.h.q..W...
   0x03f0:  3a73 9745 513c 3fb9 9076 c3ef 6213 fcc8  :s.EQ<?..v..b...
   0x0400:  1243 83d9 34a1 c9ff 0aca 93da c709 db86  .C..4...........
   0x0410:  7ead 691c dc1c 39ac 6a6a 6faf c563 3dda  ~.i...9.jjo..c=.
   0x0420:  be41 58d0 2e4f 863a 20cf 4c7d ef81 a168  .AX..O.:..L}...h
   0x0430:  4e39 8ae1 f617 1333 9b4b 8253 e61b f5c3  N9.....3.K.S....
   0x0440:  087d c32b d6af 5827 ccff 6f4b 1474 317f  .}.+..X'..oK.t1.
   0x0450:  a8a0 3818 c89d 8652 3fcf 6792 09c8 daf1  ..8....R?.g.....
   0x0460:  3d9d 70a3 81a5 8e43 733b 33c4 16d4 0749  =.p....Cs;3....I
```

```
0x0470:  9622 4d22 0d5c af57 e11d ec6c 7baa c953  ."M".\.W...l{..S
0x0480:  fdec fc05 875b e319 f756 1031 2493 666a  .....[...V.1$.fj
```

6.3.4 ER

The major components defining the ER infrastructure are the replication server and replication sets. The replication server defines the role of the current database server in the ER communication network. The replicate set defines the replicating database and table, distribution directions, schema verification information, conflict resolution strategies, and row change distribution schedules.

These two components rule the entire data replication starting from what data change has to be replicated from the log entry of the source system until the change is applied on the target server. Data are the major part in the messages sent over the network in an ER environment. In addition to sending out data, all the replication definition changes are sent out within control messages. Distributing the replication definitions across all the database servers affected ensures a consistent view of the data flow. Also, acknowledge messages are sent back to the operation requestor in order to forward the status of the requested operation.

Communication and threads

The NIF interface is the component in the database server that ensures the data transmission and receiving. This interface is represented by the existence of threads shown in Example 6-31. For each target database server to which the local server has to send the data, there is a pair of corresponding communication processing threads. One is responsible for sending messages and the other for receiving messages.

Example 6-31   Communication in the ER environment with NIF

```
51      11bb2970 10d713b8 1    sleeping secs: 16       3cpu        CDRNsT2932
54      1203d658 10d6bd00 1    cond wait  netnorm      3cpu        CDRNr2932
```

In order to encrypt the messages sent out by the NIF threads, you have to set the ENCRYPT_CDR ONCONFIG parameter to 1. This parameter works with the other encryption parameters discussed in the previous sections. In Example 6-32, we specified a specific set of ciphers to be used by the database server for encrypting the network data. The MAC hash key generation over the message is switched off. No MACFILE is needed in this case.

Example 6-32 ER environment with a low security level for the MAC values

```
ENCRYPT_CDR 1
ENCRYPT_MAC off
ENCRYPT_CIPHERS des:ecb,des:cbc,ede:cbc,desx:cbc
```

Setting value

IDS 11 provides the ability to change various ER-related ONCONFIG parameters and environment variables on the fly. This release introduces a new onstat option, **`onstat -g cdr config`**, for monitoring the current ER settings in the database server. This onstat option can also be used to monitor the status of the current encryption settings. Example 6-33 shows sample output.

Example 6-33 onstat -g cdr config output for encryption

```
$onstat -g cdr config
ENCRYPT_CDR:

    ENCRYPT_CDR configuration setting:         1
ENCRYPT_CIPHERS:

    ENCRYPT_CIPHERS configuration setting:               [None configured]
ENCRYPT_MAC:

    ENCRYPT_MAC configuration setting:                            high
ENCRYPT_MACFILE:

    ENCRYPT_MACFILE configuration setting:      /tmp/macfile1,/tmp/macfile2
ENCRYPT_SWITCH:

    ENCRYPT_SWITCH configuration setting:         60,60
```

Setting verification

Same as all other HA features, in a ER environments, a communication between the database servers can be successfully established only if the configuration parameters are compatible. In case your settings do not allow communication between the database servers, the connection is disconnected and a message is logged in online.log. Example 6-34 shows an incompatible setting the log entry.

Example 6-34 Incompatible settings between primary and target in an ER environment

```
#On the source
$onstat -c | grep ENCR
ENCRYPT_CDR 1
ENCRYPT_MAC low
ENCRYPT_MACFILE /tmp/macfile1


#On the target
onstat -c | grep ENCR

ENCRYPT_CDR 1
ENCRYPT_MAC high
ENCRYPT_MACFILE /tmp/macfile

$tail $INFORMIXDIR/tmp/online.log
14:41:48  CDR connection to server lost, id 2930, name <primary>
Reason: CDR Encryption incompatability with primary.
```

Changing the parameter on the fly

In IDS 11, the new feature to change the ER configuration values dynamically in the database server cannot be used to adjust incorrect or incompatible settings of the communication-related parameters.

Example 6-35 shows a test scenario. We tried to bring up the server with the wrong settings. As expected, the connection was dropped at the handshaking of the database servers and the associated message logged in the log file. The parameter value was changed successfully as verified by onstat, but the connection was not reestablished.

Example 6-35 Try to adjust incompatible encryption values

```
#Primary
ENCRYPT_CDR 1
ENCRYPT_MAC high
ENCRYPT_MACFILE /tmp/macfile

#Target
```

```
ENCRYPT_CDR 1
ENCRYPT_MAC low
ENCRYPT_MACFILE /tmp/macfile_1

18:07:08  CDR: Re-connected to server, id 2932, name <secc>
18:07:08  CDR Encryption incompatability with secc.
18:07:14  CDR connection to server lost, id 2932, name <secc>
Reason: CDR Encryption incompatability with secc.

#On the target
cdr cha onconfig "ENCRYPT_MAC high"
cdr cha onconfig "ENCRYPT_MACFILE /tmp/macfile"

$ onstat -g cdr config | grep -i enc
ENCRYPT_CDR:
    ENCRYPT_CDR configuration setting:         1
ENCRYPT_CIPHERS:
    ENCRYPT_CIPHERS configuration setting: [None configured]
ENCRYPT_MAC:
    ENCRYPT_MAC configuration setting: high
ENCRYPT_MACFILE:
    ENCRYPT_MACFILE configuration setting: /tmp/macfile
ENCRYPT_SWITCH:
    ENCRYPT_SWITCH configuration setting:         0,0

#change a row in the base table of a replicate
18:27:18  CDR: Re-connected to server, id 2932, name <secc>
18:27:18  CDR Encryption incompatability with secc.
```

6.3.5 ISTAR

ISTAR (distributed SQL) is the ability to request data exchange between database servers by SQL statements. In general, there is a *coordinator* database server on which the query is executed. In addition, there are remote database servers, referred as *participants*, to which the request is sent for selecting, inserting, or updating data. The remote databases referred by ISTAR queries have to have the same log mode as the local database situated on the coordinator. Example 6-36 shows some ISTAR SQL statements.

Example 6-36 ISTAR SQL statements

```
#Simple select
dbaccess -e stores_demo << EOF
SELECT *
FROM stores_demo@lech10uc6w5shm:customer remote, customer local
```

```
WHERE local.customer_num=remote.customer_num
EOF

#Update with transaction
begin work;
UPDATE stores_demo@lech10uc6w5shm:customer
SET fname="Smith"
WHERE customer_num=110;
COMMIT WORK;
```

In the case of a distributed query, the local database server opens the connection to the remote database server and acts as a client. On the remote database server, a session is established to handle the requests from the coordinator. Different from normal client-server connections, the thread handling a server-server connection is named srvinfx.

Example 6-37 shows the distributed transactions in the onstat -x output of the coordinating database server.

Example 6-37 ISTAR sessions and their appearance on the remote server

```
#local and remote connection
$onstat -g ath | tail -3
 37      6008c608 4020436c 1    cond wait  netnorm      1cpu        sqlexec
 38      500bce40 40203448 1    cond wait  netnorm      1cpu        srvinfx

#onstat -g sql
session                                        #RSAM    total      used
dynamic
id       user     tty      pid      hostname threads  memory     memory
explain
14       informix 27       3782     rios.mun 1        86016      63816      off

tid      name     rstcb    flags    curstk   status
38       srvinfx  40203448 Y--P---  3992     cond wait(netnorm)
..
sqscb info
scb      sqscb    optofc   pdqpriority sqlstats optcompind  directives
50087890 502f5018 0        0           0        2           1

Sess  SQL            Current           Iso Lock       SQL  ISAM F.E.
Id    Stmt type      Database          Lvl Mode       ERR  ERR  Vers  Explain
14    -              stores_demo       CR  Not Wait   0    0    9.50  Off

Last parsed SQL statement :
  update stores_demo:"informix".customer set fname = 'Smith'  where
    (customer_num = 110 )
```

```
#onstat -x
$ onstat -x

IBM Informix Dynamic Server Version 10.00.UC6W5   -- On-Line -- Up 00:09:42 --
46240 Kbytes

