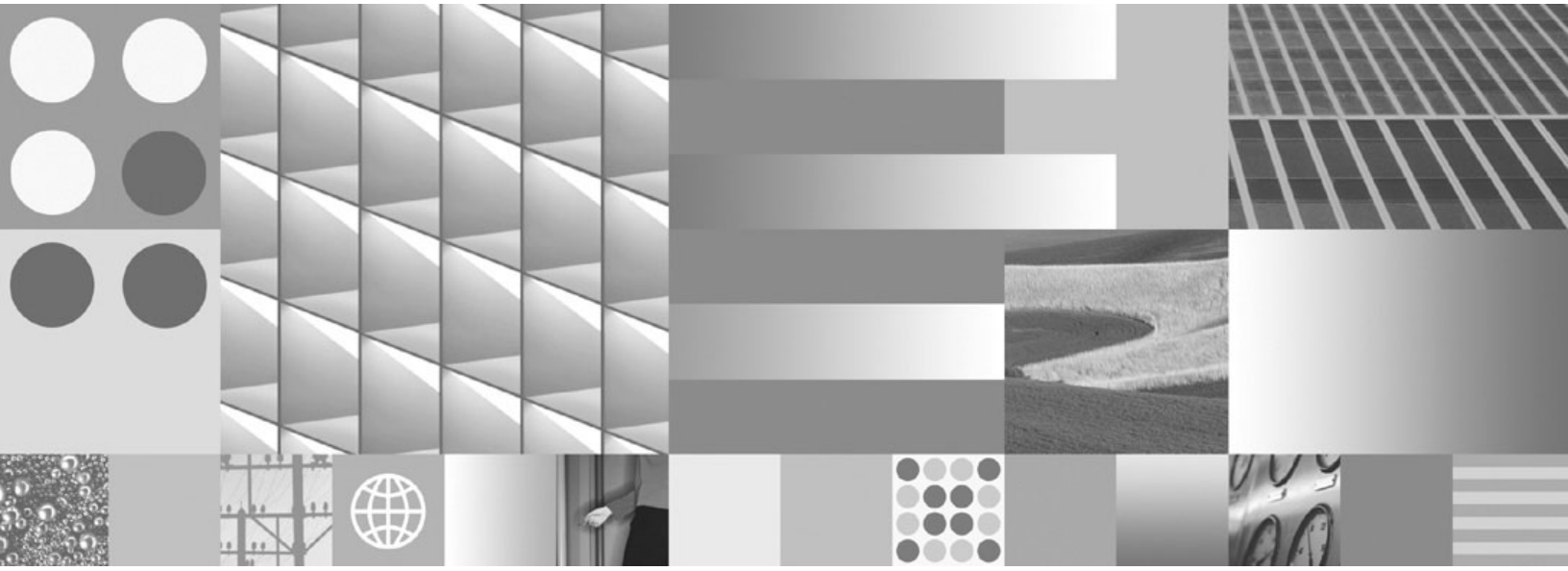
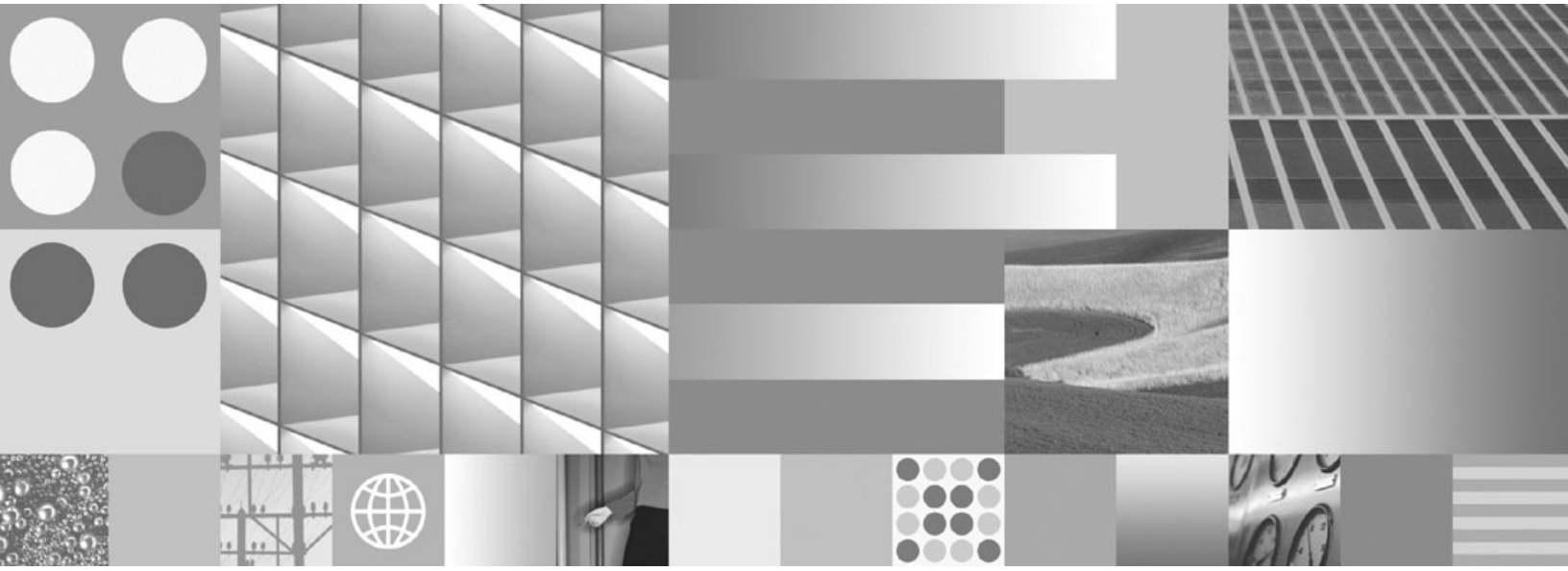
DB2 Version 9.5
for Linux, UNIX, and Windows



Database Security Guide

SC23-5850-00

DB2 Version 9.5
for Linux, UNIX, and Windows



Database Security Guide

SC23-5850-00

Note

Before using this information and the product it supports, read the general information under Appendix B, "Notices," on page 255.

Edition Notice

You can order IBM publications online or through your local IBM representative.

- To order publications online, go to the IBM Publications Center at www.ibm.com/shop/publications/order
- To find your local IBM representative, go to the IBM Directory of Worldwide Contacts at www.ibm.com/planetwide

To order DB2 publications from DB2 Marketing and Sales in the United States or Canada, call 1-800-IBM-4YOU (426-4968).

When you send information to IBM, you grant IBM a nonexclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

Contents

About this book

The Database Security Guide describes how to use DB2® security features to implement and manage the level of security you require for your database installation.

The Database Security Guide provides detailed information about:

- Managing the authentication of users who can access DB2 databases
- Setting up authorization to control user access to database objects and data

Chapter 1. DB2 security model

Two modes of security control access to the DB2 database system data and functions. Access to the DB2 database system is managed by facilities that reside outside the DB2 database system (authentication), whereas access within the DB2 database system is managed by the database manager (authorization).

Authentication

Authentication is the process by which a system verifies a user's identity. User authentication is completed by a security facility outside the DB2 database system, through an authentication security plug-in module. A default authentication security plug-in module that relies on operating-system-based authentication is included when you install the DB2 database system. To provide greater flexibility in accommodating your specific authentication needs, you can build your own authentication security plug-in module.

The authentication process produces a DB2 authorization ID. Group membership information for the user is also acquired during authentication. Default acquisition of group information relies on an operating-system based group-membership plug-in module that is included when you install the DB2 database system. If you prefer, you can acquire group membership information by using a specific group-membership plug-in module, such as lightweight directory access protocol (LDAP).

Authorization

After a user is authenticated, the database manager determines if that user is allowed to access DB2 data or resources. Authorization is the process whereby the DB2 database manager obtains information about the authenticated user, indicating which database operations that user can perform, and which data objects that user can access.

The different sources of permissions available to an authorization ID are as follows:

1. Primary permissions: those granted to the authorization ID directly.
2. Secondary permissions: those granted to the groups and roles in which the authorization ID is a member.
3. Public permissions: those granted to PUBLIC.
4. Context-sensitive permissions: those granted to a trusted context role.

Authorization can be given to users in the following categories:

- System-level authorization

  The system administrator (SYSADM), system control (SYSCTRL), system maintenance (SYSMAINT), and system monitor (SYSMON) authorities provide varying degrees of control over instance-level functions. Authorities provide a way both to group privileges and to control maintenance and utility operations for instances, databases, and database objects.
- Database-level authorization

The security administrator (SECADM), and database administrator (DBADM) authorities provide control within the database. Other database authorities include LOAD (ability to load data into a table), and CONNECT (ability to connect to a database).

- Object-level authorization

  bject level authorization involves checking privileges when an operation is performed on an object. For example, to select from a table a user must have SELECT privilege on a table (as a minimum).

- Content-based authorization

  Views provide a way to control which columns or rows of a table specific users can read. Label-based access control (LBAC) determines which users have read and write access to individual rows and individual columns.

You can use these features, in conjunction with the DB2 audit facility for monitoring access, to define and manage the level of security your database installation requires.

Authentication

Authentication of a user is completed using a security facility outside of the DB2 database system. The security facility can be part of the operating system or a separate product.

The security facility requires two items to authenticate a user: a user ID and a password. The user ID identifies the user to the security facility. By supplying the correct password, information known only to the user and the security facility, the user's identity (corresponding to the user ID) is verified.

Note: In non-root installations, operating system-based authentication must be enabled by running the db2rfe command.

After being authenticated:

- The user must be identified to DB2 using an SQL authorization name or *authid*. This name can be the same as the user ID, or a mapped value. For example, on UNIX® operating systems, when you are using the default security plug-in module, a DB2 *authid* is derived by transforming to uppercase letters a UNIX user ID that follows DB2 naming conventions.
- A list of groups to which the user belongs is obtained. Group membership may be used when authorizing the user. Groups are security facility entities that must also map to DB2 authorization names. This mapping is done in a method similar to that used for user IDs.

The DB2 database manager uses the security facility to authenticate users in one of two ways:

- A successful security system login is used as evidence of identity, and allows:
  - Use of local commands to access local data
  - Use of remote connections when the server trusts the client authentication.
- Successful validation of a user ID and password by the security facility is used as evidence of identity and allows:
  - Use of remote connections where the server requires proof of authentication
  - Use of operations where the user wants to run a command under an identity other than the identity used for login.

Note: On some UNIX systems, the DB2 database manager can log failed password attempts with the operating system, and detect when a client has exceeded the number of allowable login tries, as specified by the LOGINRETRIES parameter.

Authorization

Authorization is performed using DB2 facilities. DB2 tables and configuration files are used to record the permissions associated with each authorization name.

When an authenticated user tries to access data, the authorization name of the user, those of groups to which the user belongs, and those of roles granted to the user directly or indirectly through a group or a role, are compared with the recorded permissions. Based on this comparison, the DB2 server decides whether to allow the requested access.

The types of permissions recorded are privileges, authority levels, and LBAC credentials.

A *privilege* defines a single permission for an authorization name, enabling a user to create or access database resources. Privileges are stored in the database catalogs.

Authority levels provide a method of grouping privileges and control over higher-level database manager maintenance and utility operations. Database-specific authorities are stored in the database catalogs; system authorities are associated with group membership, and the group names that are associated with the authority levels are stored in the database manager configuration file for a given instance.

LBAC credentials are LBAC security labels and LBAC rule exemptions that allow access to data protected by label-based access control (LBAC). LBAC credentials are stored in the database catalogs.

Groups provide a convenient means of performing authorization for a collection of users without having to grant or revoke privileges for each user individually. Unless otherwise specified, group authorization names can be used anywhere that authorization names are used for authorization purposes. In general, group membership is considered for dynamic SQL and non-database object authorizations (such as instance level commands and utilities), but is not considered for static SQL. The exception to this general case occurs when privileges are granted to PUBLIC: these are considered when static SQL is processed. Specific cases where group membership does not apply are noted throughout the DB2 documentation, where applicable.

A role is a database object that groups together one or more privileges and can be assigned to users, groups, PUBLIC, or other roles by using a GRANT statement or to a trusted context by using a CREATE TRUSTED CONTEXT or ALTER TRUSTED CONTEXT statement. A role can be specified for the SESSION_USER ROLE connection attribute in a workload definition. When you use roles, you associate access permissions on database objects with the roles. Users that are members of those roles then have the privileges defined for the role with which to access database objects.

Roles provide similar functionality as groups; they perform authorization for a collection of users without having to grant or revoke privileges for each user individually. One advantage of roles is that they are managed by the DB2 database

system. The permissions granted to roles are taken into consideration during the authorization process for views, triggers, materialized query tables (MQTs), packages and SQL routines, unlike the permissions granted to groups. Permissions granted to groups are not considered during the authorization process for views, triggers, MQTs, packages and SQL routines, because the DB2 database system cannot discover when membership in a group changes, and so it cannot invalidate the objects, above, if appropriate.

Note: Permissions granted to roles that are granted to groups are not considered during the authorization process for views, triggers, MQTs, packages and SQL routines.

During an SQL statement processing, the permissions that the DB2 authorization model considers are the union of the following permissions:

1. The permissions granted to the primary authorization ID associated with the SQL statement
2. The permissions granted to the roles granted to the primary authorization ID associated with the SQL statement
3. The permissions granted to the secondary authorization IDs (groups or roles) associated with the SQL statement
4. The permissions granted to the roles granted to the secondary authorization IDs (groups or roles) associated with the SQL statement
5. The permissions granted to PUBLIC, including roles that are granted to PUBLIC, directly or indirectly through other roles.
6. The permissions granted to the trusted context role, if applicable.

Security considerations when installing and using the DB2 database manager

Security considerations are important to the DB2 administrator from the moment the product is installed.

To complete the installation of the DB2 database manager, a user ID, a group name, and a password are required. The GUI-based DB2 database manager install program creates default values for different user IDs and the group. Different defaults are created, depending on whether you are installing on Linux and UNIX or Windows® platforms:

- On UNIX and Linux® platforms, if you choose to create a DB2 instance in the instance setup window, the DB2 database install program creates, by default, different users for the DAS (`dasusr`), the instance owner (`db2inst`), and the fenced user (`db2fenc`). Optionally, you can specify different user names

  The DB2 database install program appends a number from 1-99 to the default user name, until a user ID that does not already exist can be created. For example, if the users `db2inst1` and `db2inst2` already exist, the DB2 database install program creates the user `db2inst3`. If a number greater than 10 is used, the character portion of the name is truncated in the default user ID. For example, if the user ID `db2fenc9` already exists, the DB2 database install program truncates the `c` in the user ID, then appends the `10` (`db2fen10`). Truncation does not occur when the numeric value is appended to the default DAS user (for example, `dasusr24`).
- On Windows platforms, the DB2 database install program creates, by default, the user `db2admin` for the DAS user, the instance owner, and fenced users (you can

specify a different user name during setup, if you want). Unlike Linux and UNIX platforms, no numeric value is appended to the user ID.

To minimize the risk of a user other than the administrator from learning of the defaults and using them in an improper fashion within databases and instances, change the defaults during the install to a new or existing user ID of your choice.

Note: Response file installations do not use default values for user IDs or group names. These values must be specified in the response file.

Passwords are very important when authenticating users. If no authentication requirements are set at the operating system level and the database is using the operating system to authenticate users, users will be allowed to connect. For example on Linux and UNIX operating systems, undefined passwords are treated as NULL. In this situation, any user without a defined password will be considered to have a NULL password. From the operating system's perspective, this is a match and the user is validated and able to connect to the database. Use passwords at the operating system level if you want the operating system to do the authentication of users for your database.

When working with DB2 Data Partitioning Feature (DPF) on Linux and UNIX operating system environments, the DB2 database manager by default uses the rsh utility (remsh on HP-UX) to run some commands on remote nodes. The rsh utility transmits passwords in clear text over the network, which can be a security exposure if the DB2 server is not on a secure network. You can use the DB2RSHCMD registry variable to set the remote shell program to a more secure alternative that avoids this exposure. One example of a more secure alternative is ssh. See the DB2RSHCMD registry variable documentation for restrictions on remote shell configurations.

After installing the DB2 database manager, also review, and change (if required), the default privileges that have been granted to users. By default, the installation process grants system administration (SYSADM) privileges to the following users on each operating system:

Windows environments
: A valid DB2 database user name that belongs to the Administrators group.

Linux and UNIX platforms
: A valid DB2 database user name that belongs to the primary group of the instance owner.

SYSADM privileges are the most powerful set of privileges available within the DB2 database manager. As a result, you might not want all of these users to have SYSADM privileges by default. The DB2 database manager provides the administrator with the ability to grant and revoke privileges to groups and individual user IDs.

By updating the database manager configuration parameter *sysadm_group*, the administrator can control which group of users possesses SYSADM privileges. You must follow the guidelines below to complete the security requirements for both the DB2 database installation and the subsequent instance and database creation.

Any group defined as the system administration group (by updating *sysadm_group*) must exist. The name of this group should allow for easy identification as the group created for instance owners. User IDs and groups that belong to this group have system administrator authority for their respective instances.

The administrator should consider creating an instance owner user ID that is easily recognized as being associated with a particular instance. This user ID should have as one of its groups the name of the SYSADM group created above. Another recommendation is to use this instance-owner user ID only as a member of the instance owner group and not to use it in any other group. This should control the proliferation of user IDs and groups that can modify the instance, or any object within the instance.

The created user ID must be associated with a password to provide authentication before being permitted entry into the data and databases within the instance. The recommendation when creating a password is to follow your organization's password naming guidelines.

Note: To avoid accidentally deleting or overwriting instance configuration or other files, administrators should consider using another user account, which does not belong to the same primary group as the instance owner, for day-to-day administration tasks that are performed on the server directly.

File permission requirements for the instance and database directories

The DB2 database system requires that your instance and database directories have at least the following permissions.

Note: When the instance and database directories are created by the DB2 database manager, the permissions are accurate and should not be changed.

The minimum permissions of the instance directory and the NODE000*x*/sqldbdir directory on UNIX and Linux machines must be: `u=rwx` and `go=rx`. The meaning of the letters is explained in the following table:

Character	Represents:
u	User (owner)
g	Group
o	Other users
r	Read
w	Write
x	Execute

For example, the permissions for the instance, db2inst1, in /home are:

```
drwxr-xr-x  36 db2inst1  db2grp1         4096 Jun 15 11:13 db2inst1
```

For the directories containing the databases, each and every directory level up to and including NODE000*x* needs the following permissions:

```
drwxrwxr-x  11 db2inst1  db2grp1         4096 Jun 14 15:53 NODE0000/
```

For example, if a database is located in /db2/data/db2inst1/db2inst1/NODE0000 then the directories: /db2, /db2/data, /db2/data/db2inst1, /db2/data/db2inst1/db2inst1 and /db2/data/db2inst1/db2inst1/NODE0000 need `drwxrwxr-x`.

Within the NODE000x directory, the sqldbdir directory requires the permissions `drwxrwxr-x`, for example:

```
drwx------   5 db2inst1  db2grp1         256 Jun 14 14:17 SAMPLE/
drwxr-x---   7 db2inst1  db2grp1        4096 Jun 14 13:26 SQL00001/
drwxrwxr-x   2 db2inst1  db2grp1         256 Jun 14 13:02 sqldbdir/
```

CAUTION:
To maintain the security of your files, do not change the permissions on the *DBNAME* directories (such as SAMPLE) and the SQL*xxxx* directories from the permissions they are assigned when the DB2 database manager creates them.

Authentication details

Authentication methods for your server

Access to an instance or a database first requires that the user be *authenticated*. The *authentication type* for each instance determines how and where a user will be verified. The authentication type is stored in the configuration file at the server. It is initially set when the instance is created. There is one authentication type per instance, which covers access to that database server and all the databases under its control.

If you intend to access data sources from a federated database, you must consider data source authentication processing and definitions for federated authentication types.

Note: You can check the following web site for certification information on the cryptographic routines used by the DB2 database management system to perform encryption of the userid and password when using SERVER_ENCRYPT authentication, and of the userid, password and user data when using DATA_ENCRYPT authentication: http://www.ibm.com/security/standards/st_evaluations.shtml.

Switching User on an Explicit Trusted Connection

For CLI/ODBC and XA CLI/ODBC applications, the authentication mechanism used when processing a switch user request that requires authentication is the same as the mechanism used to originally establish the trusted connection itself. Therefore, any other negotiated security attributes (for example, encryption algorithm, encryption keys, and plug-in names) used during the establishment of the explicit trusted connection are assumed to be the same for any authentication required for a switch user request on that trusted connection. JAVA applications allow the authentication method to be changed on a switch user request (by use of a datasource property).

Because a trusted context object can be defined such that switching user on a trusted connection does *not* require authentication, in order to take full advantage of the switch user on an explicit trusted connection feature, user-written security plug-ins must be able to:

- Accept a user ID-only token
- Return a valid DB2 authorization ID for that user ID

Note: An explicit trusted connection cannot be established if the CLIENT type of authentication is in effect.

Authentication types provided

The following authentication types are provided:

SERVER

Specifies that authentication occurs on the server through the security mechanism in effect for that configuration, for example, through a security plug-in module. The default security mechanism is that if a user ID and password are specified during the connection or attachment attempt, they are compared to the valid user ID and password combinations at the server to determine if the user is permitted to access the instance.

Note: The server code detects whether a connection is local or remote. For local connections, when authentication is SERVER, a user ID and password are not required for authentication to be successful.

SERVER_ENCRYPT

Specifies that the server accepts encrypted SERVER authentication schemes. If the client authentication is not specified, the client is authenticated using the method selected at the server.

CLIENT

Specifies that authentication occurs on the database partition where the application is invoked using operating system security. The user ID and password specified during a connection or attachment attempt are compared with the valid user ID and password combinations on the client node to determine if the user ID is permitted access to the instance. No further authentication will take place on the database server. This is sometimes called single signon.

If the user performs a local or client login, the user is known only to that local client workstation.

If the remote instance has CLIENT authentication, two other parameters determine the final authentication type: *trust_allclnts* and *trust_clntauth*.

CLIENT level security for TRUSTED clients only:

Trusted clients are clients that have a reliable, local security system.

When the authentication type of CLIENT has been selected, an additional option may be selected to protect against clients whose operating environment has no inherent security.

To protect against unsecured clients, the administrator can select Trusted Client Authentication by setting the *trust_allclnts* parameter to NO. This implies that all trusted platforms can authenticate the user on behalf of the server. Untrusted clients are authenticated on the Server and must provide a user ID and password. You use the *trust_allclnts* configuration parameter to indicate whether you are trusting clients. The default for this parameter is YES.

Note: It is possible to trust all clients (*trust_allclnts* is YES) yet have some of those clients as those who do not have a native safe security system for authentication.

You may also want to complete authentication at the server even for trusted clients. To indicate where to validate trusted clients, you use the *trust_clntauth* configuration parameter. The default for this parameter is CLIENT.

Note: For trusted clients only, if no user ID or password is explicitly provided when attempting to CONNECT or ATTACH, then validation of

the user takes place at the client. The *trust_clntauth* parameter is only used to determine where to validate the information provided on the USER or USING clauses.

To protect against all clients except DRDA® clients from DB2 on OS/390® and z/OS®, DB2 on VM and VSE, and DB2 on System i™, set the *trust_allclnts* parameter to DRDAONLY. Only these clients can be trusted to perform client-side authentication. All other clients must provide a user ID and password to be authenticated by the server.

The *trust_clntauth* parameter is used to determine where the above clients are authenticated: if *trust_clntauth* is "client", authentication takes place at the client. If *trust_clntauth* is "server", authentication takes place at the client when no user ID and password are provided and at the server when a user ID and password are provided.

Table 1. Authentication Modes using TRUST_ALLCLNTS and TRUST_CLNTAUTH Parameter Combinations.

TRUST_ ALLCLNTS	TRUST_ CLNTAUTH	Untrusted non–DRDA Client Authen-tication (no user ID & password)	Untrusted non–DRDA Client Authen-tication (with user ID & password)	Trusted non–DRDA Client Authen-tication (no user ID & password)	Trusted non–DRDA Client Authen-tication (with user ID & password)	DRDA Client Authen-tication (no user ID & password)	DRDA Client Authen-tication (with user ID & password)
YES	CLIENT	CLIENT	CLIENT	CLIENT	CLIENT	CLIENT	CLIENT
YES	SERVER	CLIENT	SERVER	CLIENT	SERVER	CLIENT	SERVER
NO	CLIENT	SERVER	SERVER	CLIENT	CLIENT	CLIENT	CLIENT
NO	SERVER	SERVER	SERVER	CLIENT	SERVER	CLIENT	SERVER
DRDAONLY	CLIENT	SERVER	SERVER	SERVER	SERVER	CLIENT	CLIENT
DRDAONLY	SERVER	SERVER	SERVER	SERVER	SERVER	CLIENT	SERVER

KERBEROS

Used when both the DB2 client and server are on operating systems that support the Kerberos security protocol. The Kerberos security protocol performs authentication as a third party authentication service by using conventional cryptography to create a shared secret key. This key becomes a user's credential and is used to verify the identity of users during all occasions when local or network services are requested. The key eliminates the need to pass the user name and password across the network as clear text. Using the Kerberos security protocol enables the use of a single sign-on to a remote DB2 database server. The KERBEROS authentication type is supported on various operating systems, refer to the related information section for more information.

Kerberos authentication works as follows:

1. A user logging on to the client machine using a domain account authenticates to the Kerberos key distribution center (KDC) at the domain controller. The key distribution center issues a ticket-granting ticket (TGT) to the client.
2. During the first phase of the connection the server sends the target principal name, which is the service account name for the DB2 database server service, to the client. Using the server's target principal name and the target-granting ticket, the client requests a service ticket from

the ticket-granting service (TGS) which also resides at the domain controller. If both the client's ticket-granting ticket and the server's target principal name are valid, the TGS issues a service ticket to the client. The principal name recorded in the database directory may be specified as name/instance@REALM. (This is in addition to DOMAIN\userID and userID@xxx.xxx.xxx.com formats accepted on Windows.)

3. The client sends this service ticket to the server using the communication channel (which may be, as an example, TCP/IP).
4. The server validates the client's server ticket. If the client's service ticket is valid, then the authentication is completed.

It is possible to catalog the databases on the client machine and explicitly specify the Kerberos authentication type with the server's target principal name. In this way, the first phase of the connection can be bypassed.

If a user ID and a password are specified, the client will request the ticket-granting ticket for that user account and use it for authentication.

KRB_SERVER_ENCRYPT
Specifies that the server accepts KERBEROS authentication or encrypted SERVER authentication schemes. If the client authentication is KERBEROS, the client is authenticated using the Kerberos security system. If the client authentication is SERVER_ENCRYPT, the client is authenticated using a user ID and encryption password. If the client authentication is not specified, then the client will use Kerberos if available, otherwise it will use password encryption. For other client authentication types, an authentication error is returned. The authentication type of the client cannot be specified as KRB_SERVER_ENCRYPT

Note: The Kerberos authentication types are supported on clients and servers running on specific operating systems, refer to the related information section for more information. For Windows operating systems, both client and server machines must either belong to the same Windows domain or belong to trusted domains. This authentication type should be used when the server supports Kerberos and some, but not all, of the client machines support Kerberos authentication.

DATA_ENCRYPT
The server accepts encrypted SERVER authentication schemes and the encryption of user data. The authentication works exactly the same way as that shown with SERVER_ENCRYPT. See that authentication type for more information.

The following user data are encrypted when using this authentication type:

- SQL and XQuery statements.
- SQL program variable data.
- Output data from the server processing of an SQL or XQuery statement and including a description of the data.
- Some or all of the answer set data resulting from a query.
- Large object (LOB) data streaming.
- SQLDA descriptors.

DATA_ENCRYPT_CMP
The server accepts encrypted SERVER authentication schemes and the encryption of user data. In addition, this authentication type allows compatibility with down level products not supporting DATA_ENCRYPT

authentication type. These products are permitted to connect with the SERVER_ENCRYPT authentication type and without encrypting user data. Products supporting the new authentication type must use it. This authentication type is only valid in the server's database manager configuration file and is not valid when used on the CATALOG DATABASE command.