Transactions
address  flags userthread locks  beginlg curlog  logposit   isol    retrys
coord
4022b018 A---- 40202018   0      0       16      0x76050    COMMIT  0
4022b220 A---- 40202524   0      0       0       0x0        COMMIT  0
4022b428 A---- 40202a30   0      0       0       0x0        COMMIT  0
4022b630 A---- 40202f3c   0      0       0       0x0        COMMIT  0
4022b838 A---S 40203448   3      0       0       0x0        COMMIT  0 remsvr
4022ba40 A---- 40203e60   0      0       0       0x0        COMMIT  0
4022bc48 A---- 40203954   0      0       0       0x0        COMMIT  0
4022be50 A---- 4020436c   1      0       0       0x0        COMMIT  0
 8 active, 128 total, 8 maximum concurrent
```

You cannot set up encryption for the communication for ISTAR. The messages sent between the database servers are in the raw format all the time. This also is the case when an ISTAR query runs on an HDR primary and secondary pair or an ER source and target that have data encryption set up for the communication. In these particular scenarios, the HA threads executing the communication are responsible for encrypting and decrypting the network messages.

7

Security issues with backup and restore

Backup and restore are the facilities of a modern database server that provide an easy-to-use, fast in execution, and reliable way to re-establish a damaged or lost production database environment or to set up sophisticated HA solutions by copying the backup copy from an existing system into new hardware. Since the backed up data is also a one-to-one copy of the production environment, the same security requirements have to be applied on the backup copies.

In this chapter we start with a detailed overview of the permissions needed for executing different types of backup and restore provided by the IDS11 database server. This includes a discussion about the backup utility execution permissions and the permissions for the files created by the backup.

Next, we introduce the options of using storage manager for backing up the database with data compressed or encrypted.

Finally, we describe the new IDS 11 functionality, filters, for backup and restore. This new feature enables the DBA to incorporate filter programs to compress or encrypt data to advance the default solutions provided by storage managers.

7.1 IDS 11 backup and restore strategies

Using IDS backup and restore facilities, DBA can achieve two major goals:

- Restore a production database environment after a hardware or software failure.
- Set up IDS-provided HA solutions using the backup and restore function.

  As availability of the production database environment becomes more and more crucial for the companies, backup and restore functionality has a new focus on the database-provided HA solutions. The backup and restore functionality needs to be used as the key feature to set up IDS-provided HA solutions like HDR, RSS, and ER.

Database backup processes your production databases, transmits the data over the network to the storage manager server, or writes the data to disk files. You should apply the same security standards and attention to the database backups as you do to the database objects. We provide an overview of the various IDS backup strategies to help you understand the security weakness in running a backup or restore. More detailed information about the backup and restore functionality, setup, initialization, configuration files need, and new features in IDS 11 can be obtained from the IBM Informix Information Center:

`http://publib.boulder.ibm.com/infocenter/idshelp/v111/index.jsp`

In general, IDS 11 supports the following key backup strategies:

- A serial backup of all dbspaces with ontape

  Provides the ability to back up all dbspaces in serial to:

  – Disk and directory
  – stdout
  – Tapes
  – Remote devices using the rsh
- Serial or parallel backups of all or selected dbspaces in an Storage Manager (SM) environment with onbar
  – Supports parallel and serial backups of the dbspaces in a client server storage manager solution.
  – Storage manager resides, most of the time, on a remote machine.
  – Provides the backup to:
    - Local or remote disks
    - Tapes

- External backup
  - Provides the possibility to block the database server at a checkpoint where a consistent status between memory and disk is achieved
  - Third-party application or solutions can be used for copying the dbspaces
  - Supports restore from third-party application in combination with storage manager (onbar) based log replay
- Offline backup
  - Using solely the third-party solution to back up the dbspaces maintained by the database server.
  - Offline mode of the database server is necessary.

Even though these four strategies are quite different, at the end, they all maintain or duplicate the production data of the company. It does not help you to have multiple stages of security control in your living production database server based on SQL access control for the data with views, permissions, and roles, including auditing of the activities of your users, but the backup is running into a backup file with the permissions open for anyone.

You may have noticed that in the introduction of the Informix backup strategies, we have pointed out where the data are finally stored, and the opportunities of sending the data through the network to a remote machine. Checking these backup strategies for possible working points for intruders, we derive the following major security focus points:

- Execution permissions of the backup utilities
- File permissions for the backup files
- Clear text backup stream transmission from one machine to another in a client server environment through the network

In our discussion addressing these particular points, we focus on the database-provided backup solutions onbar and ontape. For onbar and ontape, IDS has complete control of how and where the data are written, and the distribution of the database server files define the permissions of the executables used for the backup.

For offline backups and external backups, any third-party backup solutions and security standards could be used, without any possibility of an influence from the database server side.

7.2 Onbar and ontape permissions

Accessing the content of the backed up data and looking for possible leaks to switch the user representation with sticky bits are most likely the main focus of system intruders. In this section, we discuss the default permission settings for any kind of files used in the backup scenarios by the database server. We also provide you with some hints as to where the default mechanisms are sufficient and where you have to pay an more attention.

7.2.1 Execute permissions for the backup utilities on UNIX

The focus on the permissions for the executables is the starting point of our security discussion in the backup and restore area. You should know the default settings defined by Informix to notice any changes that could enable others to mask their identity in the system. In the IDS distribution, the installation program defines the standard permission definitions that are necessary for a DBA to enable the onbar and ontape utility to talk with the database server.

Example 7-1 shows the default settings for the onbar and ontape files. You may noticed the sticky bit defined on the onbar_d and ontape utility. This special execution permission is necessary for the internal redefinition of resource limits at program startup time, loading external shared libraries, and the possibility to switch the effective user in the process environment.

Example 7-1 Default permissions for the backup utilities

```
$> ls -al $INFORMIXDIR/bin/onbar*
-rwxr-xr-x  1 informix informix    3897 2007-06-19 07:49 /IDS11/bin/onbar
-rwsr-sr-x  1 root     informix 2240124 2007-06-22 17:21 /IDS11/bin/onbar_d

$> ls -al $INFORMIXDIR/bin/ontape
-rwsr-sr-x  1 root informix 1873878 2007-06-22 17:20 /IDS11/bin/ontape
```

In an common installation without role separation, only the informix user and the root user are allowed to run a backup with the onbar utility. For any other user, the onbar utility denies the service. This is similar for the ontape utility, with the small difference that all users assigned to the informix group are allowed to run backups with informix.

The difference is shown in Example 7-2. This should cause additional attention to which users are additional in the informix group. In a secured system, only the informix user is a member of the informix group.

Example 7-2 Who can run backups

```
Ontape
#User belonging to world

worlduser@Linux> ontape -s -L 0

Must be dbsa to run this program.
Program over.

#informix, root , informix group users
InformixGroupUser@Linux> ontape -s -L 0

Please mount tape 1 on /backup and press Return to continue ...
100 percent done.

Please label this tape as number 1 in the arc tape sequence.
This tape contains the following logical logs:

 9

Program over.

onbar

#All other Users than root and informix
informixgroupuser@Linux> onbar -b -L 0
(-43035) You must be user 'root' or 'informix' to run ON-Bar.

#informix and root user
informixgroupuser@Linux> onbar -b -L 0
$>
```

7.2.2 Permissions for backup executables for Windows

After an successful installation of IDS 11 distribution on Windows, you can check the permissions for the execution of onbar and ontape using Windows Explorer. Navigate to the bin directory of the IDS installation, right-click the file, and select **Properties**. In the Properties panel, you can check the actual settings under the Security tab.

In general, the execution permissions for onbar_d and ontape are given to everyone. If the current user can run the executable, the authentication is done in the executable itself. The requirement for a successful execution is that the current user is in the Informix-Admin group. An attempt to run the executables as a non-authorized user results an error, as shown in Example 7-3.

Example 7-3 Running on ontape on Windows as a user not in Informix-Admin group

```
C:\Program Files\IBM\IBM Informix Dynamic Server\11.10>ontape -a

Must be dbsa to run this program.

Program over.
```

7.2.3 Backup file permissions on UNIX

In case you back up your data to disk, either the ontape backup utility or the storage manager server is responsible for the permissions of the saved database objects. In the following section, we give an overview of the differences in handling backups with ontape to directories or file and the steps to find the backup files generated by the SM, and their default permissions.

onbar

Using onbar, the backed up data are sent to the storage manager through the dynamically loaded XBSA library. Depending on the storage definitions dedicated for the machine (here, the XBSA library) where the database server resides, the storage manager either creates a new file for this save set in the associated backup pool, or uses a large file as a common storage container for multiple backup objects differentiated by offsets. In most of the storage managers, the owner of the file is root. The group and other permissions can differ depending on which SM is used.

Example 7-4 shows how to determine where data are stored in the case of using onbar with the ISM storage manager. In this particular environment, we store each backup object in a separate file identified by a saveset ID. The saveset ID is maintained by the SM. The permissions for the file are very strict. The risk of having disallowed reads on the backup data is quite low.

Example 7-4 File permissions in the ISM

```
#identify the volume/directory relation ship
$ ism_show -devices
file disk v1 mounted on /data, write enabled
file disk v2 mounted on /logs, write enabled
#
```

```
#identify the volume/Pool relation ship
$ ism_show -volumes
   volume                        pool          flags written (%) expires
   v1                            ISMData              49 MB 100% 07/11/09
   v2                            ISMLogs              16 MB 100% 07/11/09

$> ls -al /data
insgesamt 2087
drwxr-xr-x  2 informix informix     128 2005-03-28 13:25 .
drwxr-xr-x  4 informix informix    1336 2005-05-03 18:52 ..
-rw-------  1 root     root     2097152 2005-03-28 13:25 774300415.0
-rw-------  1 root     root          47 2005-03-28 12:29 .nsr
-rw-------  1 root     root       65536 2005-03-28 12:30 volume
```

For the backup tests, we use Tivoli® Storage Manager Server 5.1, which comes with a Web-based administration console. To find the disk space for the backup client, perform these two steps:

1. Find the relationship between your client and the storage pool:
   a. Follow **client** → **ClientNodes** to find the policy domain for the client.
   b. Follow **PolicyDomains** → **PolicySet** → **ManagementClass** → **BackupCopyGroup**. Copy destination defines the storage pool.
2. Find the directory where the data are stored. Follow **ServerStorage** → **StoragePools** → **QueryStoragePoolVolumes**.

You get a list of the available storage pools and the container definition. Example 7-5 shows a sample output.

Example 7-5 Storage pools in TSM

```
Volume Name              Storage     Device      Estimated   Pct   Volume
                         Pool Name   Class Name   Capacity   Util  Status
                                                      (MB)
------------------------ ----------- ---------- ---------- -----  --------
/usr/tivoli/tsm/server/- SPACEMGPOOL DISK              8.0   0.0  On-Line
 bin/spcmgmt.dsm
/work3/tivoli/archivevo- ARCHIVEPOOL DISK          8,196.0   0.0  On-Line
 l1.dsm
/work3/tivoli/backupvol- BACKUPPOOL  DISK         12,288.0   3.5  On-Line
 1.dsm
```

ontape

In the case of using the ontape utility, this program is responsible for the file permission when the backup target is a directory. All backup files are created by the utility itself. The only prerequisite checked is the permissions on the directory that has to exist before the backup starts. The minimum requirement for the permission is rwxrwx---. The owner and the group of the directory have to be informix. Any backup files created by ontape in the target storage directory have 660 permission with the owner informix and group informix.

Example 7-6 shows the general permissions for backup into directories. Given that we do not assign any other user to the informix group, this access permission is also quite safe.

Example 7-6 Permissions for ontape into directory