GSSPLUGIN
Specifies that the server uses a GSS-API plug-in to perform authentication. If the client authentication is not specified, the server returns a list of server-supported plug-ins, including any Kerberos plug-in that is listed in the *srvcon_gssplugin_list* database manager configuration parameter, to the client. The client selects the first plug-in found in the client plug-in directory from the list. If the client does not support any plug-in in the list, the client is authenticated using the Kerberos authentication scheme (if it is returned). If the client authentication is the GSSPLUGIN authentication scheme, the client is authenticated using the first supported plug-in in the list.

GSS_SERVER_ENCRYPT
Specifies that the server accepts plug-in authentication or encrypted server authentication schemes. If client authentication occurs through a plug-in, the client is authenticated using the first client-supported plug-in in the list of server-supported plug-ins.

If the client authentication is not specified and an implicit connect is being performed (that is, the client does not supply a user ID and password when making the connection), the server returns a list of server-supported plug-ins, the Kerberos authentication scheme (if one of the plug-ins in the list is Kerberos-based), and the encrypted server authentication scheme. The client is authenticated using the first supported plug-in found in the client plug-in directory. If the client does not support any of the plug-ins that are in the list, the client is authenticated using the Kerberos authentication scheme. If the client does not support the Kerberos authentication scheme, the client is authenticated using the encrypted server authentication scheme, and the connection will fail because of a missing password. A client supports the Kerberos authentication scheme if a DB2-supplied Kerberos plug-in exists for the operating system, or a Kerberos-based plug-in is specified for the *srvcon_gssplugin_list* database manager configuration parameter.

If the client authentication is not specified and an explicit connection is being performed (that is, both the user ID and password are supplied), the authentication type is equivalent to SERVER_ENCRYPT.

Note:

1. Do not inadvertently lock yourself out of your instance when you are changing the authentication information, since access to the configuration file itself is protected by information in the configuration file. The following database manager configuration file parameters control access to the instance:
   - AUTHENTICATION *
   - SYSADM_GROUP *
   - TRUST_ALLCLNTS
   - TRUST_CLNTAUTH
   - SYSCTRL_GROUP
   - SYSMAINT_GROUP

* Indicates the two most important parameters, and those most likely to cause a problem.

There are some things that can be done to ensure this does not happen: If you do accidentally lock yourself out of the DB2 database system, you have a fail-safe option available on all platforms that will allow you to override the usual DB2 database security checks to update the database manager configuration file using a highly privileged local operating system security user. This user *always* has the privilege to update the database manager configuration file and thereby correct the problem. However, this security bypass is restricted to a local update of the database manager configuration file. You cannot use a fail-safe user remotely or for any other DB2 database command. This special user is identified as follows:

- UNIX platforms: the instance owner
- Windows platform: someone belonging to the local "administrators" group
- Other platforms: there is no local security on the other platforms, so all users pass local security checks anyway

Authentication considerations for remote clients

When cataloging a database for remote access, the authentication type can be specified in the database directory entry.

The authentication type is not required. If it is not specified, the client will default to SERVER_ENCRYPT. However, if the server does not support SERVER_ENCRYPT, the client attempts to retry using a value supported by the server. If the server supports multiple authentication types, the client will not choose among them, but instead returns an error. The error is returned to ensure that the correct authentication type is used. In this case, the client must catalog the database using a supported authentication type. If an authentication type is specified, authentication can begin immediately provided that value specified matches that at the server. If a mismatch is detected, DB2 database attempts to recover. Recovery may result in more flows to reconcile the difference, or in an error if the DB2 database cannot recover. In the case of a mismatch, the value at the server is assumed to be correct.

The authentication type DATA_ENCRYPT_CMP is designed to allow clients from a previous release that does not support data encryption to a server using SERVER_ENCRYPT authentication instead of DATA_ENCRYPT. This authentication does not work when the following statements are true:

- The client level is Version 7.2.
- The gateway level is Version 8 FixPak7 or later.
- The server is Version 8 FixPak 7 or later.

When these are all true, the client cannot connect to the server. To allow the connection, you must either upgrade your client to Version 8, or have your gateway level at Version 8 FixPak 6 or earlier.

The determination of the authentication type used when connecting is made by specifying the appropriate authentication type as a database catalog entry at the gateway. This is true for both DB2 Connect™ scenarios and for clients and servers in a partitioned database environment where the client has set the DB2NODE registry variable. You will catalog the authentication type at the catalog partition with the intent to "hop" to the appropriate partition. In this scenario, the authentication type cataloged at the gateway is not used because the negotiation is solely between the client and the server.

You may have a need to catalog multiple database aliases at the gateway using different authentication types if they need to have clients that use differing authentication types. When deciding which authentication type to catalog at a gateway, you can keep the authentication type the same as that used at the client and server; or, you can use the NOTSPEC authentication type with the understanding that NOTSPEC defaults to SERVER.

Partitioned database authentication considerations

In a partitioned database, each partition of the database must have the same set of users and groups defined. If the definitions are not the same, the user may be authorized to do different things on different partitions. Consistency across all partitions is recommended.

Kerberos authentication details

The DB2 database system provides support for the Kerberos authentication protocol on AIX®, Solaris, Linux IA32 and AMD64, and Windows operating systems.

The Kerberos support is provided as a GSS-API security plugin named "IBMkrb5" which is used as both a server and as a client authentication plugin. The library is placed in the sqllib/security{32|64}/plugin/IBM/{client|server} directories for UNIX and Linux; and the sqllib/security/plugin/IBM{client|server} directories for Windows.

Note: For 64-bit Windows, the plugin library is called IBMkrb564.dll. Furthermore, the actual plugin source code for the UNIX and Linux plugin, IBMkrb5.C, is available in the sqllib/samples/security/plugins directory.

A good understanding of using and configuring Kerberos is strongly recommended before attempting to use Kerberos authentication with DB2 database system.

Kerberos description and introduction

Kerberos is a third party network authentication protocol that employs a system of shared secret keys to securely authenticate a user in an unsecured network environment. A three-tiered system is used in which encrypted tickets (provided by a separate server called the Kerberos Key Distribution Center, or KDC for short) are exchanged between the application server and client rather than a text user ID and password pair. These encrypted service tickets (called *credentials*) have a finite lifetime and are only understood by the client and the server. This reduces the security risk, even if the ticket is intercepted from the network. Each user, or *principal* in Kerberos terms, possesses a private encryption key that is shared with the KDC. Collectively, the set of principals and computers registered with a KDC are known as a *realm*.

A key feature of Kerberos is that it permits a single sign-on environment whereby a user only needs to verify his identity to the resources within the Kerberos realm once. When working with DB2 database, this means that a user is able to connect or attach to a DB2 database server without providing a user ID or password. Another advantage is that the user ID administration is simplified because a central repository for principals is used. Finally, Kerberos supports mutual authentication which allows the client to validate the identity of the server.

Kerberos set-up

DB2 database system and its support of Kerberos relies upon the Kerberos layer being installed and configured properly on all machines involved prior to the involvement of DB2 database. This includes, but is not necessarily limited to, the following requirements:

1. The client and server machines and principals must belong to the same realm, or else trusted realms (or trusted domains in the Windows terminology)
2. Creation of appropriate principals
3. Creation of server keytab files, where appropriate
4. All machines involved must have their system clocks synchronized (Kerberos typically permits a 5 minute time skew, otherwise a preauthentication error may occur when obtaining credentials).

For details on installing and configuring Kerberos please refer to the documentation provided with the installed Kerberos product.

The sole concern of DB2 database system will be whether the Kerberos security context is successfully created based on the credentials provided by the connecting application (that is, authentication). Other Kerberos features, such as the signing or encryption of messages, will not be used. Furthermore, whenever available, mutual authentication will be supported.

The Kerberos prerequisites are as follows:

- For the AIX, Solaris operating environment, and Linux platforms, the IBM® Network Authentication Service (NAS) Toolkit v1.4 or higher is required. You can download NAS Toolkits at https://www6.software.ibm.com/dl/dm/dm-nas-p
- For the Windows platforms, there are no prerequisites.

Kerberos and client principals

The principal may be found in either a 2-part or multi-part format, (that is, *name@REALM* or *name/instance@REALM*). As the "name" part will be used in the authorization ID (AUTHID) mapping, the name must adhere to the DB2 database naming rules. This means that the name may be up to 30 characters long and it must adhere to the existing restrictions on the choice of characters used. (AUTHID mapping is discussed in a later topic.)

Note: Windows directly associates a Kerberos principal with a domain user. An implication of this is that Kerberos authentication is not available to Windows machines that are not associated with a domain or realm. Furthermore, Windows only supports 2-part names (that is, *name@domain*).

The principal itself must be capable of obtaining outbound credentials with which it may request and receive service tickets to the target database. This is normally accomplished with the kinit command on UNIX or Linux, and is done implicitly at logon time on Windows.

Kerberos and authorization ID mapping

Unlike operating system user IDs whose scope of existence is normally restricted to a single machine, Kerberos principals have the ability to be authenticated in realms other than their own. The potential problem of duplicated principal names

is avoided by using the realm name to fully qualify the principal. In Kerberos, a fully qualified principal takes the form name/instance@REALM where the instance field may actually be multiple instances separated by a "/", that is, name/instance1/instance2@REALM, or it may be omitted altogether. The obvious restriction is that the realm name must be unique within all the realms defined within a network. The problem for DB2 database is that in order to provide a simple mapping from the principal to the AUTHID, a one-to-one mapping between the principal name, that is, the "name" in the fully qualified principal, and the AUTHID is desirable. A simple mapping is needed as the AUTHID is used as the default schema in DB2 database and should be easily and logically derived. As a result, the database administrator needs to be aware of the following potential problems:

- Principals from different realms but with the same name will be mapped to the same AUTHID.
- Principals with the same name but different instances will be mapped to the same AUTHID.

Giving consideration to the above, the following recommendations are made:

- Maintain an unique namespace for the name within all the trusted realms that will access the DB2 database server
- All principals with the same name, regardless of the instance, should belong to the same user.

Kerberos and server principals

On UNIX or Linux, the server principal name for the DB2 database instance is assumed to be <instance name>/<fully qualified hostname>@REALM. This principal must be able to accept Kerberos security contexts and it must exist before starting the DB2 database instance since the server name is reported to DB2 database by the plugin at initialization time.

On Windows, the server principal is taken to be the domain account under which the DB2 database service started. An exception to this is the instance may be started by the local SYSTEM account, in which case, the server principal name is reported as host/<hostname>; this is only valid if both the client and server belong to Windows domains.

Windows does not support greater than 2-part names. This poses a problem when a Windows client attempts to connect to a UNIX server. As a result, a Kerberos principal to Windows account mapping may need to be set up in the Windows domain if interoperability with UNIX Kerberos is required. (Please refer to the appropriate Microsoft® documentation for relevant instructions.)

You can override the Kerberos server principal name used by the DB2 server on UNIX and Linux operating systems. Set the DB2_KRB5_PRINCIPAL environment variable to the desired fully qualified server principal name. The instance must be restarted because the server principal name is only recognized by the DB2 database system after **db2start** is run.

Kerberos keytab files

Every Kerberos service on UNIX or Linux wishing the accept security context requests must place its credentials in a *keytab* file. This applies to the principals used by DB2 database as server principals. Only the default keytab file is searched

for the server's key. For instructions on adding a key to the keytab file, please refer to the documentation provided with the Kerberos product.

There is no concept of a keytab file on Windows and the system automatically handles storing and acquiring the credentials handle for a principal.

Kerberos and groups

Kerberos is an authentication protocol that does not possess the concept of groups. As a result, DB2 database relies upon the local operating system to obtain a group list for the Kerberos principal. For UNIX or Linux, this requires that an equivalent system account should exist for each principal. For example, for the principal name@REALM, DB2 database collects group information by querying the local operating system for all group names to which the operating system user *name* belongs. If an operating system user does not exist, then the AUTHID will only belong to the PUBLIC group. Windows, on the other hand, automatically associates a domain account to a Kerberos principal and the additional step to create a separate operating system account is not required.

Enabling Kerberos authentication on the client

The *clnt_krb_plugin* database manager configuration parameter should be updated to the name of the Kerberos plugin being used. On the supported platforms this should be set to IBMkrb5. This parameter will inform DB2 database that it is capable of using Kerberos for connections and local instance-level actions if the AUTHENTICATION parameter is set to KERBEROS or KRB_SERVER_ENCRYPT. Otherwise, no client-side Kerberos support is assumed.

Note: No checks are performed to validate that Kerberos support is available.

Optionally, when cataloging a database on the client, an authentication type may be specified:

```
db2 catalog db testdb at node testnode authentication kerberos target
   principal service/host@REALM
```

However, if the authentication information is not provided, then the server sends the client the name of the server principal.

Enabling Kerberos authentication on the server

The *srvcon_gssplugin_list* database manager configuration parameter should be updated with the server Kerberos plugin name. Although this parameter may contain a list of supported plugins, only one Kerberos plugin may be specified. However, if this field is blank and AUTHENTICATION is set to KERBEROS or KRB_SERVER_ENCRYPT, the default Kerberos plugin (IBMkrb5) is assumed and used. Either the AUTHENTICATION or SVRCON_AUTH parameter should be set to KERBEROS or KRB_SERVER_ENCRYPT if Kerberos authentication is to be used depending upon whether it is used for everything or just for incoming connections.

Creating a Kerberos plugin

There are several considerations you should consider when creating a Kerberos plugin:

- Write a Kerberos plugin as a GSS-API plugin with the notable exception that the *plugintype* in the function pointer array returned to DB2 database in the initialization function must be set to DB2SEC_PLUGIN_TYPE_KERBEROS.

- Under certain conditions, the server principal name may be reported to the client by the server. As such, the principal name should not be specified in the GSS_C_NT_HOSTBASED_SERVICE format (service@host), since DRDA stipulates that the principal name be in the GSS_C_NT_USER_NAME format (server/host@REALM).
- In a typical situation, the default keytab file may be specified by the KRB5_KTNAME environment variable. However, as the server plugin will run within a DB2 database engine process, this environment variable may not be accessible.

zSeries® and System i compatibility

For connections to zSeries and System i, the database must be cataloged with the AUTHENTICATION KERBEROS parameter and the TARGET PRINCIPAL parameter name must be explicitly specified.

Neither zSeries nor System i support mutual authentication.

Windows issues

When you are using Kerberos on Windows platforms, you need to be aware of the following issues:

- Due to the manner in which Windows detects and reports some errors, the following conditions result in an unexpected client security plug-in error (SQL30082N, rc=36):
  - Expired account
  - Invalid password
  - Expired password
  - Password change forced by administrator
  - Disabled account

  Furthermore, in all cases, the DB2 administration log or db2diag.log will indicate ″Logon failed″ or ″Logon denied″.
- If a domain account name is also defined locally, connections explicitly specifying the domain name and password will fail with the following error: `The Local Security Authority cannot be contacted`.

  The error is a result of Windows locating the local user first. The solution is to fully qualify the user in the connection string. For example: `name@DOMAIN.IBM.COM`
- Windows accounts cannot include the @ character in their name because the character is assumed to be the domain separator by the DB2 Kerberos plug-in.
- When interoperating with a non-Windows platform, ensure that all Windows domain server accounts and all Windows client accounts are configured to use DES encryption. If the account used to start the DB2 service is not configured to use DES encryption, the DB2 server will fail to accept Kerberos contexts. In particular, DB2 will fail with an unexpected server plug-in error, and will log that the AcceptSecurityContext API returned SEC_I_CONTINUE_NEEDED (0x00090312L).

  To determine if Windows accounts are configured to use DES encryption, look under **Account properties** in the **Active Directory**. A restart might be required if the account properties are changed.
- If the client and server are both on Windows, then the DB2 service can be started under the local system account. However, if the client and server are in

different domains, the connection might fail with an invalid target principal name error. The workaround is to explicitly catalog the target principal name on the client using the fully qualified server host name and the fully qualified domain name, in the following format: host/*server hostname@server domain name*

For example: `host/myhost.domain.ibm.com@DOMAIN.IBM.COM`

Otherwise, you must start the DB2 service under a valid domain account.

Maintaining passwords on servers

You might be required to perform password maintenance tasks. Because such tasks are typically required at the server, and many users are not able or comfortable working with the server environment, performing these tasks can pose a significant challenge.

DB2 database system provides a way to update and verify passwords without having to be at the server. You can use DB2 database products to change passwords on AIX, Linux and Windows operating systems.

For example, if an error message SQL1404N "Password expired" or SQL30082N "Security processing failed with reason 1 (PASSWORD EXPIRED)" is received, use the CONNECT statement to change the password as follows:

```
CONNECT TO database USER userid USING
    password NEW new_password CONFIRM new_password
```

The ATTACH command and the **Password change** dialog of the DB2 Configuration Assistant (CA) can also be used to change the password.

Authorization, privileges, and object ownership

Users (identified by an authorization ID) can successfully execute SQL or XQuery statements only if they have the authority to perform the specified function. To create a table, a user must be authorized to create tables; to alter a table, a user must be authorized to alter the table; and so forth.

There are three forms of authorization, *administrative authority*, *privileges*, and *LBAC credentials*, discussed below.

The database manager requires that each user be specifically authorized, either implicitly or explicitly, to use each database function needed to perform a specific task. *Explicit* authorities or privileges are granted to the user (GRANTEETYPE of U in the database catalogs). *Implicit* authorities or privileges are granted to a group to which the user belongs (GRANTEETYPE of G in the database catalogs) or to a role in which the user, the group or another role is a member (GRANTEETYPE of R in the database catalogs).

Administrative authority

The person or persons holding administrative authority are charged with the task of controlling the database manager and are responsible for the safety and integrity of the data. Those with administrative authority levels of SYSADM and DBADM implicitly have all privileges on all objects except objects pertaining to database security and control who will have access to the database manager and the extent of this access.

Authority levels provide a method of grouping privileges and higher-level database manager maintenance and utility operations. *Database authorities* enable users to perform activities at the database level. A user, group, or role can have one or more of the following authorities:

- Administrative authority level that operates at the instance level, SYSADM (system administrator)

  The SYSADM authority level provides control over all the resources created and maintained by the database manager. The system administrator possesses all the authorities of DBADM, SYSCTRL, SYSMAINT, and SYSMON, and the authority to grant and revoke DBADM authority and SECADM authority.

  The user who has SYSADM authority is responsible both for controlling the database manager, and for ensuring the safety and integrity of the data. SYSADM authority provides implicit DBADM authority within a database but does not provide implicit SECADM authority within a database.
- Administrative authority levels that operate at the database level:
  - DBADM (database administrator)

    The DBADM authority level applies at the database level and provides administrative authority over a single database. This database administrator possesses the privileges required to create objects, issue database commands, and access table data. The database administrator can also grant and revoke CONTROL and individual privileges.
  - SECADM (security administrator)

    The SECADM authority level applies at the database level and is the authority required to create, alter (where applicable), and drop roles, trusted contexts, audit policies, security label components, security policies, and security labels, which are used to protect tables. It is also the authority required to grant and revoke roles, security labels and exemptions as well as to grant and revoke the SETSESSIONUSER privilege. A user with the SECADM authority can transfer the ownership of objects that they do not own. They can also use the AUDIT statement to associate an audit policy with a particular database or database object at the server.

    The SECADM authority has no inherent privilege to access data stored in tables and has no other additional inherent privilege. It can only be granted by a user with SYSADM authority. The SECADM authority can be granted to a user but cannot be granted to a group, a role or to PUBLIC.
- System control authority levels that operate at the instance level:
  - SYSCTRL (system control)

    The SYSCTRL authority level provides control over operations that affect system resources. For example, a user with SYSCTRL authority can create, update, start, stop, or drop a database. This user can also start or stop an instance, but cannot access table data. Users with SYSCTRL authority also have SYSMON authority.
  - SYSMAINT (system maintenance)

    The SYSMAINT authority level provides the authority required to perform maintenance operations on all databases associated with an instance. A user with SYSMAINT authority can update the database configuration, backup a database or table space, restore an existing database, and monitor a database. Like SYSCTRL, SYSMAINT does not provide access to table data. Users with SYSMAINT authority also have SYSMON authority.
- The SYSMON (system monitor) authority level

  SYSMON provides the authority required to use the database system monitor. It operates at the instance level.

- Database authorities

  To perform activities such as creating a table or a routine, or for loading data into a table, specific database authorities are required. For example, the LOAD database authority is required for use of the load utility to load data into tables (a user must also have INSERT privilege on the table).

Figure 1 illustrates the relationship between authorities and their span of control (database, database manager).

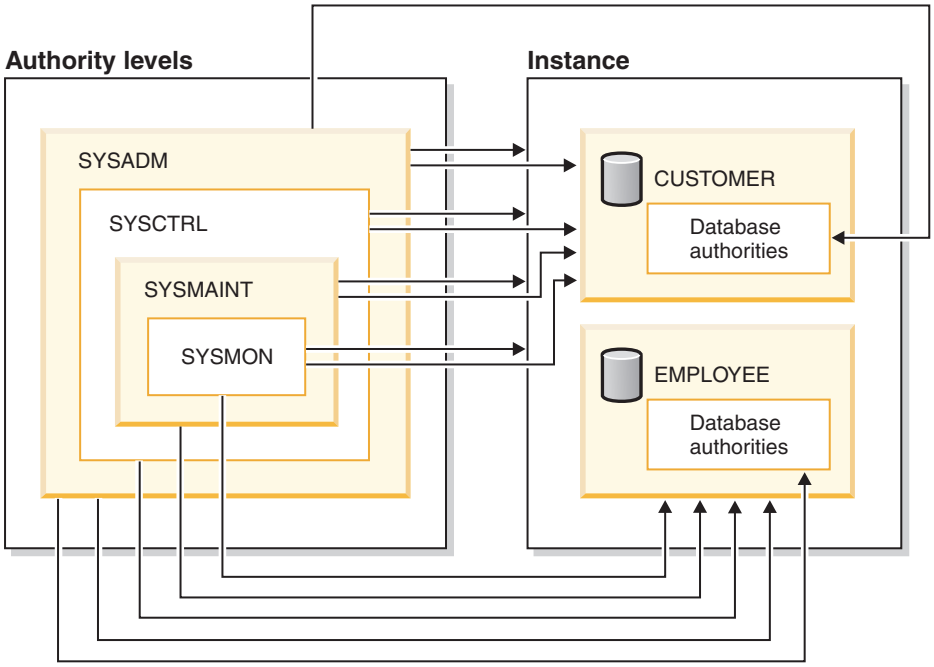


Figure 1. Hierarchy of Authorities

Privileges

Privileges are those activities that a user is allowed to perform. Authorized users can create objects, have access to objects they own, and can pass on privileges on their own objects to other users by using the GRANT statement.

Privileges may be granted to individual users, to groups, or to PUBLIC. PUBLIC is a special group that consists of all users, including future users. Users that are members of a group will indirectly take advantage of the privileges granted to the group, where groups are supported.

The CONTROL privilege: Possessing the CONTROL privilege on an object allows a user to access that database object, and to grant and revoke privileges to or from other users on that object.

Note: The CONTROL privilege only apples to tables, views, nicknames, indexes, and packages.

If a different user requires the CONTROL privilege to that object, a user with SYSADM or DBADM authority could grant the CONTROL privilege to that object. The CONTROL privilege cannot be revoked from the object owner, however, the object owner can be changed by using the TRANSFER OWNERSHIP statement.

In some situations, the creator of an object automatically obtains the CONTROL privilege on that object.

Individual privileges: Individual privileges can be granted to allow a user to carry out specific tasks on specific objects. Users with administrative authority (SYSADM or DBADM) or the CONTROL privilege can grant and revoke privileges to and from users.

Individual privileges and database authorities allow a specific function, but do not include the right to grant the same privileges or authorities to other users. The right to grant table, view, schema, package, routine, and sequence privileges to others can be extended to other users through the WITH GRANT OPTION on the GRANT statement. However, the WITH GRANT OPTION does not allow the person granting the privilege to revoke the privilege once granted. You must have SYSADM authority, DBADM authority, or the CONTROL privilege to revoke the privilege.

Privileges on objects in a package or routine: When a user has the privilege to execute a package or routine, they do not necessarily require specific privileges on the objects used in the package or routine. If the package or routine contains static SQL or XQuery statements, the privileges of the owner of the package are used for those statements. If the package or routine contains dynamic SQL or XQuery statements, the authorization ID used for privilege checking depends on the setting of the DYNAMICRULES bind option of the package issuing the dynamic query statements, and whether those statements are issued when the package is being used in the context of a routine.

A user or group can be authorized for any combination of individual privileges or authorities. When a privilege is associated with an object, that object must exist. For example, a user cannot be given the SELECT privilege on a table unless that table has previously been created.

Note: Care must be taken when an authorization name representing a user or a group is granted authorities and privileges and there is no user, or group created with that name. At some later time, a user or a group can be created with that name and automatically receive all of the authorities and privileges associated with that authorization name.

The REVOKE statement is used to revoke previously granted privileges. The revoking of a privilege from an authorization name revokes the privilege granted by all authorization names.

Revoking a privilege from an authorization name does not revoke that same privilege from any other authorization names that were granted the privilege by that authorization name. For example, assume that CLAIRE grants SELECT WITH GRANT OPTION to RICK, then RICK grants SELECT to BOBBY and CHRIS. If CLAIRE revokes the SELECT privilege from RICK, BOBBY and CHRIS still retain the SELECT privilege.

LBAC credentials

Label-based access control (LBAC) lets the security administrator decide exactly who has write access and who has read access to individual rows and individual columns. The security administrator configures the LBAC system by creating security policies. A security policy describes the criteria used to decide who has

access to what data. Only one security policy can be used to protect any one table but different tables can be protected by different security policies.

After creating a security policy, the security administrator creates database objects, called security labels and exemptions that are part of that policy. A security label describes a certain set of security criteria. An exemption allows a rule for comparing security labels not to be enforced for the user who holds the exemption, when they access data protected by that security policy.

Once created, a security label can be associated with individual columns and rows in a table to protect the data held there. Data that is protected by a security label is called protected data. A security administrator allows users access to protected data by granting them security labels. When a user tries to access protected data, that user's security label is compared to the security label protecting the data. The protecting label blocks some security labels and does not block others.

Object ownership

When an object is created, one authorization ID is assigned *ownership* of the object. Ownership means the user is authorized to reference the object in any applicable SQL or XQuery statement.

When an object is created within a schema, the authorization ID of the statement must have the required privilege to create objects in the implicitly or explicitly specified schema. That is, the authorization name must either be the owner of the schema, or possess the CREATEIN privilege on the schema.

Note: This requirement is not applicable when creating table spaces, buffer pools or database partition groups. These objects are not created in schemas.

When an object is created, the authorization ID of the statement is the owner of that object.

Note: One exception exists. If the AUTHORIZATION option is specified for the CREATE SCHEMA statement, any other object that is created as part of the CREATE SCHEMA operation is owned by the authorization ID specified by the AUTHORIZATION option. Any objects that are created in the schema after the initial CREATE SCHEMA operation, however, are owned by the authorization ID associated with the specific CREATE statement.

For example, the statement `CREATE SCHEMA SCOTTSTUFF AUTHORIZATION SCOTT CREATE TABLE T1 (C1 INT)` creates the schema `SCOTTSTUFF` and the table `SCOTTSTUFF.T1`, which are both owned by SCOTT. Assume that the user BOBBY is granted the CREATEIN privilege on the `SCOTTSTUFF` schema and creates an index on the `SCOTTSTUFF.T1` table. Because the index is created after the schema, BOBBY owns the index on `SCOTTSTUFF.T1`.

Privileges are assigned to the object owner based on the type of object being created:

- The CONTROL privilege is implicitly granted on newly created tables, indexes, and packages. This privilege allows the object creator to access the database object, and to grant and revoke privileges to or from other users on that object. If a different user requires the CONTROL privilege to that object, a user with SYSADM or DBADM authority must grant the CONTROL privilege to that object. The CONTROL privilege cannot be revoked by the object owner.

- The CONTROL privilege is implicitly granted on newly created views if the object owner has the CONTROL privilege on all the tables, views, and nicknames referenced by the view definition.
- Other objects like triggers, routines, sequences, table spaces, and buffer pools do not have a CONTROL privilege associated with them. The object owner does, however, automatically receive each of the privileges associated with the object (and can provide these privileges to other users, where supported, by using the WITH GRANT option of the GRANT statement). In addition, the object owner can alter, add a comment on, or drop the object. These authorizations are implicit for the object owner and cannot be revoked.