```
holgerk@Linux:/tmp/dir> ls -al
insgesamt 24560
drwxrwxrwx   2 informix informix      328 .
drwxrwxrwt  52 root     root         9224 ..
-rw-rw----   1 informix informix 12648448 Linux_1_20070930_222603_L0
-rw-rw----   1 informix informix  1409558 Linux_1_20070930_222655_L0
-rw-rw----   1 informix informix  1013282 Linux_1_20070930_224334_L0
-rw-rw----   1 informix informix   992893 Linux_1_20070930_225347_L0
-rw-rw----   1 informix informix  9011200 Linux_1_L0
```

Any other permissions on the target backup directory lead to the error message shown in Example 7-7.

Example 7-7 Attempting to back up with insufficient permissions

```
$> ontape -s -L 0
Directory must have RWX permission for owner/group informix /backup/
Archive failed - function open archive tape failed code -1 errno 0

Program over.
```

In addition to backup to a directory, ontape also supports backing up to a file. The target file has to be created before the backup is started. Ontape does some basic permission checking for the existing file. The minimum required permissions are the group informix and the owner, and the group is able to read and write to the file.

You also can back up the data to a file that has read and write open for everyone, and the file owner is in the informix group. The owner is not required to be the informix user or root. DBA should pay attention to this loose security setting. Do not give any permissions to *others* here.

We have compiled some permission settings and the status returned by the ontape based on these settings in Example 7-8.

Example 7-8   ontape backups info a file and permission settings

```
#Case 1 owner informix but file is open for others
$> ls -al /backup/file
-rwxrwxrwx  1 informix informix 12648448 2007-09-30 23:05 /backup/file

$> ontape -s -L 0

Please mount tape 1 on /backup/file and press Return to continue ...
10 percent done.
100 percent done.

Please label this tape as number 1 in the arc tape sequence.
This tape contains the following logical logs:

 8

Program over.

#Case 2 Owner informix but no write permissions
$> ls -al /backup/file
-r--r--r-- 1 informix informix 0 2007-09-30 23:05 /backup/file

$> ontape -s -L 0

Please mount tape 1 on /backup/file and press Return to continue ...
Bad READ/WRITE permissions: '(mode 33060) /backup/file'.
could not write archive tape.
Please mount tape 1 on /backup/file and press Return to continue ...

#Case 3 Owner is not informix, Group is not informix

$> ls -al /backup/file
-rwxrwxrwx  1 holgerk users 12648448 2007-09-30 23:07 /backup/file

$> ontape -s -L 0

Please mount tape 1 on /backup/file and press Return to continue ...
Bad READ/WRITE permissions: '(owner/group) /backup/file'.
could not write archive tape.
Please mount tape 1 on /backup/file and press Return to continue ...
Interrupt received ...
```

```
#Case 4 Only the group is informix, but open for others
$> ls -al /backup/file
-rwxrwxrwx  1 holgerk informix 12648448 2007-09-30 23:07 /backup/file
$> ontape -s -L 0

Please mount tape 1 on /tmp/a and press Return to continue ...
10 percent done.
100 percent done.

Please label this tape as number 1 in the arc tape sequence.
This tape contains the following logical logs:

 8

Program over.
```

7.2.4 Adjust the backup file permissions on Windows

On Windows, all files created by the backup utilities onbar and ontape inherit the permissions from the parent folder's security options. The worst case scenario is that the parent folder has read and write access to all users defined on the local machine then the newly created backup files allow everyone to read and write them.

In order to define appropriate permissions for the files in your backup folder, you have to remove the inherit permissions option from the parent folder using the following steps:

1. Open Windows Explorer and go to the backup directory.
2. Right-click the directory name for which you want to change the permission.
3. Select **Sharing and Security** and go to the **Security** tab.
4. On the Security tab, click **Advanced**.
5. In the Advanced Security Setting panel, the inherit parent folder permission is enabled. Uncheck the box shown in Figure 7-1 to disable the function and apply the change.

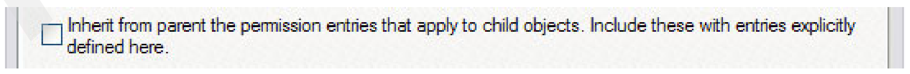

☐ Inherit from parent the permission entries that apply to child objects. Include these with entries explicitly defined here.

Figure 7-1 Inherit permission disabled

6. Now you are able to adjust the access permissions for this particular folder:
   a. Remove the permissions for the users.
   b. Specify each user or group that needs the access separately.

After the folder permission is changed, verify the new permissions by running a backup with ontape. In case you failed to give the user the appropriate permissions, the backup will report error messages, as shown in Example 7-9.

Example 7-9 Insufficient permissions for backup into file

```
C:\Program Files\IBM\IBM Informix Dynamic Server\11.10>ontape -s -L 0

Please mount tape 1 on c:\backup\a and press Return to continue ...
could not write archive tape.
Please mount tape 1 on c:\backup\a and press Return to continue ...
could not write archive tape.
Please mount tape 1 on c:\backup\a and press Return to continue ...
```

When adjusting the permissions, consider the following differences for using ontape:

- The ontape backup into a file requires only the write permission on the directory for the Informix-Admin group or the particular user belonging to the group.
- The ontape backup into a directory requires additional modify permission because the file generated by the archive is renamed at the end of a successful backup. If you do not have the modify permission, the backup reports an error, as shown in Example 7-10.

Example 7-10 Insufficient permissions on windows running ontape into a directory

```
C:\Program Files\IBM\IBM Informix Dynamic Server\11.10>ontape -s -L 0
10 percent done.
20 percent done.
30 percent done.
100 percent done.
Backup to directory error: Cannot rename file from
c:\backup\ibm_image_2_L0 to c
:\backup\ibm_image_2_20071005_101534_L0, errno = 13
Archive failed -
```

7.2.5 Permissions for backup log and control files

Since the ontape utility does not create a log or control file, we focus only on the onbar utility. Onbar writes the status information into the file specified by the BAR_ACT_LOG ONCONFIG parameter. Check the permissions and ownership of the files specified. The owner of this file has to be informix. Any other owner of the specified file causes onbar to return an error, as shown in Example 7-11.

Example 7-11 Bar_act.log file ownership and onbar

```
holgerk@Linux:/tmp> six onbar -b -w -L 0
(-43039) The ON-Bar log file must be owned by user 'informix'.
(-43079) Unable to open file /tmp/bar_act.log

holgerk@Linux:/tmp> ls -al /tmp/bar_act.log
-rwxrwxrwx  1 holgerk informix 8539 2007-10-03 22:46 /tmp/bar_act.log
```

The maintenance of the backed up objects with the relationship to backup time, saveset ID at the storage manager, and several other backup object related parameters is done by using ixbar.<servername> in the $INFORMIXDIR/etc directory. There are no requirements in view of permissions.

7.2.6 Database permissions and the onbar utility

In order to support warm restores of non-critical database spaces, the IDS database server creates, at initialization time, a database named sysutils. This database maintains all the objects backed up by the onbar utility including object names, backup time, saveset ID, and so on. The default permission has the root and informix user defined as the DBA of this database. This can be explained with the fact that, by default, root and informix are also the users allowed to run the onbar utility. Any other users have only the permission to connect to the database and select tables.

7.3 Backups over the network

Network traffic is another important backup-related security issue. Both backup utilities ontape and onbar are intended to run locally at the database server. However, both utilities are able to send the backed up data with the backup stream over the network to a remote server.

Ontape uses the **rsh** command to send the data to a remote tape or to a remote file, as shown in Example 7-12. Therefore, additional security-related configuration on the remote server may be required, for example, starting the rsh daemon is not a default setting on some UNIX environments like Linux. In addition, a trusted user connection to the remote host is necessary, which is associated with an entry in the .rhosts file on the remote machine for the informix user.

Example 7-12 ontape and a remote backup

```
$> ps -eaf | grep rsh
root     13471 13420  0 22:51 pts/3 rsh Linux dd of=/tmp/a ibs=32768 obs=32768
```

The onbar utility is intended to be used in an storage manager environment. Beside using with ISM storage manager, which is distributed with IDS and can only be used on the database server, this configuration most likely uses a dedicated remote machine for the storage manager server. This infrastructure requires network traffic between the storage manager client (mostly a third-party XBSA library dynamically loaded at run time by the onbar utility) and the storage manager server.

Without any configuration change for compression or encryption in either the particular storage manager environment or BACKUP_FILTER provided by IDS, the data sent are readable. This behavior opens a door for third-party readers to sniff the network for useful information. We discuss BACKUP_FILTER provided by IDS 11 in 7.6, "Basic security solutions with backup filters" on page 257.

To give you an idea of how easily a tool like tcpdump can be used to read the sent data, we used the tcpdump utility to read from the local loopback on a SuSE Linux machine during a backup with onbar. The significant difference in the output between an encrypted communication and a raw backup is shown in Example 7-13 and Example 7-14 on page 252. The complete statement used for generating the output was:

```
tcpdump -XX  -s 8192 -i lo
```

In Example 7-13, you can see the complete configuration page of the backup data and the root reserved pages from the rootdbs at the begin of the output.

Example 7-13 Output of an raw backup with tcpdump

Begin of the communication:

```
23:26:32.100477 IP Linux.site.48350 > Linux.site.pdap-np: P 1:362(361)
ack 1 win 8192 <nop,nop,timestamp 1526948 1526948>
   0x0000:  0000 0000 0000 0000 0000 0000 0800 4500  ..............E.
   0x0010:  019d 0370 4000 4006 37e7 7f00 0002 7f00  ...p@.@.7.......
   0x0020:  0002 bcde 05f6 6818 4fc9 683d 8f57 8018  ......h.O.h=.W..
   0x0030:  2000 ff93 0000 0101 080a 0017 4ca4 0017  ............L...
   0x0040:  4ca4 0169 013d 0000 0064 0065 0000 003d  L..i.=...d.e...=
   0x0050:  0006 4945 4545 4900 006c 6f6e 7461 7065  ..IEEEI..lontape
```

Begin of sending the backup data:

```
23:26:32.794800 IP Linux.site.8508 > Linux.site.7941: .
1005:17389(16384) ack 49 win 8180 <nop,nop,timestamp 1527122 1527122>
.....
   0x1970:  0000 0000 0000 524f 4f54 4e41 4d45 2072  ......ROOTNAME.r
   0x1980:  6f6f 7464 6273 0052 4f4f 5450 4154 4820  ootdbs.ROOTPATH.
   0x1990:  2f76 6f62 732f 4944 532f 6368 756e 6b73  /vobs/IDS/chunks
   0x19a0:  2f72 6f6f 7463 6875 6e6b 0052 4f4f 544f  /rootchunk.ROOTO
   0x19b0:  4646 5345 5420 3000 524f 4f54 5349 5a45  FFSET.0.ROOTSIZE
   0x19c0:  2032 3030 3030 3000 4d49 5252 4f52 2030  .200000.MIRROR.0
   0x19d0:  004d 4952 524f 5250 4154 4820 004d 4952  .MIRRORPATH..MIR
   0x19e0:  524f 524f 4646 5345 5420 3000 4442 5f4c  ROROFFSET.0.DB_L
   0x19f0:  4942 5241 5259 5f50 4154 4820 004a 5650  IBRARY_PATH..JVP
   0x1a00:  484f 4d45 202f 7573 722f 696e 666f 726d  HOME./usr/inform
   0x1a10:  6978 2f65 7874 656e 642f 6b72 616b 6174  ix/extend/krakat
   0x1a20:  6f61 004a 5650 4a41 5641 484f 4d45 202f  oa.JVPJAVAHOME./
```

We used openssl from Linux to encrypt the data with the *salt* option. With the salt option in addition to the provided password, an internal generated value is also used to encrypt the data, even though this option clearly indicated where the backup starts with the string `Salted`, as shown in Example 7-14. In general, it makes your data more secure with encryption. In case you use the same password all the time, this option adds a random factor in generating the encryption key.

Example 7-14 Output of an encrypted backup with tcpdump

Begin of the communication -- similar to the raw backup

```
23:35:51.804762 IP Linux.site.44876 > Linux.site.pdap-np: P 1:362(361)
ack 1 win 8192 <nop,nop,timestamp 1666866 1666865>
   0x0000:  0000 0000 0000 0000 0000 0000 0800 4500  ..............E.
   0x0010:  019d 260b 4000 4006 154c 7f00 0002 7f00  ..&.@.@..L......
   0x0020:  0002 af4c 05f6 8b5f 4937 8c0a 580b 8018  ...L..._I7..X...
   0x0030:  2000 ff93 0000 0101 080a 0019 6f32 0019  ............o2..
   0x0040:  6f31 0169 013d 0000 0064 0065 0000 003d  o1.i.=...d.e...=
   0x0050:  0006 4945 4545 4900 006c 6f6e 7461 7065  ..IEEEI..lontape
```

Begin of sending the backup data:

```
23:35:52.545421 IP Linux.site.8507 > Linux.site.7941: .
1005:17389(16384) ack 49 win 8180 <nop,nop,timestamp 1667051 1667051>
.....
   0x0150:  0000 0000 0000 0000 3400 0000 f800 5361  ........4.....Sa
   0x0160:  6c74 6564 5f5f 3e28 06c7 7640 6a3c e55e  lted__>(..v@j<.^
   0x0170:  855b c437 db29 b9eb 013d 3c02 1614 ccce  .[.7.)...=<.....
   0x0180:  2635 9c66 0ead 0f76 4d1d 94f4 2373 820c  &5.f...vM...#s..
   0x0190:  dd4b 0831 54ed 887a 532b a934 ff7a 22c8  .K.1T..zS+.4.z".
   0x01a0:  f101 33d6 ae59 757d 992d dbf0 19a9 c0e9  ..3..Yu}.-......
   0x01b0:  da20 f4df 23f4 1605 0f79 8403 a648 56bd  ....#....y...HV.
   0x01c0:  2b0a 06ab 5b19 4f6e 6f6f 4294 2b5c 9a70  +...[.OnooB.+\.p
   0x01d0:  65af ce56 ae5b 0af1 f033 6778 07ae cd84  e..V.[...3gx....
   0x01e0:  70f7 3618 76d8 09a1 aad9 8ce8 2d2a 9394  p.6.v.......-*..
   0x01f0:  ff2d a1f8 b6d7 fc81 fb01 f202 9b2a 5c85  .-...........*\.
   0x0200:  3e7c c96e 351a 5815 7b2b e0da ac49 87c0  >|.n5.X.{+...I..
   0x0210:  fbf9 cecc 7321 1f4a bbd3 a33b f64e 5887  ....s!.J...;.NX.
   0x0220:  4e0f 61cb 99fa b330 213d 9a74 956a eb73  N.a....0!=.t.j.s
   0x0230:  6e2b c304 9e12 330d fbd6 e5f6 374a 7116  n+....3.....7Jq.
   0x0240:  134d 14a0 578f 689f b28e 8786 0317 8699  .M..W.h.........
```

We used openssl from Linux to encrypt the data with the *salt* option. With the salt option, in addition to the provided password, an internal generated value is also used to encrypt the data. For Example 7-13 on page 251, the options make more clear where the backup starts. Look for the string `Salted`.

On the other hand, the use of the salt option is, in general, a good idea to make the encryption of your data more secure. It adds a random factor generating the encryption key in case you frequently use the same password.

7.4 Security features provided by storage managers

Backing up the dbspace objects from IDS with onbar gives the DBA the flexibility to integrate the existing file system database backup strategy with the company's favorite storage manager solution. The communication between onbar as a client and the storage manager server is implemented by the use of the XBSA standard.

When backing up using obar with the storage manager, as a good practice with the client server solution, the server should not reside on the same machine as the client. In addition to the request for disk or tape storage space for saving the database objects, reasonable network traffic and network utilization are also required.

In the previous section we discussed that sending the raw backup data over the network could cause a potential security problem since network sniffing can happen. The common storage managers provide two different solutions to help prevent this problem: compression and encryption. With compression, besides achieving a higher level of security, you also can expect a reduction of network utilization due to the XBSA client sending significantly less data.

In the following sections we introduce the setting requirements for a selected list of storage managers. Keep in mind that this list is not exhaustive. There are other SM solutions available.

Informix ISM

The Informix ISM Storage Manager Version 2.20 is shipped with IDS 11. This SM is a clone of an older Legato version with some limitations in terms of parallelization and remote backups. ISM provides an easy-to-use interface to enable and to check compression and encryption.

To enable the encryption, you have to specify the ISM_ENCRYPTION environment variable with the value TRUE in the onbar environment.

Compression can be enabled by setting the environment variable ISM_COMPRESSION to TRUE.

Example 7-15 shows a possible ISM environment variable setting. In this example, the compression is turned off.

Example 7-15 Possible settings for ISM related to compression and encryption

```
export ISM_ENCRYPTION=TRUE
export ISM_COMPRESSION=FALSE
```

Unfortunately, the onbar does not reflect the current settings with messages in the bar_act.log. A way to differentiate the backups with different setting is by comparing the size and the content of the consecutive backups files. Example 7-16 shows the differences of the storage manager objects backed up by raw, compressed, and encrypted. In the example environment the backups are made into a volume, maintaining the backup data in a directory. The files you can see are the backup objects. The file names are inherited by the saveset IDs, correlated to the ID in the ixbar file maintained by onbar.

Example 7-16 ls -al in a backup volume in ISM for different backup options

```
-rw-------  1 root     root     8683520 2007-10-03 23:09 774300423.0
-rw-------  1 root     root     8683520 2007-10-03 23:11 774300424.0
-rw-------  1 root     root     8519680 2007-10-03 23:13 774300425.0
-rw-------  1 root     root     8519680 2007-10-03 23:16 774300426.0
```

```
#Backups without any options
-rw-------  1 root     root     8519680 2007-10-03 23:17 774300427.0
-rw-------  1 root     root     8519680 2007-10-03 23:26 774300428.0
-rw-------  1 root     root     8617984 2007-10-03 23:35 774300429.0
#Backup with encryption
-rw-------  1 root     root     1933312 2007-10-04 17:14 774300430.0
-rw-------  1 root     root       32768 2007-10-04 17:15 774300431.0
-rw-------  1 root     root       32768 2007-10-04 17:15 774300432.0
-rw-------  1 root     root       65536 2007-10-04 17:15 774300433.0
#Backup with compression
-rw-------  1 root     root     1900544 2007-10-04 17:17 774300434.0
-rw-------  1 root     root       32768 2007-10-04 17:17 774300435.0
-rw-------  1 root     root       32768 2007-10-04 17:17 774300436.0
-rw-------  1 root     root       65536 2007-10-04 17:18 774300437.0
-rw-------  1 root     root          47 2005-03-28 12:29 .nsr
-rw-------  1 root     root       65536 2005-03-28 12:30 volume
```

Settings with values other than TRUE or FALSE are reported by onbar with an error in bar_act.log file, as shown in the Example 7-17.

Example 7-17 wrong settings for ISM environment variables and their errors

```
#export ISM_ENCRYPTION=XOR
#onbar -b -w -L 0

#tail /tmp/bar_act.log
2007-10-05 09:02:19 5656736  4247556 Begin level 0 backup rootdbs.
 2007-10-05 09:02:19 5656736  4247556 XBSA Error: (BSAInit) Invalid
environment keyword.
 2007-10-05 09:02:30 5656736  4247556 /sqldists/10.00.UC5W5/bin/onbar_d
complete , returning 12 (0x0c)
```

Legato storage manager

Similar to the Informix ISM, the Legato product requires the setting of the environment variables to enable compression and encryption. The variables have to be set in the onbar shell environment. To enable the particular feature, you need set the variables NSR_COMPRESSION and NSR_ENCRYPTION to YES.

Tivoli Storage Manager

All TSM clients with Version 5.3 or later API can support encryption. This includes the Tivoli Data Protection (TDP) library for IDS shipped with IDS 11. The TDP represents the XBSA library provided by TSM. It enables the onbar utility to exchange the data with the TSM server. This library is based on Tivoli API Version 5.3.2.

To enable the encryption, you have to change dsm.sys file, which can be located in the Tivoli installation directory. The following settings are required:

- Add:

  ```
  ENABLECLIENTCRYPTKEY YES
  ```

- If you want to have another encryption besides aes128 (which is des 56), add:

  ```
  ENCRYPTIONTYPE AES128
  ```

- Define which objects should be encrypted. For all objects, add:

  ```
  include.encrypt /.../*
  ```

Under Windows add your changes to the dsm.opt file instead of the dsm.sys file.

In order to enable compression for a TSM client, add to the following option to the dsm.opt or the dsm.sys file:

```
COMPRESSAlways Yes
```

shows a sample setting of the dsm.opt file on a UNIX system.

Example 7-18 dsm.opt settings for Informix TDP clients with TSM

```
***********************************************************************
*
* Tivoli Storage Manager
*
* Sample Client User Options file for AIX and SunOS (dsm.opt.smp)
*
***********************************************************************
*

*  This file contains an option you can use to specify the TSM
*  server to contact if more than one is defined in your client
*  system options file (dsm.sys).  Copy dsm.opt.smp to dsm.opt.
*  If you enter a server name for the option below, remove the
*  leading asterisk (*).

***********************************************************************
*

* SErvername       A server name defined in the dsm.sys file
SErvername    isar
```

```
ENCRYPTKEY SAVE
ENABLECLIENTCRYPTKEY YES
ENCRYPTIONTYPE AES128
include.encrypt /.../*
```

7.5 Compression and encryption on Windows

On Windows, the sharing and security option in the explorer program provides the ability to apply compression and encryption to the backup directory used by the storage manager or the ontape utility.

Perform the following steps to enable compression or encryption:

1. Open the explorer and go to the parent directory of the backup directory.
2. Right-click and select **Sharing and Security**.
3. Go to **General** tab.
4. Click **Advanced**.
5. Select either compression or encryption for your target directory
6. Apply the change also to the sub folders and files.

Note that enabling compression or encryption currently only works for backup to files in that particular directory. Using the backup to directory option with ontape is not supported. Trying to do that results in an error similar to a permission problem, as shown in Example 7-9 on page 249.

7.6 Basic security solutions with backup filters

The file permissions of the several backup solutions provided by the IDS database server secure the data backup to a certain extent. When backing up to a file or to a remote system using ontape, additional security might be necessary.

Even though the onbar utility provides, in combination with the third-party storage manager software, a stable and secure backup solution, the default plug-ins for compression and encryption provided by the storage manager server might not follow the standard of a company's security policy.

IDS 11 provides the ability to plug in a filter program in backup before the data is written to disk or given to the XBSA library to the storage manager. For restore, the filter program is invoked first to interpret the data to be restored. Common filter applications could be compress and uncompress programs or encryption and decryption solutions.

The advantage of using compressing software with backup is the reduction of the disk space used by the backup data. Using encryption software would significantly improve the security of production environment backup in view of preventing illegal scanning.

7.6.1 General implementation

IDS 11 introduces the new feature of defining backup and restore filters. This capability allows DBAs to plug in a program between the database server backup utility and the final storage target of the backup. In the case of using the ontape utility, the filter program is executed before the ontape program writes the data out to the disk or to the tape device. In a storage manager environment, the filter program is invoked before the data is sent out to the storage manager server by invocation of the XBSA library. Pipes are used for communication with the filter program.

With a special focus of security, the filter program can ensure complete encryption of the backup data and a data compression to reduce the disk space used by the backup files. Using a filer for compressing the backup data also reduces the network traffic in a client server based storage manager environment.

Even though the data encryption and the data compression are provided today by most of storage manager server providers, the ability to define your own security-oriented solutions provides you with much more flexibility to meet the company's policy requirements. In addition, it enhances the ontape functionality.

7.6.2 Configuration parameter

There are two new ONCONFIG file parameters for specifying the backup and the restore filters:

- BACKUP_FILTER
- RESTORE_FILTER

You have to specify the program name with the full path of the executable or ensure that the executable can be executed from the startup environment of IDS. If there are additional parameters required for running the filter program, the value of the ONCONFIG parameter has to be enclosed with single quotation marks.

Example 7-19 illustrates some sample settings for the BACKUP_FILTER parameter. The first two settings are related to a filter providing backup stream compression. The last sample setting provides data encryption with a password based on AES.

Example 7-19   BACKUP_FILTER settings

```
BACKUP_FILTER '/usr/bin/gzip'
BACKUP_FILTER 'gzip -9 '


BACKUP_FILTER 'openssl enc -e -aes-256-cbc -k password -salt'
```

The RESTORE_FILTER parameter should specify a filter to perform the reverse operation, as specified by the BACKUP_FILTER. For instance, if you use data encryption in the backup, you have to specify a program for decryption in the restore. In the case of restore, it does not make sense to specify a compression program.

Example 7-20 shows some common settings for the RESTORE_FILTER ONCONFIG parameter. The first two examples uncompress the restore stream from a backup that has used a gzip backup filter. The last example decrypts the backup stream with the specification of the same password, as provided with the backup filter.

Example 7-20   RESTORE_FILTER settings

```
RESTORE_FILTER '/usr/bin/gunzip'
RESTORE_FILTER 'gunzip'

RESTORE_FILTER 'openssl enc -d -aes-256-cbc -k password '
```

Changing the values of these particular ONCONFIG parameters does not require a database server restart. The new values are considered at the next startup of the database backup utility. We recommend that you carefully document the current settings to ensure a successful restore.

7.6.3 Verification of success

Before you establish the new backup filters in your production environment, you should verify the settings and check the behavior of the filter program. You can run the verification with the following steps:

1. Create a small test system with only one dbspace.
2. Set the filter values in your ONCONFIG file.

3. Back up the system with ontape or onbar to disk.
4. Check the backup files.
5. Bring down the test instance.
6. Try to restore the previous backup.

To have a more in-depth discussion of the verification process, here we demonstrate a simple setup using the openssl utility under Linux to encrypt the backup data to disk with ontape.

In a test instance, specify the following parameter in the ONCONFIG, as described in Example 7-21.

Example 7-21 ONCONFIG settings for encryption and decryption filter

```
BACKUP_FILTER 'openssl enc -e -aes-256-cbc -k password -salt'
RESTORE_FILTER 'openssl enc -d -aes-256-cbc -k password '
```

You can then back up your data with ontape or onbar. Example 7-22 shows backup with ontape and the output.

Example 7-22 Backup messages using ontape with backup filter

```
-bash-3.00$ ontape -s -L 0

Please mount tape 1 on /tmp/aa and press Return to continue ...
Using the backup and restore filter openssl enc -e -aes-256-cbc -k
holgerk -salt.
10 percent done.
20 percent done.
100 percent done.

Please label this tape as number 1 in the arc tape sequence.
This tape contains the following logical logs:

 458607

Program over.
```

If you want to set up the test in an onbar environment, onbar does not report the messages on the panel. You have to look into the message file specified by the BAR_ACT_LOG ONCONFIG parameter. In case of a successful invocation of the openssl utility, you should see the output shown in Example 7-23 in the file.

Example 7-23   Backup messages using onbar with an encryption filter

```
2007-10-01 15:45:20 2366  2364 Archive started on rootdbs, datadbs01,
datadbs03, datadbs09, datadbs10 (Requested Level 0).
 2007-10-01 15:45:20 2366  2364 Begin level 0 backup rootdbs.
 2007-10-01 15:45:20 2366  2364 Starting Filter openssl enc -e -aes-256-cbc -k
holgerk -salt.
 2007-10-01 15:45:20 2368  2366 Successfully connected to Storage Manager.
 2007-10-01 15:45:22 2368  2366 child process blocks read = 927
 2007-10-01 15:45:22 2368  2366 The child process for the backup and restore
filter is terminating with exit code 0.
 2007-10-01 15:45:22 2366  2364 Successfully connected to Storage Manager.
 2007-10-01 15:45:22 2366  2364 Completed level 0 backup rootdbs (Storage
Manager copy ID: 1938862761 0).
2007-10-01 15:45:23 2372  2366 Begin level 0 backup datadbs03.
 2007-10-01 15:45:23 2372  2366 Starting Filter openssl enc -e -aes-256-cbc -k
holgerk -salt.
 2007-10-01 15:45:23 2371  2366 Begin level 0 backup datadbs01.
 2007-10-01 15:45:23 2371  2366 Starting Filter openssl enc -e -aes-256-cbc -k
holgerk -salt.
 2007-10-01 15:45:23 2373  2366 Begin level 0 backup datadbs09.
 2007-10-01 15:45:23 2373  2366 Starting Filter openssl enc -e -aes-256-cbc -k
holgerk -salt.
 2007-10-01 15:45:23 2374  2366 Begin level 0 backup datadbs10.
 2007-10-01 15:45:23 2374  2366 Starting Filter openssl enc -e -aes-256-cbc -k
holgerk -salt.
```

To check further the success of the work of the backup filter, we verify the content of the backup file. In the traditional backup case, the backup file contains a one-to-one copy of the backed up data. This means that the majority portion of the file is readable with a utility like od -c. The difference is obvious when we compare the content of two backup files, as shown in Example 7-24. At the beginning of a backup file, a page with the current ONCONFIG settings is stored after the default header. In case of encryption, the output should not be readable. In the default backup case, it is readable.

Example 7-24   Content verification of the backup

```
Default backup content
od -x
0113760  \0  \0  \0  \0  \0  \0  \0  \0 030  \0 374  \0   j 375 003  \0
0114000 001  \0  \0  \0 001  \0 343   .   R  \0  \0 020 264 006  \0  \0
0114020  \0  \0  \0  \0  \0  \0  \0  \0   R   O   O   T   N   A   M   E
```