Certain privileges on the object, such as altering a table, can be granted by the owner, and can be revoked from the owner by a user who has SYSADM or DBADM authority. Certain privileges on the object, such as commenting on a table, cannot be granted by the owner and cannot be revoked from the owner. Use the TRANSFER OWNERSHIP statement to move these privileges to another user. When an object is created, the authorization ID of the statement is the owner of the object. However, when a package is created and the OWNER bind option is specified, the owner of objects created by the static SQL statements in the package is the value of the OWNER bind option. In addition, if the AUTHORIZATION clause is specified on a CREATE SCHEMA statement, the authorization name specified after the AUTHORIZATION keyword is the owner of the schema.

A security administrator or the object owner can use the TRANSFER OWNERSHIP statement to change the ownership of a database object. An administrator can therefore create an object on behalf of an authorization ID, by creating the object using the authorization ID as the qualifier, and then using the TRANSFER OWNERSHIP statement to transfer the ownership that the administrator has on the object to the authorization ID.

Authorization IDs in different contexts

An authorization ID is used for two purposes: identification and authorization checking. For example, the session authorization ID is used for initial authorization checking.

When referring to the use of an authorization ID in a specific context, the reference to the authorization is qualified to identify the context, as shown below.

Contextual reference to authorization ID
Definition

System authorization ID
The authorization ID used to do any initial authorization checking, such as checking for CONNECT privilege during CONNECT processing. As part of the authentication process during CONNECT processing, an authorization ID compatible with DB2 naming requirements is produced that represents the external user ID within the DB2 database system. The system authorization ID represents the user that created the connection. Use the SYSTEM_USER special register to see the current value of the system authorization ID. The system authorization ID cannot be changed for a connection.

Session authorization ID
The authorization ID used for any session authorization checking subsequent to the initial checks performed during CONNECT processing. The default value of the session authorization ID is the value of the system authorization ID. Use the SESSION_USER special register to see the current

value of the session authorization ID. The USER special register is a synonym for the SESSION_USER special register. The session authorization ID can be changed by using the SET SESSION AUTHORIZATION statement.

Package authorization ID
The authorization ID used to bind a package to the database. This authorization ID is obtained from the value of the OWNER bind option. The package authorization ID is sometimes referred to as the package binder or package owner.

Routine owner authorization ID
The authorization ID listed in the system catalogs as the owner of the SQL routine that has been invoked.

Routine invoker authorization ID
The authorization ID that is the statement authorization ID for the statement that invoked an SQL routine.

Statement authorization ID
The authorization ID associated with a specific SQL statement that is to be used for any authorization requirements as well as for determining object ownership (where appropriate). It takes its value from the appropriate source authorization ID, depending on the type of SQL statement:

- Static SQL

  The package authorization ID is used.
- Dynamic SQL (from non-routine context)

  The table shows which authorization ID is used in each case:

Value of DYNAMICRULES option for issuing the package	Authorization ID used
RUN	Session authorization ID
BIND	Package authorization ID
DEFINERUN, INVOKERUN	Session authorization ID
DEFINEBIND, INVOKEBIND	Package authorization ID

- Dynamic SQL (from routine context)

  The table shows which authorization ID is used in each case:

Value of DYNAMICRULES option for issuing the package	Authorization ID used
DEFINERUN, DEFINEBIND	Routine owner authorization ID
INVOKERUN, INVOKEBIND	Routine invoker authorization ID

Use the CURRENT_USER special register to see the current value of the statement authorization ID. The statement authorization ID cannot be changed directly; it is changed automatically by the DB2 database system to reflect the nature of each SQL statement.

Instance level authorities

System administration authority (SYSADM)

The SYSADM authority level is the highest level of administrative authority. Users with SYSADM authority can run utilities, issue database and database manager commands, and access any data that is not protected by LBAC in any table in any

database within the database manager instance. It provides the ability to control all database objects in the instance, including databases, tables, views, indexes, packages, schemas, servers, aliases, data types, functions, procedures, triggers, table spaces, database partition groups, buffer pools, and event monitors.

SYSADM authority is assigned to the group specified by the *sysadm_group* configuration parameter. Membership in that group is controlled outside the database manager through the security facility used on your platform.

Only a user with SYSADM authority can perform the following functions:

- Migrate a database
- Change the database manager configuration file (including specifying the groups having SYSCTRL, SYSMAINT, or SYSMON authority)
- Grant and revoke DBADM authority.
- Grant and revoke SECADM authority

While SYSADM authority does provide all abilities provided by most other authorities, it does not provide any of the abilities of the SECADM authority. The abilities provided by the SECADM authority are not provided by any other authority. SYSADM authority also does not provide access to data that is protected by LBAC.

Note: When a user with SYSADM authority creates a database, that user is automatically granted explicit DBADM authority on the database. If the database creator is removed from the SYSADM group and you want to prevent that user from accessing that database as a DBADM, you must explicitly revoke the user's DBADM authority.

System control authority (SYSCTRL)

SYSCTRL authority is the highest level of system control authority. This authority provides the ability to perform maintenance and utility operations against the database manager instance and its databases. These operations can affect system resources, but they do not allow direct access to data in the databases. System control authority is designed for users administering a database manager instance containing sensitive data.

SYSCTRL authority is assigned to the group specified by the *sysctrl_group* configuration parameter. If a group is specified, membership in that group is controlled outside the database manager through the security facility used on your platform.

Only a user with SYSCTRL authority or higher can do the following:

- Update a database, node, or distributed connection services (DCS) directory
- Force users off the system
- Create or drop a database
- Drop, create, or alter a table space
- Restore to a new database.

In addition, a user with SYSCTRL authority can perform the functions of users with system maintenance authority (SYSMAINT) and system monitor authority (SYSMON).

Users with SYSCTRL authority also have the implicit privilege to connect to a database.

Note: When users with SYSCTRL authority create databases, they are automatically granted explicit DBADM authority on the database. If the database creator is removed from the SYSCTRL group, and if you want to also prevent them from accessing that database as a DBADM, you must explicitly revoke this DBADM authority.

System maintenance authority (SYSMAINT)

SYSMAINT authority is the second level of system control authority. This authority provides the ability to perform maintenance and utility operations against the database manager instance and its databases. These operations can affect system resources, but they do not allow direct access to data in the databases. System maintenance authority is designed for users maintaining databases within a database manager instance that contains sensitive data.

SYSMAINT authority is assigned to the group specified by the *sysmaint_group* configuration parameter. If a group is specified, membership in that group is controlled outside the database manager through the security facility used on your platform.

Only a user with SYSMAINT or higher system authority can do the following:

- Update database configuration files
- Back up a database or table space
- Restore to an existing database
- Perform roll forward recovery
- Start or stop an instance
- Restore a table space
- Run trace
- Take database system monitor snapshots of a database manager instance or its databases.

A user with SYSMAINT, DBADM, or higher authority can do the following:

- Query the state of a table space
- Update log history files
- Quiesce a table space
- Reorganize a table
- Collect catalog statistics using the RUNSTATS utility.

Users with SYSMAINT authority also have the implicit privilege to connect to a database, and can perform the functions of users with system monitor authority (SYSMON).

System monitor authority (SYSMON)

SYSMON authority provides the ability to take database system monitor snapshots of a database manager instance or its databases. SYSMON authority is assigned to the group specified by the *sysmon_group* configuration parameter. If a group is specified, membership in that group is controlled outside the database manager through the security facility used on your platform.

SYSMON authority enables the user to run the following commands:

- GET DATABASE MANAGER MONITOR SWITCHES
- GET MONITOR SWITCHES
- GET SNAPSHOT
- LIST ACTIVE DATABASES

- LIST APPLICATIONS
- LIST DCS APPLICATIONS
- RESET MONITOR
- UPDATE MONITOR SWITCHES

SYSMON authority enables the user to use the following APIs:
- db2GetSnapshot - Get Snapshot
- db2GetSnapshotSize - Estimate Size Required for db2GetSnapshot() Output Buffer
- db2MonitorSwitches - Get/Update Monitor Switches
- db2ResetMonitor - Reset Monitor

SYSMON authority enables the user use the following SQL table functions:
- All snapshot table functions without previously running `SYSPROC.SNAP_WRITE_FILE`

  `SYSPROC.SNAP_WRITE_FILE` takes a snapshot and saves its content into a file. If any snapshot table functions are called with null input parameters, the file content is returned instead of a real-time system snapshot.

Users with the SYSADM, SYSCTRL, or SYSMAINT authority level also possess SYSMON authority.

Database level authorities

Security administration authority (SECADM)

SECADM authority is the authority required to create, alter (where applicable), and drop roles, trusted contexts, audit policies, security label components, security policies and security labels. It is also the authority required to grant and revoke roles, security labels and exemptions, and the SETSESSIONUSER privilege. SECADM authority has no inherent privilege to access data stored in tables.

SECADM (security administrator) authority can only be granted by the system administrator (who holds SYSADM authority) and can be granted to a user but not to a group or a role. It gives these and only these abilities:
- Create, alter, comment on, and drop security label components
- Create, alter, comment on, and drop security policies
- Create, comment on, and drop security labels
- Create, comment on, and drop roles
- Create, alter, comment on, and drop trusted contexts
- Create, alter, comment on, and drop audit policies
- Use of the audit system stored procedures and table function: SYSPROC.AUDIT_ARCHIVE, SYSPROC.AUDIT_LIST_LOGS, and SYSPROC.AUDIT_DELIM_EXTRACT. These can only be invoked by the security administrator.
- Use the AUDIT statement to associate an audit policy with a particular database or database object at the server
- Grant and revoke security labels
- Grant and revoke exemptions
- Grant and revoke SETSESSIONUSER privileges
- Grant and revoke roles
- Execute the SQL statement TRANSFER OWNERSHIP on objects not owned by the authorization ID of the statement

No other authority gives these abilities, not even SYSADM.

Database administration authority (DBADM)

DBADM authority is an administrative authority for a specific database and it allows the user to perform certain actions, and issue database commands on that database. The DBADM authority allows access to the data in any table in the database unless that data is protected by LBAC. To access data protected by LBAC you must have appropriate LBAC credentials.

When DBADM authority is granted, the following database authorities are also explicitly granted for the same database (and are not automatically revoked if the DBADM authority is later revoked):

- BINDADD
- CONNECT
- CREATETAB
- CREATE_EXTERNAL_ROUTINE
- CREATE_NOT_FENCED_ROUTINE
- IMPLICIT_SCHEMA
- QUIESCE_CONNECT
- LOAD

Only a user with SYSADM authority can grant or revoke DBADM authority. Users with DBADM authority can grant privileges on the database to others and can revoke any privilege from any user regardless of who granted it.

Holding the DBADM, or higher, authority for a database allows a user to perform these actions on that database:

- Read log files
- Create, activate, and drop event monitors.

A user with DBADM authority for a database or with SYSMAINT authority or higher can perform these actions on the database:

- Query the state of a table space
- Update log history files
- Quiesce a table space.
- Reorganize a table
- Collect catalog statistics using the RUNSTATS utility.

While DBADM authority does provide some of the same abilities as other authorities, it does not provide any of the abilities of the SECADM authority. The abilities provided by the SECADM authority are not provided by any other authority.

LOAD authority

Users having LOAD authority at the database level, as well as INSERT privilege on a table, can use the LOAD command to load data into a table.

Users having LOAD authority at the database level, as well as INSERT privilege on a table, can LOAD RESTART or LOAD TERMINATE if the previous load operation is a load to insert data.

Users having LOAD authority at the database level, as well as the INSERT and DELETE privileges on a table, can use the LOAD REPLACE command.

If the previous load operation was a load replace, the DELETE privilege must also have been granted to that user before the user can LOAD RESTART or LOAD TERMINATE.

If the exception tables are used as part of a load operation, the user must have INSERT privilege on the exception tables.

The user with this authority can perform QUIESCE TABLESPACES FOR TABLE, RUNSTATS, and LIST TABLESPACES commands.

Database authorities

Each database authority allows the authorization ID holding it to perform some particular type of action on the database as a whole. Database authorities are different from privileges, which allow a certain action to be taken on a particular database object, such as a table or an index.

These are the database authorities.

SECADM
Gives the holder the ability to act as a security administrator and create and drop security objects, grant and revoke authorization or privileges for security objects and transfer ownership of objects. The security administrator manages trusted contexts, audit policies, database roles, and LBAC-protection of data.

DBADM
Gives the holder the authority to act as the database administrator. In particular it gives the holder all of the other database authorities except for SECADM.

CONNECT
Allows the holder to connect to the database.

BINDADD
Allows the holder to create new packages in the database.

CREATETAB
Allows the holder to create new tables in the database.

CREATE_EXTERNAL_ROUTINE
Allows the holder to create a procedure for use by applications and other users of the database.

CREATE_NOT_FENCED_ROUTINE
Allows the holder to create a user-defined function (UDF) or procedure that is “not fenced”. CREATE_EXTERNAL_ROUTINE is automatically granted to any user who is granted CREATE_NOT_FENCED_ROUTINE.

Attention: The database manager does not protect its storage or control blocks from UDFs or procedures that are “not fenced”. A user with this authority must, therefore, be very careful to test their UDF extremely well before registering it as “not fenced”.

IMPLICIT_SCHEMA
Allows any user to create a schema implicitly by creating an object using a CREATE statement with a schema name that does not already exist.

SYSIBM becomes the owner of the implicitly created schema and PUBLIC is given the privilege to create objects in this schema.

LOAD
Allows the holder to load data into a table

QUIESCE_CONNECT
Allows the holder to access the database while it is quiesced.

Only authorization IDs with the SYSADM authority can grant the SECADM and DBADM authorities. All other authorities can be granted by authorization IDs that hold SYSADM or DBADM authorities.

When a database is created, the following database authorities are automatically granted to PUBLIC for the new database:

- CREATETAB
- BINDADD
- CONNECT
- IMPLICIT_SCHEMA

In addition, these privileges are granted:

- USE privilege on USERSPACE1 table space
- SELECT privilege on the system catalog views.

To remove any database authority from PUBLIC, an authorization ID with DBADM or SYSADM authority must explicitly revoke it.

Implicit schema authority (IMPLICIT_SCHEMA) considerations

When a new database is created, PUBLIC is given IMPLICIT_SCHEMA database authority. With this authority, any user can create a schema by creating an object and specifying a schema name that does not already exist. SYSIBM becomes the owner of the implicitly created schema and PUBLIC is given the privilege to create objects in this schema.

If control of who can implicitly create schema objects is required for the database, IMPLICIT_SCHEMA database authority should be revoked from PUBLIC. Once this is done, there are only three (3) ways that a schema object is created:

- Any user can create a schema using their own authorization name on a CREATE SCHEMA statement.
- Any user with DBADM authority can explicitly create any schema which does not already exist, and can optionally specify another user as the owner of the schema.
- Any user with DBADM authority has IMPLICIT_SCHEMA database authority (independent of PUBLIC) so that they can implicitly create a schema with any name at the time they are creating other database objects. SYSIBM becomes the owner of the implicitly created schema and PUBLIC has the privilege to create objects in the schema.

Privileges

Authorization ID privileges

Authorization ID privileges involve actions on authorization IDs. There is currently only one such privilege: the SETSESSIONUSER privilege.

The SETSESSIONUSER privilege can be granted to a user or to a group and allows the holder to switch identities to any of the authorization IDs on which the privilege was granted. The identity switch is made by using the SQL statement SET SESSION AUTHORIZATION. The SETSESSIONUSER privilege can only be granted by a user holding SECADM authority.

Note: When you migrate a Version 8 database to Version 9.1, or later, authorization IDs with explicit DBADM authority on that database are automatically granted SETSESSIONUSER privilege on PUBLIC. This prevents breaking applications that rely on authorization IDs with DBADM authority being able to set the session authorization ID to any authorization ID. This does not happen when the authorization ID has SYSADM authority but has not been explicitly granted DBADM.

Schema privileges

Schema privileges are in the object privilege category. Object privileges are shown in Figure 2.

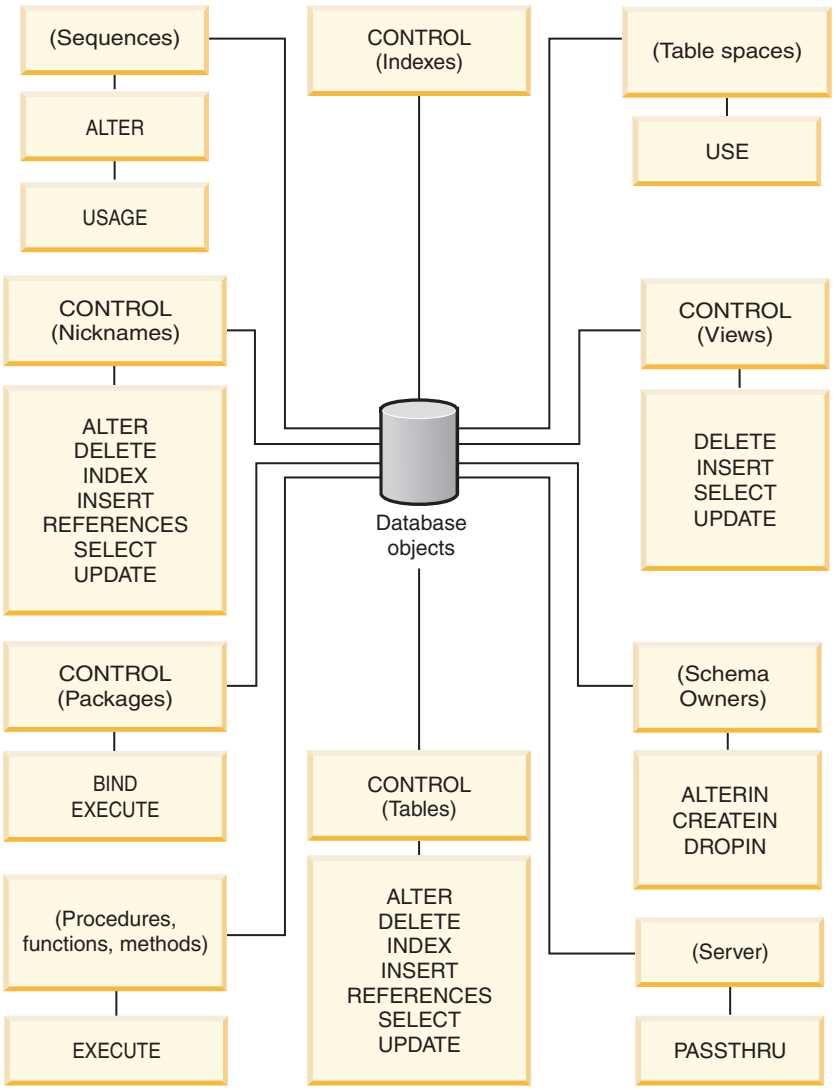


Figure 2. Object Privileges

Schema privileges involve actions on schemas in a database. A user may be granted any of the following privileges:

- CREATEIN allows the user to create objects within the schema.
- ALTERIN allows the user to alter objects within the schema.
- DROPIN allows the user to drop objects from within the schema.

The owner of the schema has all of these privileges and the ability to grant them to others. The objects that are manipulated within the schema object include: tables, views, indexes, packages, data types, functions, triggers, procedures, and aliases.

Table space privileges

The table space privileges involve actions on the table spaces in a database. A user may be granted the USE privilege for a table space which then allows them to create tables within the table space.

The owner of the table space, typically the creator who has SYSADM or SYSCTRL authority, has the USE privilege and the ability to grant this privilege to others. By default, at database creation time the USE privilege for table space USERSPACE1 is granted to PUBLIC, though this privilege can be revoked.

The USE privilege cannot be used with SYSCATSPACE or any system temporary table spaces.

Table and view privileges

Table and view privileges involve actions on tables or views in a database.

A user must have CONNECT authority on the database to use any of the following privileges:

- CONTROL provides the user with all privileges for a table or view including the ability to drop it, and to grant and revoke individual table privileges. You must have SYSADM or DBADM authority to grant CONTROL. The creator of a table automatically receives CONTROL privilege on the table. The creator of a view automatically receives CONTROL privilege only if they have CONTROL privilege on all tables, views, and nicknames referenced in the view definition, or they have SYSADM or DBADM authority.
- ALTER allows the user to modify on a table, for example, to add columns or a unique constraint to the table. A user with ALTER privilege can also COMMENT ON a table, or on columns of the table. For information about the possible modifications that can be performed on a table, see the ALTER TABLE and COMMENT statements.
- DELETE allows the user to delete rows from a table or view.
- INDEX allows the user to create an index on a table. Creators of indexes automatically have CONTROL privilege on the index.
- INSERT allows the user to insert a row into a table or view, and to run the IMPORT utility.
- REFERENCES allows the user to create and drop a foreign key, specifying the table as the parent in a relationship. The user might have this privilege only on specific columns.
- SELECT allows the user to retrieve rows from a table or view, to create a view on a table, and to run the EXPORT utility.
- UPDATE allows the user to change an entry in a table, a view, or for one or more specific columns in a table or view. The user may have this privilege only on specific columns.

The privilege to grant these privileges to others may also be granted using the WITH GRANT OPTION on the GRANT statement.

Note: When a user or group is granted CONTROL privilege on a table, all other privileges on that table are automatically granted WITH GRANT OPTION. If you subsequently revoke the CONTROL privilege on the table from a user, that user will still retain the other privileges that were automatically granted. To revoke all the privileges that are granted with the CONTROL privilege, you must either explicitly revoke each individual privilege or specify the ALL keyword on the REVOKE statement, for example:

```
REVOKE ALL
  ON EMPLOYEE FROM USER HERON
```

When working with typed tables, there are implications regarding table and view privileges.

Note: Privileges may be granted independently at every level of a table hierarchy. As a result, a user granted a privilege on a supertable within a hierarchy of typed tables may also indirectly affect any subtables. However, a user can only operate directly on a subtable if the necessary privilege is held on that subtable.

The supertable/subtable relationships among the tables in a table hierarchy mean that operations such as SELECT, UPDATE, and DELETE will affect the rows of the operation's target table and all its subtables (if any). This behavior can be called *substitutability*. For example, suppose that you have created an Employee table of type Employee_t with a subtable Manager of type Manager_t. A manager is a (specialized) kind of employee, as indicated by the type/subtype relationship between the structured types Employee_t and Manager_t and the corresponding table/subtable relationship between the tables Employee and Manager. As a result of this relationship, the SQL query:

```
SELECT * FROM Employee
```

will return the object identifier and Employee_t attributes for both employees and managers. Similarly, the update operation:

```
UPDATE Employee SET Salary = Salary + 1000
```

will give a thousand dollar raise to managers as well as regular employees.

A user with SELECT privilege on Employee will be able to perform this SELECT operation even if they do not have an explicit SELECT privilege on Manager. However, such a user will not be permitted to perform a SELECT operation directly on the Manager subtable, and will therefore not be able to access any of the non-inherited columns of the Manager table.

Similarly, a user with UPDATE privilege on Employee will be able to perform an UPDATE operation on Manager, thereby affecting both regular employees and managers, even without having the explicit UPDATE privilege on the Manager table. However, such a user will not be permitted to perform UPDATE operations directly on the Manager subtable, and will therefore not be able to update non-inherited columns of the Manager table.

Package privileges

A package is a database object that contains the information needed by the database manager to access data in the most efficient way for a particular application program. Package privileges enable a user to create and manipulate packages.

The user must have CONNECT authority on the database to use any of the following privileges:

- CONTROL provides the user with the ability to rebind, drop, or execute a package as well as the ability to extend those privileges to others. The creator of a package automatically receives this privilege. A user with CONTROL privilege is granted the BIND and EXECUTE privileges, and can also grant these privileges to other users by using the GRANT statement. (If a privilege is granted using WITH GRANT OPTION, a user who receives the BIND or EXECUTE privilege can, in turn, grant this privilege to other users.) To grant CONTROL privilege, the user must have SYSADM or DBADM authority.
- BIND privilege on a package allows the user to rebind or bind that package and to add new package versions of the same package name and creator.
- EXECUTE allows the user to execute or run a package.

Note: All package privileges apply to all VERSIONs that share the same package name and creator.

In addition to these package privileges, the BINDADD database privilege allows users to create new packages or rebind an existing package in the database.

Objects referenced by nicknames need to pass authentication checks at the data sources containing the objects. In addition, package users must have the appropriate privileges or authority levels for data source objects at the data source.

It is possible that packages containing nicknames might require additional authorization steps because DB2 database uses dynamic queries when communicating with DB2 Family data sources. The authorization ID running the package at the data source must have the appropriate authority to execute the package dynamically at that data source.

Index privileges

The creator of an index or an index specification automatically receives CONTROL privilege on the index. CONTROL privilege on an index is really the ability to drop the index. To grant CONTROL privilege on an index, a user must have SYSADM or DBADM authority.

The table-level INDEX privilege allows a user to create an index on that table.

The nickname-level INDEX privilege allows a user to create an index specification on that nickname.

Sequence privileges

The creator of a sequence automatically receives the USAGE and ALTER privileges on the sequence. The USAGE privilege is needed to use NEXT VALUE and PREVIOUS VALUE expressions for the sequence. To allow other users to use the NEXT VALUE and PREVIOUS VALUE expressions, sequence privileges must be granted to PUBLIC. This allows all users to use the expressions with the specified sequence.

ALTER privilege on the sequence allows the user to perform tasks such as restarting the sequence or changing the increment for future sequence values. The creator of the sequence can grant the ALTER privilege to other users, and if WITH GRANT OPTION is used, these users can, in turn, grant these privileges to other users.

Routine privileges

Execute privileges involve actions on all types of routines such as functions, procedures, and methods within a database. Once having EXECUTE privilege, a user can then invoke that routine, create a function that is sourced from that routine (applies to functions only), and reference the routine in any DDL statement such as CREATE VIEW or CREATE TRIGGER.

The user who defines the externally stored procedure, function, or method receives EXECUTE WITH GRANT privilege. If the EXECUTE privilege is granted to another user via WITH GRANT OPTION, that user can, in turn, grant the EXECUTE privilege to another user.

Usage privilege on workloads

To enable use of a workload, the database administrator can grant USAGE privilege on that workload to a user, a group, or a role using the GRANT USAGE ON WORKLOAD statement.

When the DB2 database system finds a matching workload, it checks whether the session user has USAGE privilege on that workload. If the session user does not have USAGE privilege on that workload, then the DB2 database system searches for the next matching workload in the ordered list. In other words, the workloads that the session user does not have USAGE privilege on are treated as if they do not exist.

The USAGE privilege information is stored in the catalogs and can be viewed through the SYSCAT.WORKLOADAUTH view.

The USAGE privilege can be revoked using the REVOKE USAGE ON WORKLOAD statement.

A user with SYSADM or DBADM authority can use any workload that exists in the catalog as long as the workload matches the connection attributes.

The SYSDEFAULTUSERWORKLOAD workload and the USAGE privilege

USAGE privilege on SYSDEFAULTUSERWORKLOAD is granted to PUBLIC at database creation time, if the database is created without the RESTRICT option. Otherwise, the USAGE privilege must be explicitly granted by a user with SYSADM or DBADM authority.

If the session user does not have USAGE privilege on any of the workloads, including SYSDEFAULTUSERWORKLOAD, an SQL error is returned.

The SYSDEFAULTADMWORKLOAD workload and the USAGE privilege

USAGE privilege on SYSDEFAULTADMWORKLOAD cannot be explicitly granted to any user. Only users who issue the SET WORKLOAD TO SYSDEFAULTADMWORKLOAD command and whose session authorization ID has SYSADM or DBADM authority are allowed to use this workload.

The GRANT USAGE ON WORKLOAD and REVOKE USAGE ON WORKLOAD statements do not have any effect on SYSDEFAULTADMWORKLOAD.

Tasks and required authorizations

Not all organizations divide job responsibilities in the same manner. Table 2 lists some other common job titles, the tasks that usually accompany them, and the authorities or privileges that are needed to carry out those tasks.