```
0114040       r   o   o   t   d   b   s  \0   R   O   O   T   P   A   T
0114060   H       /   v   o   b   s   /   I   D   S   /   c   h   u   n
0114100   k   s   /   r   o   o   t   c   h   u   n   k  \0   R   O   O
0114120   T   O   F   F   S   E   T       0  \0   R   O   O   T   S   I
0114140   Z   E       2   0   0   0   0   0  \0   M   I   R   R   O   R
0114160       0  \0   M   I   R   R   O   R   P   A   T   H      \0   M
0114200   I   R   R   O   R   O   F   F   S   E   T       0  \0   D   B
0114220   _   L   I   B   R   A   R   Y   _   P   A   T   H      \0   J
0114240   V   P   H   O   M   E       /   u   s   r   /   i   n   f   o
0114260   r   m   i   x   /   e   x   t   e   n   d   /   k   r   a   k
0114300   a   t   o   a  \0   J   V   P   J   A   V   A   H   O   M   E
0114320       /   u   s   r   /   i   n   f   o   r   m   i   x   /   e
0114340   x   t   e   n   d   /   k   r   a   k   a   t   o   a   /   j

Encoded backup content
0100000   S   a   l   t   e   d   _   _ 002   % 314 341 235   U 354 311
0100020   d 377   Y   j   > 252 307 205 031 021 261 255 036 326   v 371
0100040 003 023 343   V   <   A   % 365 317 324 324 252 332   8 377 227
0100060 244   $ 331   h   w 244 321 330   L   m 362 226 255 305   = 334
0100100   U 306   x   W 026 234 023 207 341 307 333   6   D 335 006 247
0100120 307 206 341 217 367   o   C 306 334   # 226 321   4 313 254   l
0100140 211   q 226 266 021   {   # 365 274   2 207   Q   u 234 217   C
0100160   : 271   } 375 276 334   Z   ?   u 200   z 221 374 237 034   j
0100200   m 265 353   P 326 231 314   i 200 360 215   H   P 274   l   O
0100220   B 237 206 337 211   . 273 202 345 210 031   |   6   D 300 365
0100240   O 341   6   B 262 033  \t 261 252 347   P 211   1 260 354 336
0100260 240 243   v 232 313   # 200   h 372 023   f   6 266 376 276   Z
0100300   = 276 230 324 216 264 021 026   I 334   D   & 255 246   8 272
0100320 366 304 027 020 312 351 320   X 303   B  \t 322   m   Q 036 367
```

If onbar is used, you also can perform this verification on the storage manager server. You have to identify the object by the saveset ID. With the ISM storage manager distributed with IDS, you can use the ism_show devices to identify the directory where the files are stored. Perform the following steps to identify the backup file in a ISM volume:

1. Check the location of the file, as shown in Example 7-25.

Example 7-25 Check file location

```
-bash-3.00$  ism_show -devices
file disk vol2 mounted on /home/informix/data, write enabled
file disk vol1 mounted on /home/informix/log, write enabled
```

2. Identify the saveset ID of the object from ixbar file. It is the seventh column in the file. See Example 7-26.

Example 7-26 Identify the saveset ID

```
$cat $INFORMIXDIR/etc/ixbar.0 | grep roodbs
holgerk          rootdbs          R  0 617   0 1938862761 0
2007-10-01 15:45:20 1     458607 -1876484643 617   0     - - 458607
1191246320  1191245836  1301
```

3. Find the backup file. See Example 7-27.

Example 7-27 Find the backup file

```
-bash-3.00$ ls -alt /home/informix/log/1938862761* | more
-rw-------  1 root informix 59244544 Oct  1 15:45
/home/informix/log/1938862761.0
```

Using environments with storage managers like Tivoli Storage Manager (TSM), the administrator is able to define disk storage volumes with containers. This means that the volume defines a single file to store multiple clients' backups or multiple backups from one client. In this configuration case, the verification is much more difficult comparing to the one-to-one relationship of saveset to file in ISM. Using temporary policy domains for the backup client is helpful for verification.

The last step of the verification is a restore of the recently backed up system. Example 7-28 shows an ontape restore.

Example 7-28 Restore with ontape using a restore filter

```
-bash-3.00$ ontape -r

Please mount tape 1 on /backup and press Return to continue ...

Archive Tape Information

Tape type:      Archive Backup Tape
Online version: IBM Informix Dynamic Server Version 11.10.FC1
Archive date:   Mon Oct  1 15:44:27 2007
User id:        informix
Terminal id:    /dev/pts/12
Archive level:  0
Tape device:    /backup
Tape blocksize (in k): 32
Tape size (in k): 1000240
Tape number in series: 1
```

```
Using the backup and restore filter openssl enc -d -aes-256-cbc -k
holgerk .

Spaces to restore:1 [rootdbs ]
2 [datadbs03 ]
3 [datadbs09 ]
4 [datadbs10 ]
5 [datadbs01 ]
```

7.6.4 Troubleshooting

In this section we show you the most common problems that you may encounter when using backup filters. We provide examples to give you some idea of what to consider during the setup and what messages to expect if the setup is incorrect.

Filter program does not exists or is not executable

Example 7-29 illustrates the errors of using a non-existing file to back up with ontape. Using a script without execution permissions results in the same error.

Example 7-29 Filter script does not work

```
$> ontape -s -L 0
Using the backup and restore filter /tmp/notexists .
Could not start filter: /tmp/notexists
Archive failed - Could not start filter: /tmp/notexists

ONCONFIG Settings
BACKUP_FILTER '/tmp/notexists '
#BACKUP_FILTER '/tmp/noexecutionperms '

Online log file
22:44:33  Level 0 Archive started on rootdbs
22:44:34  Archive on rootdbs ABORTED.
22:44:34  Aborted by client.
```

Filter program cannot read from a pipe

The filter program is talking with the backup utility through pipes. If you specify a program that is not able to read from a pipe or write to a pipe, the backup fails. See Example 7-30.

Example 7-30 Using a filter generating no output with ontape

```
-bash-3.00$ ontape -s -L 0

Please mount tape 1 on /backup and press Return to continue ...
Using the backup and restore filter /tmp/script_wo_pipes.
Write failed on parent's output pipe errno = 32
Archive failed - Write failed on parent's output pipe errno = 32



Online.log
16:40:04  Level 0 Archive started on rootdbs, datadbs03, datadbs09,
datadbs10, datadbs01
16:40:05  Archive on rootdbs, datadbs03, datadbs09, datadbs10,
datadbs01 ABORTED.
16:40:05  Aborted by client.
```

The output of the same backup using onbar is more descriptive and can be found in the bar_act.log, as shown in Example 7-31.

Example 7-31 Using a filter generating no output with ontape

```
2007-10-01 16:40:45 2848  2846 /sqldists/11FC1B7/bin/onbar_d -b -L 0
 2007-10-01 16:40:45 2848  2846 Archive started on rootdbs, datadbs01,
datadbs03, datadbs09, datadbs10 (Requested Level 0).
 2007-10-01 16:40:45 2848  2846 Begin level 0 backup rootdbs.
 2007-10-01 16:40:45 2848  2846 Starting Filter /tmp/bb.
 2007-10-01 16:40:45 2851  2848 Successfully connected to Storage
Manager.
  2848  2846 Filter terminated.
 2007-10-01 16:40:45 2848  2846 (-43082) Writing to backup and restore
filter failed with error -16843009.
 2007-10-01 16:40:45 2851  2848 child process blocks read = 1
 2007-10-01 16:40:45 2851  2848 The child process for the backup and
restore filter is terminating
with exit code 0.
```

Using multiple filter programs with pipes

A most likely common expectation to this feature is the possibility of chaining multiple filter programs with pipes, as on a shell. Unfortunately, this does not work when you specify the program chain in the ONCONFIG file.

Example 7-32 illustrates the error of combining openssl and gzip together. The output of the Linux strace clearly identifies the fail reason. The pipe was not used as a chain. It is treated as a parameter to the gzip executable.

Example 7-32 Chaining filter programs

```
$ONCONFIG file
BACKUP_FILTER openssl enc -e -aes-256-cbc -k holgerk -salt | gzip'
BACKUP_FILTER 'gzip -9 -c -f | gunzip | gzip -9 -f'


$ontape -s -L 0
Using the backup and restore filter gzip -9 -c -f \| gunzip \| gzip -9
-f.
gzip: |: No such file or directory
gzip: gunzip: No such file or directory
gzip: |: No such file or directory
gzip: gzip: No such file or directory


strace output

11375 1191185833.644422 execve("/vobs/IDS11/bin/gzip", ["gzip", "-9",
"-c", "-f", "\\|", "gunzip", "\\|", "gzip", "-9", "-f"], [/* 95 vars
*/]) = -1 ENOENT (No such file or directory)
```

Example 7-33 shows using a shell script as a work around to achieve the chaining.

Example 7-33 Use multiple filter programs together chained with pipes

```
BACKUP_FILTER 'script_with_pipes"

$cat script_with_pipes
openssl enc -e -aes-256-cbc -k holgerk -salt | gzip
```

Backup and restore filters are not compatible

In case the configuration was set up with mixed filter settings. Where the restore filter is not compatible with the backup filter, the database server is not able to restore any data. The database server returns the error messages shown in Example 7-34.

Example 7-34   Wrong restore filter used by ontape

```
holgerk@Linux:/tmp/dir> ontape -r
Restore file /tmp/dir/Linux_1_L0 and press Return to continue ...