Table 2. Common Job Titles, Tasks, and Required Authorization

JOB TITLE	TASKS	REQUIRED AUTHORIZATION
Department Administrator	Oversees the departmental system; creates databases	SYSCTRL authority. SYSADM authority if the department has its own instance.
Security Administrator	Authorizes other users for some or all authorizations and privileges	SYSADM or DBADM authority.
Database Administrator	Designs, develops, operates, safeguards, and maintains one or more databases	DBADM and SYSMAINT authority over one or more databases. SYSCTRL authority in some cases.
System Operator	Monitors the database and carries out backup functions	SYSMAINT authority.
Application Programmer	Develops and tests the database manager application programs; may also create tables of test data	BINDADD, BIND on an existing package, CONNECT and CREATETAB on one or more databases, some specific schema privileges, and a list of privileges on some tables. CREATE_EXTERNAL_ROUTINE may also be required.
User Analyst	Defines the data requirements for an application program by examining the system catalog views	SELECT on the catalog views; CONNECT on one or more databases.
Program End User	Executes an application program	EXECUTE on the package; CONNECT on one or more databases. See the note following this table.
Information Center Consultant	Defines the data requirements for a query user; provides the data by creating tables and views and by granting access to database objects	DBADM authority over one or more databases.
Query User	Issues SQL statements to retrieve, add, delete, or change data; may save results as tables	CONNECT on one or more databases; CREATEIN on the schema of the tables and views being created; and, SELECT, INSERT, UPDATE, DELETE on some tables and views.

Note: If an application program contains dynamic SQL statements, the Program End User may need other privileges in addition to EXECUTE and CONNECT (such as SELECT, INSERT, DELETE, and UPDATE).

Granting, revoking and monitoring access

Granting privileges

To grant privileges on most database objects, the user must have SYSADM authority, DBADM authority, or CONTROL privilege on that object; or, the user must hold the privilege WITH GRANT OPTION. Privileges can be granted only on existing objects.

To grant CONTROL privilege to someone else, the user must have SYSADM or DBADM authority. To grant DBADM authority, the user must have SYSADM authority.

The GRANT statement allows an authorized user to grant privileges. A privilege can be granted to one or more authorization names in one statement; or to PUBLIC, which makes the privileges available to all users. Note that an authorization name can be either an individual user or a group.

On operating systems where users and groups exist with the same name, you should specify whether you are granting the privilege to the user or group. Both the GRANT and REVOKE statements support the keywords USER and GROUP. If these optional keywords are not used, the database manager checks the operating system security facility to determine whether the authorization name identifies a user or a group. If the authorization name could be both a user and a group, an error is returned. The following example grants SELECT privileges on the EMPLOYEE table to the user HERON:

```
GRANT SELECT
 ON EMPLOYEE TO USER HERON
```

The following example grants SELECT privileges on the EMPLOYEE table to the group HERON:

```
GRANT SELECT
   ON EMPLOYEE TO GROUP HERON
```

In the Control Center, you can use the Schema Privileges notebook, the Table Space Privileges notebook, and the View Privileges notebook to grant and revoke privileges for these database objects. To open one of these notebooks, follow these steps:

1. In the Control Center, expand the object tree until you find the folder containing the objects you want to work with, for example, the Views folder.
2. Click the folder.

   Any existing database objects in this folder are displayed in the contents pane.
3. Right-click the object of interest in the contents pane and select **Privileges** in the pop-up menu.

   The appropriate Privileges notebook opens.

Revoking privileges

The REVOKE statement allows authorized users to revoke privileges previously granted to other users.

To revoke privileges on database objects, you must have DBADM authority, SYSADM authority, or CONTROL privilege on that object. Note that holding a privilege WITH GRANT OPTION is not sufficient to revoke that privilege. To revoke CONTROL privilege from another user, you must have SYSADM or DBADM authority. To revoke DBADM authority, you must have SYSADM authority. Privileges can only be revoked on existing objects.

Note: A user without DBADM authority or CONTROL privilege is not able to revoke a privilege that they granted through their use of the WITH GRANT OPTION. Also, there is no cascade on the revoke to those who have received privileges granted by the person being revoked.
If an explicitly granted table (or view) privilege is revoked from a user with DBADM authority, privileges **will not** be revoked from other views defined on that

table. This is because the view privileges are available through the DBADM authority and are not dependent on explicit privileges on the underlying tables.

If a privilege has been granted to both a user and a group with the same name, you must specify the GROUP or USER keyword when revoking the privilege. The following example revokes the SELECT privilege on the EMPLOYEE table from the user HERON:

```
REVOKE SELECT
   ON EMPLOYEE FROM USER HERON
```

The following example revokes the SELECT privilege on the EMPLOYEE table from the group HERON:

```
REVOKE SELECT
   ON EMPLOYEE FROM GROUP HERON
```

Note that revoking a privilege from a group may not revoke it from all members of that group. If an individual name has been directly granted a privilege, it will keep it until that privilege is directly revoked.

If a table privilege is revoked from a user, privileges are also revoked on any view created by that user which depends on the revoked table privilege. However, only the privileges implicitly granted by the system are revoked. If a privilege on the view was granted directly by another user, the privilege is still held.

If a table privilege is revoked from a user, privileges are also revoked on any view created by that user which depends on the revoked table privilege. However, only the privileges implicitly granted by the system are revoked. If a privilege on the view was granted directly by another user, the privilege is still held.

You may have a situation where you want to GRANT a privilege to a group and then REVOKE the privilege from just one member of the group. There are only a couple of ways to do that without receiving the error message SQL0556N:

- You can remove the member from the group; or, create a new group with fewer members and GRANT the privilege to the new group.
- You can REVOKE the privilege from the group and then GRANT it to individual users (authorization IDs).

Note: When CONTROL privilege is revoked from a user on a table or a view, the user continues to have the ability to grant privileges to others. When given CONTROL privilege, the user also receives all other privileges WITH GRANT OPTION. Once CONTROL is revoked, all of the other privileges remain WITH GRANT OPTION until they are explicitly revoked.

All packages that are dependent on revoked privileges are marked invalid, but can be validated if rebound by a user with appropriate authority. Packages can also be rebuilt if the privileges are subsequently granted again to the binder of the application; running the application will trigger a successful implicit rebind. If privileges are revoked from PUBLIC, all packages bound by users having only been able to bind based on PUBLIC privileges are invalidated. If DBADM authority is revoked from a user, all packages bound by that user are invalidated including those associated with database utilities. Attempting to use a package that has been marked invalid causes the system to attempt to rebind the package. If this rebind attempt fails, an error occurs (SQLCODE -727). In this case, the packages must be explicitly rebound by a user with:

- Authority to rebind the packages

- Appropriate authority for the objects used within the packages

These packages should be rebound at the time the privileges are revoked.

If you define a trigger or SQL function based on one or more privileges and you lose one or more of these privileges, the trigger or SQL function cannot be used.

Managing implicit authorizations by creating and dropping objects

The database manager implicitly grants certain privileges to a user creates a database object such as a table or a package. Privileges are also granted when objects are created by users with SYSADM or DBADM authority. Similarly, privileges are removed when an object is dropped.

When the created object is a table, nickname, index, or package, the user receives CONTROL privilege on the object. When the object is a view, the CONTROL privilege for the view is granted implicitly only if the user has CONTROL privilege for all tables, views, and nicknames referenced in the view definition.

When the object explicitly created is a schema, the schema owner is given ALTERIN, CREATEIN, and DROPIN privileges WITH GRANT OPTION. An implicitly created schema has CREATEIN granted to PUBLIC.

Establishing ownership of a package

The BIND and PRECOMPILE commands create or change an application package. On either one, use the OWNER option to name the owner of the resulting package.

There are simple rules for naming the owner of a package:

- Any user can name themselves as the owner. This is the default if the OWNER option is not specified.
- An ID with SYSADM or DBADM authority can name any authorization ID as the owner using the OWNER option.

Not all operating systems that can bind a package using DB2 database products support the OWNER option.

Implicit privileges through a package

Access to data within a database can be requested by application programs, as well as by persons engaged in an interactive workstation session. A package contains statements that allow users to perform a variety of actions on many database objects. Each of these actions requires one or more privileges.

Privileges granted to individuals binding the package and to PUBLIC, as well as to the roles granted to the individuals and to PUBLIC, are used for authorization checking when static SQL and XQuery statements are bound. Privileges granted through groups and the roles granted to groups are *not* used for authorization checking when static SQL and XQuery statements are bound. The user with a valid *authID* who binds a package must either have been explicitly granted all the privileges required to execute the static SQL or XQuery statements in the package, or have been implicitly granted the necessary privileges through PUBLIC, the roles granted to PUBLIC or the roles granted to the user, unless VALIDATE RUN was specified when binding the package. If VALIDATE RUN was specified at BIND time, all authorization failures for any static SQL or XQuery statements within this package will not cause the BIND to fail, and those SQL or XQuery statements are revalidated at run time. PUBLIC, group, roles, and user privileges *are all* used

when checking to ensure the user has the appropriate authorization (BIND or BINDADD privilege) to bind the package.

Packages may include both static and dynamic SQL and XQuery statements. To process a package with static queries, a user need only have EXECUTE privilege on the package. This user can then implicitly obtain the privileges of the package binder for any static queries in the package but only within the restrictions imposed by the package.

If the package includes dynamic SQL or XQuery statements, the required privileges depend on the value that was specified for DYNAMICRULES when the package was precompiled or bound. For more information, see the topic that describes the effect of DYNAMICRULES on dynamic queries.

Indirect privileges through a package containing nicknames

When a package contains references to nicknames, authorization processing for package creators and package users is slightly more complex. When a package creator successfully binds packages that contain nicknames, the package creator does not have to pass authentication checking or privilege checking for the tables and views that the nicknames reference at the data source. However, the package executor must pass authentication and authorization checking at data sources.

For example, assume that a package creator's .SQC file contains several SQL or XQuery statements. One static statement references a local table. Another dynamic statement references a nickname. When the package is bound, the package creator's authid is used to verify privileges for the local table and the nickname, but no checking is done for the data source objects that the nickname identifies. When another user executes the package, assuming they have the EXECUTE privilege for that package, that user does not have to pass any additional privilege checking for the statement referencing the table. However, for the statement referencing the nickname, the user executing the package must pass authentication checking and privilege checking at the data source.

When the .SQC file contains only dynamic SQL and XQuery statements and a mixture of table and nickname references, DB2 database authorization checking for local objects and nicknames is similar. Package users must pass privilege checking for any local objects (tables, views) within the statements and also pass privilege checking for nickname objects (package users must pass authentication and privilege checking at the data source containing the objects that the nicknames identify). In both cases, users of the package must have the EXECUTE privilege.

The ID and password of the package executor is used for all data source authentication and privilege processing. This information can be changed by creating a user mapping.

Note: Nicknames cannot be specified in static SQL and XQuery statements. Do not use the DYNAMICRULES option (set to BIND) with packages containing nicknames.

It is possible that packages containing nicknames might require additional authorization steps because DB2 database uses dynamic SQL when communicating with DB2 Family data sources. The authorization ID running the package at the data source must have the appropriate authority to execute the package dynamically at that data source.

Controlling access to data with views

A view provides a means of controlling access or extending privileges to a table.

Using a view allows the following kinds of control over access to a table:

- Access only to designated columns of the table.

  For users and application programs that require access only to specific columns of a table, an authorized user can create a view to limit the columns addressed only to those required.
- Access only to a subset of the rows of the table.

  By specifying a WHERE clause in the subquery of a view definition, an authorized user can limit the rows addressed through a view.
- Access only to a subset of the rows or columns in data source tables or views. If you are accessing data sources through nicknames, you can create local DB2 database views that reference nicknames. These views can reference nicknames from one or many data sources.

  Note: Because you can create a view that contains nickname references for more than one data source, your users can access data in multiple data sources from one view. These views are called *multi-location views*. Such views are useful when joining information in columns of sensitive tables across a distributed environment or when individual users lack the privileges needed at data sources for specific objects.

To create a view, a user must have SYSADM authority, DBADM authority, or CONTROL or SELECT privilege for each table, view, or nickname referenced in the view definition. The user must also be able to create an object in the schema specified for the view. That is, CREATEIN privilege for an existing schema or IMPLICIT_SCHEMA authority on the database if the schema does not already exist.

If you are creating views that reference nicknames, you do not need additional authority on the data source objects (tables and views) referenced by nicknames in the view; however, users of the view must have SELECT authority or the equivalent authorization level for the underlying data source objects when they access the view.

If your users do not have the proper authority at the data source for underlying objects (tables and views), you can:

1. Create a data source view over those columns in the data source table that are OK for the user to access
2. Grant the SELECT privilege on this view to users
3. Create a nickname to reference the view

Users can then access the columns by issuing a SELECT statement that references the new nickname.

The following scenario provides a more detailed example of how views can be used to restrict access to information.

Many people might require access to information in the STAFF table, for different reasons. For example:

- The personnel department needs to be able to update and look at the entire table.

This requirement can be easily met by granting SELECT and UPDATE privileges on the STAFF table to the group PERSONNL:

```
GRANT SELECT,UPDATE ON TABLE STAFF TO GROUP PERSONNL
```

- Individual department managers need to look at the salary information for their employees.

  This requirement can be met by creating a view for each department manager. For example, the following view can be created for the manager of department number 51:

  ```
  CREATE VIEW EMP051 AS
     SELECT NAME,SALARY,JOB FROM STAFF
     WHERE DEPT=51
  GRANT SELECT ON TABLE EMP051 TO JANE
  ```

  The manager with the authorization name JANE would query the EMP051 view just like the STAFF table. When accessing the EMP051 view of the STAFF table, this manager views the following information:

NAME	SALARY	JOB
Fraye	45150.0	Mgr
Williams	37156.5	Sales
Smith	35654.5	Sales
Lundquist	26369.8	Clerk
Wheeler	22460.0	Clerk

- All users need to be able to locate other employees. This requirement can be met by creating a view on the NAME column of the STAFF table and the LOCATION column of the ORG table, and by joining the two tables on their respective DEPT and DEPTNUMB columns:

  ```
  CREATE VIEW EMPLOCS AS
     SELECT NAME, LOCATION FROM STAFF, ORG
     WHERE STAFF.DEPT=ORG.DEPTNUMB
  GRANT SELECT ON TABLE EMPLOCS TO PUBLIC
  ```

  Users who access the employee location view will see the following information:

NAME	LOCATION
Molinare	New York
Lu	New York
Daniels	New York
Jones	New York
Hanes	Boston
Rothman	Boston
Ngan	Boston
Kermisch	Boston
Sanders	Washington
Pernal	Washington
James	Washington
Sneider	Washington
Marenghi	Atlanta
O'Brien	Atlanta
Quigley	Atlanta

NAME	LOCATION
Naughton	Atlanta
Abrahams	Atlanta
Koonitz	Chicago
Plotz	Chicago
Yamaguchi	Chicago
Scoutten	Chicago
Fraye	Dallas
Williams	Dallas
Smith	Dallas
Lundquist	Dallas
Wheeler	Dallas
Lea	San Francisco
Wilson	San Francisco
Graham	San Francisco
Gonzales	San Francisco
Burke	San Francisco
Quill	Denver
Davis	Denver
Edwards	Denver
Gafney	Denver

Controlling access by users holding SYSADM and DBADM authority

You may want to monitor or control access to data by users holding SYSADM and DBADM authorities.