Archive Tape Information

Tape type:      Archive Backup Tape
Online version: IBM Informix Dynamic Server Version 11.10.UC1
Archive date:   Sun Sep 30 22:26:55 2007
User id:        holgerk
Terminal id:    /dev/pts/4
Archive level:  0
Tape device:    /tmp/dir/
Tape blocksize (in k): 32
Tape size (in k): system defined for directory
Tape number in series: 1
Using the backup and restore filter gzip .
Physical restore failed - Read 16384 bytes from pipe


Program over.
```

A

Audit event mnemonics

This appendix provides a list of available audit event mnemonics.

Audit event mnemonic table

Table A-1 lists the available audit event mnemonics.

Table A-1 Audit event mnemonic

Mnemonic	Event
ACTB	Access Table
ADCK	Chunk, Add
ADLG	Transaction Log, Add
ALFR	Alter Fragment
ALIX	Index, Alter
ALLC	Security Label Component, Alter
ALME	Access Method, Alter
ALOC	Operator Class, Alter
ALOP	Optical Cluster, Alter
ALSQ	Sequence, Alter
ALTB	Table, Alter
BGTX	Transaction, Begin
CLDB	Database, Close
CMTX	Transaction, Commit
CRAG	Aggregate, Create
CRAM	Audit Mask, Create
CRBS	Storage Space, Create
CRBT	Opaque Type, Create
CRCT	Cast, Create
CRDB	Database, Create
CRDM	Domain, Create
CRDS	Dbspace, Create
CRDT	Distinct Type, Create
CRIX	Index, Create

Mnemonic	Event
CRLB	Security Label, Create
CRLC	Security Label Component, Create
CRME	Access Method, Create
CROC	Operator Class, Create
CROP	Optical Cluster, Create
CRPL	Security Policy, Create
CRPT	Encryption/Decryption
CRRL	Create Role
CRRT	Named Row Type, Create
CRSN	Synonym, Create
CRSP	SPL Routine, Create
CRSQ	Sequence, Create
CRTB	Table, Create
CRTR	Trigger, Create
CRVW	View, Create
CRXD	XADatasource, Create
CRXT	XADatasource Type, Create
DLRW	Row, Delete
DNCK	Chunk, Bring Off-line
DNDM	Disk Mirroring, Disable
DPPG	Display Page
DRAG	Aggregate, Drop
DRAM	Audit Mask, Delete
DRBS	Storage Space, Drop
DRCK	Chunk, Drop
DRCT	Cast, Drop
DRDB	Database, Drop

Mnemonic	Event
DRDM	Domain, Drop
DRDS	Dbspace, Drop
DRIX	Index, Drop
DRLB	Security Label, Drop
DRLC	Security Label Component, Drop
DRLG	Transaction Log, Drop
DRME	Access Method, Drop
DROC	Operator Class, Drop
DROP	Optical Cluster, Drop
DRPL	Security Policy, Drop
DRRL	Role, Drop
DRRT	Named Row Type, Drop
DRSN	Synonym, Drop
DRSP	SPL Routine, Drop
DRSQ	Sequence, Drop
DRTB	Table, Drop
DRTR	Trigger, Drop
DRTY	Type, Drop
DRVW	View, Drop
DRXD	XADatasource, Drop
DRXT	XADatasource Type, Drop
EXSP	SPL Routine, Execute
GRDB	Grant Database Access
GRDR	Grant Default Role
GRFR	Grant Fragment Access
GRLB	Grant Security Label
GRRL	Grant Role

Mnemonic	Event
GRSA	Grant DBSECADM
GRSS	Grant SETSESSIONAUTH
GRTB	Grant Table Access
GRXM	Grant Exemption
INRW	Row, Insert
LGDB	Database Log Mode, Change
LKTB	Table, Lock
LSAM	Audit Masks, List
LSDB	Databases, List
MDLG	Modify Transaction Logging
ONAU	onaudit
ONBR	onbar
ONCH	oncheck
ONIN	oninit
ONLG	onlog
ONLO	onload
ONMN	onmonitor
ONMO	onmode
ONPA	onparams
ONPL	onpload
ONSP	onspaces
ONST	onstat
ONTP	ontape
ONUL	onunload
OPDB	Database, Open
RDRW	Row, Read
RLOP	Optical Cluster, Release

Mnemonic	Event
RLTX	Transaction, Rollback
RMCK	Chunks, Clear Mirrored
RNDB	Rename Database
RNDS	Rename dbspace
RNIX	Rename index
RNLB	Security Label, Rename
RNLC	Security Label Component, Rename
RNPL	Security Policy, Rename
RNSQ	Sequence, Rename
RNTC	Table/ Column, Rename
RSOP	Optical Cluster, Reserve
RVDB	Revoke Database Access
RVDR	Revoke Default Role
RVFR	Revoke Fragment Access
RVLB	Revoke Security Label
RVRL	Revoke Role
RVSA	Revoke DBSECADM
RVSS	Revoke SETSESSIONAUTH
RVTB	Revoke Table Access
RVXM	Revoke Exemption
SCSP	SPL Routine, System Command
STCO	Collation®, Set
STCN	Constraint, Set
STDF	Set Debug File
STDP	Set Database Password
STDS	Set Dataskip
STEP	Set Encryption Password

Mnemonic	Event
STEV	Set Environment
STEX	Set Explain
STIL	Isolation Level, Set
STLM	Set Lock Mode
STNC	Set No Collation
STOM	Set Object Mode
STOP	Stop Statement
STPR	Set Pdqpriority
STRL	Set Role
STRS	Set Resident
STRT	Start Statement
STSA	Set Session Authorization
STSC	Set Statement Cache
STSN	Start New Session
STTX	Set Transaction Mode
SVXD	Save External Directives
TCTB	Truncate Table
TMOP	Optical Cluster, Time
ULTB	Table, Unlock
UPAM	Audit Mask, Update
UPCK	Chunk, Bring Online
UPDM	Disk Mirroring, Enable
UPRW	Row, Update Current
USSP	SPL Routine, Update Statistics
USTB	Table, Update Statistics

B

PAM API and macros

This appendix provides the header files required for a PAM implementation.

PAM_APPL.H and PAM_MODULE.H

Example B-1 lists the example header file PAM_APPL.H.

Example: B-1 PAM_APPL.H

```
#define  PAM_SUCCESS            0 /* Normal function return */
#define  PAM_OPEN_ERR           1 /* Failure in loading service
module*/
#define  PAM_SYMBOL_ERR         2 /* Symbol not found */
#define  PAM_SERVICE_ERR       3 /* Error in underlying service module
*/
#define  PAM_SYSTEM_ERR         4 /* System error */
#define  PAM_BUF_ERR            5 /* Memory buffer error */
#define  PAM_CONV_ERR           6 /* Conversation failure */
#define  PAM_PERM_DENIED        7 /* Permission denied */
#define  PAM_MAXTRIES           8 /* Maximum number of tries exceeded
*/
#define  PAM_AUTH_ERR           9 /* Authentication failure */
#define  PAM_NEW_AUTHTOK_REQD  10 /* Get new auth token from the user
*/
#define  PAM_CRED_INSUFFICIENT 11 /* can not access auth data b/c */
                              /* of insufficient credentials  */
#define  PAM_AUTHINFO_UNAVAIL  12 /* Can not retrieve auth information
*/
#define  PAM_USER_UNKNOWN      13 /* No account present for user */
#define  PAM_CRED_UNAVAIL      14 /* can not retrieve user credentials
*/
#define  PAM_CRED_EXPIRED      15 /* user credentials expired */
#define  PAM_CRED_ERR          16 /* failure setting user credentials
*/
#define  PAM_ACCT_EXPIRED      17 /* user account has expired */
#define  PAM_AUTHTOK_EXPIRED   18 /* Password expired and no longer */
#define  PAM_SESSION_ERR       19 /* can not make/remove entry for */
                              /* specified session */
#define  PAM_AUTHTOK_ERR           20 /* Authentication token */
                                  /* manipulation error */
#define  PAM_AUTHTOK_RECOVERY_ERR  21 /* Old authentication token */
                                  /* cannot be recovered */
#define  PAM_AUTHTOK_LOCK_BUSY     22 /* Authentication token */
                                  /* lock busy */
#define  PAM_AUTHTOK_DISABLE_AGING 23 /* Authentication token aging */
                                  /* is disabled */
#define  PAM_NO_MODULE_DATA        24 /* module data not found */
#define  PAM_IGNORE                25 /* ignore module */
```

```
#define  PAM_ABORT                 26 /* General PAM failure */
#define  PAM_TRY_AGAIN             27 /* Unable to update password */
                                      /* Try again another time */
#define  PAM_MODULE_UNKNOWN        28 /* Module unknown */
#define  PAM_DOMAIN_UNKNOWN        29 /* Domain unknown */

/*
 * structure pam_message is used to pass prompt, error message,
 * or any text information from scheme to application/user.
 */

struct pam_message {
    int msg_style;    /* Msg_style - see below */
    char *msg;        /* Message string */
};

/*
 * msg_style defines the interaction style between the
 * scheme and the application.
 */
#define  PAM_PROMPT_ECHO_OFF  1 /* Echo off when getting response */
#define  PAM_PROMPT_ECHO_ON   2 /* Echo on when getting response */
#define  PAM_ERROR_MSG        3 /* Error message */
#define  PAM_TEXT_INFO        4 /* Textual information */

/*
 * max # of messages passed to the application through the
 * conversation function call
 */
#define  PAM_MAX_NUM_MSG     32

/*
 * max size (in chars) of each messages passed to the application
 * through the conversation function call
 */
#define  PAM_MAX_MSG_SIZE   512

/*
 * max size (in chars) of each response passed from the application
 * through the conversation function call
 */
#define  PAM_MAX_RESP_SIZE  512

/*
 * structure pam_response is used by the scheme to get the user's
 * response back from the application/user.
 */

struct pam_response {
    char *resp;    /* Response string */
    int resp_retcode;    /* Return code - for future use */
};

/*
 * structure pam_conv is used by authentication applications for
 * passing call back function pointers and application data pointers
 * to the scheme
 */
struct pam_conv {
    int (*conv)(int, struct pam_message **,
    struct pam_response **, void *);
    void *appdata_ptr;   /* Application data ptr */
};

/* the pam handle */
typedef struct pam_handle pam_handle_t;

/*
 * pam_start() is called to initiate an authentication exchange
 * with PAM.
 */
extern int
pam_start(
    const char *service_name,        /* Service Name */
    const char *user,                /* User Name */
    const struct pam_conv *pam_conv, /* Conversation structure */
    pam_handle_t **pamh              /* Address to store handle */
);

/*
 * pam_end() is called to end an authentication exchange with PAM.
 */
extern int
pam_end(
    pam_handle_t *pamh,    /* handle from pam_start() */
    int status             /* the final status value that */
                           /* gets passed to cleanup functions */
);

/*
```

```
 * pam_set_item is called to store an object in PAM handle.
 */
extern int
pam_set_item(
    pam_handle_t *pamh,     /* PAM handle */
    int item_type,          /* Type of object - see below */
    const void *item        /* Address of place to put pointer */
                            /* to object */
);

/*
 * pam_get_item is called to retrieve an object from the static data
area
 */
extern int
pam_get_item(
    const pam_handle_t *pamh,   /* PAM handle */
    int item_type,              /* Type of object - see below */
    void              **item    /* Address of place to put pointer */
                                /* to object */
);

/* Items supported by pam_[sg]et_item() calls */
#define  PAM_SERVICE     1    /* The program/service name */
#define  PAM_USER        2    /* The user name */
#define  PAM_TTY         3    /* The tty name */
#define  PAM_RHOST       4    /* The remote host name */
#define  PAM_CONV        5    /* The conversation structure */
#define  PAM_AUTHTOK     6    /* The authentication token */
#define  PAM_OLDAUTHTOK  7    /* Old authentication token */
#define  PAM_RUSER       8    /* The remote user name */
#define  PAM_USER_PROMPT 9    /* The user prompt */

/*
 * pam_get_user is called to retrieve the user name (PAM_USER). If
 * PAM_USER is not set then this call will prompt for the user name
 * using the conversation function. This function should only be used
 * by modules, not applications.
 */

extern int
pam_get_user(
    pam_handle_t *pamh,     /* PAM handle */
    char **user,            /* User Name */
    const char *prompt      /* Prompt */
```

```
);

/*
 * pam_set_data is used to create module specific data, and
 * to optionally add a cleanup handler that gets called by pam_end.
 *
 */
extern int
pam_set_data(
    pam_handle_t *pamh,             /* PAM handle */
    const char *module_data_name,   /* unique module data name */
    const void *data,               /* the module specific data */
    void (*cleanup)(pam_handle_t *pamh, void *data, int pam_end_status)
);

/*
 * get module specific data set by pam_set_data.
 * returns PAM_NO_MODULE_DATA if specified module data was not found.
 */
extern int
pam_get_data(
    const pam_handle_t *pamh,
    const char *module_data_name,
    void **data
);

/*
 * PAM equivalent to strerror();
 */
extern char *
pam_strerror(
    pam_handle_t *pamh,     /* pam handle */
    int errnum              /* error number */
);

/* general flag for pam_* functions */
#define  PAM_SILENT    0x80000000

/*
 * pam_authenticate is called to authenticate the current user.
 */
extern int
pam_authenticate(
    pam_handle_t *pamh,
    int flags
```

```
);

/*
 * Flags for pam_authenticate
 */

#define  PAM_DISALLOW_NULL_AUTHTOK 0x1 /* The password must be
non-null*/

/*
 * pam_authenticate_secondary is called to authenticate the current
user
to a secondary domain.
 */
extern int
pam_authenticate_secondary (
    pam_handle_t * pamh,
    char * target_username,
    char * target_module_type,
    char * target_authn_domain,
    char * target_supp_data,
    unsigned char * target_module_authtok,
    int flags
);

/*
 * pam_acct_mgmt is called to perform account management processing
 */
extern int
pam_acct_mgmt(
    pam_handle_t *pamh,
    int flags
);

/*
 * pam_open_session is called to note the initiation of new session
 * in the appropriate administrative data bases.
 */
extern int
pam_open_session(
    pam_handle_t *pamh,
    int flags
);

/*
```

```
 * pam_close_session records the termination of a session.
 */
extern int
pam_close_session(
    pam_handle_t    *pamh,
    int              flags
);

/* pam_setcred is called to set the credentials of the current user */
extern int
pam_setcred(
    pam_handle_t *pamh,
    int flags
);

/* flags for pam_setcred() */
#define  PAM_ESTABLISH_CRED    0x1 /* set scheme specific user id */
#define  PAM_DELETE_CRED       0x2 /* unset scheme specific user id */
#define  PAM_REINITIALIZE_CRED 0x4 /* reinitialize user credentials */
                                   /* (after a password has changed */
#define  PAM_REFRESH_CRED      0x8 /* extend lifetime of credentials */

/* pam_chauthtok is called to change authentication token */

extern int
pam_chauthtok(
    pam_handle_t    *pamh,
    int              flags
);

/*
 * Be careful - there are flags defined for pam_sm_chauthtok() in
 * pam_modules.h also.
 */
#define  PAM_CHANGE_EXPIRED_AUTHTOK 0x4 /* update expired passwords
only */

/* pam_getenv is called to retrieve the value of a
 * PAM environment variable
 */

char* pam_getenv (
    pam_handle_t * pamh,
    const char * name,
);
```

```
/* pam_getenvlist is called to retrieve a list of alli
 * the PAM environment variables
 */

char** pam_getenvlist (
    pam_handle_t * pamh,
);

/* pam_putenv sets the value of a PAM environment variable */

int pam_putenv (
    pam_handle_t * pamh,
    const char * namevalue,
);

/*  pam_get_mapped_authok gets a mapped password for the user  */

int pam_get_mapped_authtok (
    pam_handle_t * pamh,
    const char * target_module_username, |
    const char * target_module_type, |
    const char * target_authn_domain, |
    size_t * target_authtok_len,
    unsigned char ** target_module_authtok |
);

/* pam_set_mapped_authtok stores a mapped password for
 * the username supplied
 */

int pam_set_mapped_authtok (
    pam_handle_t * pamh,
    char * target_module_username,
    size_t * target_authtok_len,
    unsigned char * target_module_authtok,
    char * target_module_type,
    char * target_authn_domain,
);

/* pam_get_mapped_username retrieves a valid matched identity
 * in a new domain
 */

int pam_get_mapped_username (
    pam_handle_t * pamh,
    const char * src_username,
    const char * src_module_type,
    const char * src_authn_domain,
    const char * target_module_type,
    const char * target_authn_domain,
    char ** target_module_username,
);

/* pam_set_mapped_username stores a mapped name for the
 * username supplied
 */

int pam_set_mapped_username (
    pam_handle_t * pamh,
    char * src_username, |
    char * src_module_type, |
    char * src_authn_domain, |
    char * target_module_username,
    char * target_module_type, |
    char * target_authn_domain, |
);

/* pam_get_user retrieves the current user name from a PAM handle */

int pam_get_user (
    pam_handle_t * pamh,
    char ** user,
    const char * prompt
);
```

Example B-2 lists the header file PAM_MODULE.H.

Example: B-2 PAM_MODULE.H

```
extern int
pam_sm_authenticate(
    pam_handle_t    *pamh,
    int              flags,
    int              argc,
    const char     **argv);

extern int
pam_sm_authenticate_secondary(
    pam_handle_t    *pamh,
    int              flags,
    int              argc,
    const char     **argv);

extern int
pam_sm_setcred(
    pam_handle_t    *pamh,
    int              flags,
    int              argc,
    const char     **argv);

extern int
pam_sm_acct_mgmt(
    pam_handle_t    *pamh,
    int              flags,
    int              argc,
    const char     **argv);

extern int
pam_sm_open_session(
    pam_handle_t    *pamh,
    int              flags,
    int              argc,
    const char     **argv);

extern int
pam_sm_close_session(
    pam_handle_t    *pamh,
    int              flags,
    int              argc,
    const char     **argv);

/*
 * Be careful - there are flags defined for pam_chauthtok() in
 * pam_appl.h also.
 */
#define  PAM_PRELIM_CHECK      0x1
#define  PAM_UPDATE_AUTHTOK    0x2

extern int
pam_sm_chauthtok(
    pam_handle_t    *pamh,
    int              flags,
    int              argc,
    const char     **argv);
```

```
extern int
pam_sm_get_mapped_authtok(
    pam_handle_t    *pamh,
    int              flags,
    int              argc,
    const char     **argv);

extern int
pam_sm_set_mapped_authtok(
    pam_handle_t    *pamh,
    int              flags,
    int              argc,
    const char     **argv);

extern int
pam_sm_get_mapped_username(
    pam_handle_t    *pamh,
    int              flags,
    int              argc,
    const char     **argv);

extern int
pam_sm_set_mapped_username(
    pam_handle_t    *pamh,
    int              flags,
    int              argc,
    const char     **argv);
```

Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this book.

IBM Redbooks

For information about ordering these publications, see "How to get Redbooks" on page 289. Note that some of the documents referenced here may be available in softcopy only.

- *Informix Dynamic Server 11: Advanced Functionality for Modern Business*, SG24-7465

Other publications

These publications are also relevant as further information sources:

- IBM white paper, Label-Based Access Control, by Dave Desautels and Lynette Adayilamuriyil
- *Guide to SQL: Syntax,* G251-228
- *IBM Informix Administration Reference Guide*, G251-2268-01

How to get Redbooks

You can search for, view, or download Redbooks, Redpapers, Technotes, draft publications and Additional materials, as well as order hardcopy Redbooks, at this Web site:

`ibm.com/redbooks`

Help from IBM

IBM Support and downloads

`ibm.com/support`

IBM Global Services

ibm.com/services

Index

IBM

Redbooks

Security and Compliance Solutions for IBM Informix Dynamic Server

Security and Compliance Solutions for IBM Informix Dynamic Server

Understand IDS security concepts and technologies

Learn security implementation through examples

Protect your data with IDS security features

In this IBM Redbooks publication, we discuss, in detail, the security features available in IBM Informix Database Server (IDS). These enriched IDS security features provide you with the capability to protect your data and comply with regulatory requirements.

We discuss how IDS integrates with operating system security functions for user authentication and user permissions. The role separation divides the security duty among administrators. Auditing enables the database server to log sensitive operations performed by users and administrators for analysis and identifying system misuses.

Discretionary access control (DAC) is the primary access control mechanism that enables access to SQL objects using privileges and roles. Using label-based access control (LBAC), you can control read and write access of users to individual rows and columns at the table level. We then discuss how to secure server-server and server-client communication in an IDS environment, as well as address the security issues for backup and restore.

SG24-7556-00 ISBN 0738485721