To monitor and control access by system administrators and database administrators, follow these steps:

1. Create an audit policy that monitors the events you want to capture for users who hold SYSADM and DBADM authority.
2. Associate this audit policy with the SYSADM authority and the DBADM authority.
3. Create a role and grant DBADM authority to that role.
4. Define a trusted context and make the role the default role for this trusted context.

   Do not grant membership in the role to any authorization ID explicitly. This way, the role is available only through this trusted context and a user acquires DBADM capability only when they are within the confines of the trusted context.

   Note: This option does not protect against users who have SYSADM authority, because such users have implicit DBADM authority.
5. There are two ways you can control how users access the trusted context:
   - Implicit access: Create a unique trusted context for each user. When the user establishes a regular connection that matches the attributes of the trusted context, they are implicitly trusted and gain access to the role.

- Explicit access: Create a trusted context using the WITH USE FOR clause to define all users who can access it. Create an application through which those users can make database requests. The application establishes an explicit trusted connection, and when a user issues a request, the application switches to that user ID and executes the request as that user on the database.

6. Create an audit policy that monitors the events you want to capture for users of the trusted context, and associate it with the trusted context.
7. If you have highly sensitive data, create an audit policy that monitors the EXECUTE category and associate this policy with the tables containing the sensitive data that you want to monitor. The EXECUTE category captures all queries that access these tables, regardless of who issues them.

Note: To explicitly prevent users who hold SYSADM and DBADM authority from accessing data in tables, consider using the LBAC (label-based access control) security mechanism on the sensitive tables.

Data encryption

To encrypt data in storage (described here), you can use encryption and decryption built-in functions: ENCRYPT, DECRYPT_BIN, DECRYPT_CHAR, and GETHINT. To encrypt data in transit between clients and DB2 databases, you can use the DATA_ENCRYPT and SERVER_ENCRYPT authentication types, or, the DB2 database system support of Secure Socket Layer (SSL).

The ENCRYPT built-in function encrypts data using a password-based encryption method. These functions also allow you to encapsulate a password hint. The password hint is embedded in the encrypted data. Once encrypted, the only way to decrypt the data is by using the correct password. Developers that choose to use these functions should plan for the management of forgotten passwords and unusable data.

The result of the ENCRYPT functions is VARCHAR FOR BIT DATA (with a limit of 32631).

Only CHAR, VARCHAR, and FOR BIT DATA can be encrypted.

The DECRYPT_BIN and DECRYPT_CHAR functions decrypt data using password-based decryption.

DECRYPT_BIN always returns VARCHAR FOR BIT DATA while DECRYPT_CHAR always returns VARCHAR. Since the first argument may be CHAR FOR BIT DATA or VARCHAR FOR BIT DATA, there are cases where the result is not the same as the first argument.

The length of the result depends on the bytes to the next 8 byte boundary. The length of the result could be the length of the data argument plus 40 plus the number of bytes to the next 8 byte boundary when the optional hint parameter is specified. Or, the length of the result could be the length of the data argument plus 8 plus the number of bytes to the next 8 byte boundary when the optional hint parameter is not specified.

The GETHINT function returns an encapsulated password hint. A password hint is a phrase that will help data owners remember passwords. For example, the word “Ocean” can be used as a hint to remember the password ″Pacific″.

The password that is used to encrypt the data is determined in one of two ways:

- Password Argument. The password is a string that is explicitly passed when the ENCRYPT function is invoked. The data is encrypted and decrypted with the given password.
- Encryption password special register. The SET ENCRYPTION PASSWORD statement encrypts the password value and sends the encrypted password to the database manager to store in a special register. ENCRYPT, DECRYPT_BIN and DECRYPT_CHAR functions invoked without a password parameter use the value in the ENCRYPTION PASSWORD special register. The ENCRYPTION PASSWORD special register is only stored in encrypted form.

  The initial or default value for the special register is an empty string.

Valid lengths for passwords are between 6 and 127 inclusive. Valid lengths for hints are between 0 and 32 inclusive.

Configuring Secure Socket Layer (SSL) support in a DB2 instance

The DB2 database system supports SSL, which means that a client application that uses the IBM DB2 Driver for JDBC and SQLJ can connect to a DB2 database using an SSL socket. To enable SSL support in a DB2 instance, set the **DB2COMM** registry variable to SSL, create a SSL configuration file, and restart the instance.

Before configuring SSL support:

- Ensure that the path to the GSKit libraries appears in the **PATH** environment variable on Windows and the **LIBPATH**, **SHLIB_PATH** or **LD_LIBRARY_PATH** environment variables on Linux and UNIX.
- Ensure that the connection concentrator is not activated. SSL support will not be enabled in the DB2 instance if connection concentrator is running.

  To determine whether connection concentrator is activated, issue the GET DATABASE MANAGER CONFIGURATION command. If the configuration parameter **MAX_CONNECTIONS** is set to a value greater than the value of **MAX_COORDAGENTS**, connection concentrator is activated.

SSL is supported for communication between IBM DB2 Driver for JDBC and SQLJ (type 4 connections) and DB2 database products.

The supported platforms that contain SSL support for the DB2 data server are:

- AIX
- HP-UX on Itanium-based HP Integrity Series systems (IA-64)
- Linux on x86, x64, IA64, 64-bit POWER™ servers, and 64-bit zSeries or System z9™
- Solaris on x64
- Windows on 32-bit, x64 and Itanium-based systems

The SSL communication will always be in FIPS mode.

If you are using DB2 Connect or DB2 Enterprise Edition on an intermediate server machine to connect DB2 clients to a host or iSeries™ database, SSL support is not available between the gateway DB2 database product and the host or iSeries database. SSL support is available, however, between the IBM DB2 Driver for JDBC and SQLJ (type 4 connectivity) on the DB2 client and the gateway DB2 database product in that scenario.

To configure SSL support in a DB2 instance:

1. Log in as the DB2 instance owner.
2. Create an SSL configuration file:
   - Linux and UNIX:*INSTHOME*/cfg/SSLconfig.ini
   - Windows: *INSTHOME*/SSLconfig.ini

   where *INSTHOME* is the home directory of the instance.

   It is recommended that you set the file permission to limit access to the SSLconfig.ini, as the file might contain sensitive data. For example, limit read and write authority on the file to members of the SYSADM group if the file contains the password for KeyStore.
3. Add SSL parameters to the SSL configuration file. The SSLconfig.ini file contains the SSL parameters that are used to load and start SSL. The list of SSL parameters are as follows:

Table 3. SSL parameters in the SSL configuration file

SSL parameter name	Description
DB2_SSL_KEYSTORE_FILE	Fully qualified file name of the KeyStore that stores the Server Certificate.
DB2_SSL_KEYSTORE_PW	Password of the KeyStore that stores the Server Certificate.
DB2_SSL_KEYSTORE_LABEL	Label for the Server Certificate.
DB2_SSL_LISTENER	Service name or port number for the SSL listener.

   Note:
   - **DB2_SSL_KEYSTORE_PW** is nullable and can be omitted if a password is not needed for the KeyStore file.
   - If the **DB2_SSL_KEYSTORE_LABEL** parameter is omitted, the default server certificate will be used. If the default server certificate does not exist, SSL setup will fail.
   - The value used for the **DB2_SSL_LISTENER** parameter must be different than the value used in the **SVCENAME** database manager configuration parameter. An SQL5043N error will occur if you try to start the DB2 instance and both SSL and TCP/IP are listening on the same port number.

   The following is an example of an SSLconfig.ini file:

```
DB2_SSL_KEYSTORE_FILE=/home/test1/GSKit/Keystore/key.kdb
DB2_SSL_LISTENER=20397
DB2_SSL_KEYSTORE_PW=aaa111
```

4. Add the value SSL to the **DB2COMM** registry variable. For example:

```
db2set -i db2inst1 DB2COMM=SSL
```

   where *db2inst1* is the DB2 instance name. The database manager can support multiple protocols at the same time. For example, to enable both TCP/IP and SSL communication protocols:

```
db2set -i db2inst1 DB2COMM=SSL,TCPIP
```

5. Restart the DB2 instance. For example:

```
db2stop
db2start
```

Auditing DB2 activities

Introduction to the DB2 audit facility

To manage access to your sensitive data, you can use a variety of authentication and access control mechanisms to establish rules and controls for known and acceptable data access behaviors. But to protect against and discover unknown or unacceptable behaviors you also need to monitor data access. To assist you in this task, the DB2 database system provides an audit facility.

Successful monitoring of unwanted data access and subsequent analysis can lead to improvements in the control of data access and the ultimate prevention of malicious or careless unauthorized access to the data. The monitoring of application and individual user access, including system administration actions, can provide a historical record of activity on your database systems.

The DB2 audit facility generates, and allows you to maintain, an audit trail for a series of predefined database events. The records generated from this facility are kept in an audit log file. The analysis of these records can reveal usage patterns which would identify system misuse. Once identified, actions can be taken to reduce or eliminate such system misuse.

The audit facility provides the ability to audit at both the instance and the individual database level, independently recording all instance and database level activities with separate logs for each. The system administrator (who holds SYSADM authority at the instance level) can use the db2audit tool to configure audit at the instance level as well as to control when such audit information is collected. The system administrator can use the db2audit tool to archive both instance and database audit logs as well as to extract audit data from archived logs of either type.

The security administrator (who holds SECADM authority at the database level) can use audit policies in conjunction with the SQL statement, AUDIT, to configure and control the audit requirements for an individual database. The security administrator can use the SYSPROC.AUDIT_ARCHIVE stored procedure, the SYSPROC.AUDIT_LIST_LOGS table function, and the SYSPROC.AUDIT_DELIM_EXTRACT stored procedure, to archive audit logs, locate logs of interest, and extract data into delimited files for analysis.

When working in a partitioned database environment, many of the auditable events occur at the database partition at which the user is connected (the coordinator partition) or at the catalog partition (if they are not the same database partition). The implication of this is that audit records can be generated by more than one database partition. Part of each audit record contains information identifying the coordinator partition and originating partition (the partition where audit record originated).

At the instance level, the audit facility must be stopped and started explicitly by use of the db2audit start and db2audit stop commands. When you start instance-level auditing, the audit facility uses existing audit configuration information. Since the audit facility is independent of the DB2 database server, it will remain active even if the instance is stopped. In fact, when the instance is stopped, an audit record may be generated in the audit log. To start auditing at the database level, you associate an audit policy with whatever object you want to audit.

Categories of audit records

There are different categories of audit records that may be generated. In the following description of the categories of events available for auditing, you should notice that following the name of each category is a one-word keyword used to identify the category type. The categories of events available for auditing are:

- Audit (AUDIT). Generates records when audit settings are changed or when the audit log is accessed.
- Authorization Checking (CHECKING). Generates records during authorization checking of attempts to access or manipulate DB2 database objects or functions.
- Object Maintenance (OBJMAINT). Generates records when creating or dropping data objects, and when altering certain objects.
- Security Maintenance (SECMAINT). Generates records when:
  - Granting or revoking database privileges or database authorities
  - Granting or revoking security labels or exemptions
  - Altering the group authorization, role authorization, or override or restrict attributes of an LBAC security policy
  - Granting or revoking the SETSESSIONUSER privilege
  - Granting or revoking DBADM or SECADM authorities
  - Modifying any of the SYSADM_GROUP, SYSCTRL_GROUP, SYSMAINT_GROUP, or SYSMON_GROUP configuration parameters.
- System Administration (SYSADMIN). Generates records when operations requiring SYSADM, SYSMAINT, or SYSCTRL authority are performed.
- User Validation (VALIDATE). Generates records when authenticating users or retrieving system security information.
- Operation Context (CONTEXT). Generates records to show the operation context when a database operation is performed. This category allows for better interpretation of the audit log file. When used with the log's event correlator field, a group of events can be associated back to a single database operation. For example, a query statement for dynamic queries, a package identifier for static queries, or an indicator of the type of operation being performed, such as CONNECT, can provide needed context when analyzing audit results.

  Note: The SQL or XQuery statement providing the operation context might be very long and is completely shown within the CONTEXT record. This can make the CONTEXT record very large.
- Execute (EXECUTE). Generates records during the execution of SQL statements.

For any of the above categories, you can audit failures, successes, or both.

Any operations on the database server may generate several records. The actual number of records generated in the audit log depends on the number of categories of events to be recorded as specified by the audit facility configuration. It also depends on whether successes, failures, or both, are audited. For this reason, it is important to be selective of the events to audit.

Audit policies

The security administrator can use audit policies to configure the audit system to gather information only about the data and objects that are needed.

The security administrator can create audit policies to control what is audited within an individual database. The following objects can have an audit policy associated with them:

- The whole database

  All auditable events that occur within the database are audited according to the audit policy.
- Tables

  All data manipulation language (DML) and XQUERY access to the table (untyped), MQT (materialized query table), or nickname is audited. Only EXECUTE category audit events with or without data are generated when the table is accessed even if the policy indicates that other categories should be audited.
- Trusted contexts

  All auditable events that happen within a trusted connection defined by the particular trusted context are audited according to the audit policy.
- Authorization IDs representing users, groups, or roles

  All auditable events that are initiated by the specified user are audited according to the audit policy.

  All auditable events that are initiated by users that are a member of the group or role are audited according to the audit policy. Indirect role membership, such as through other roles or groups, is also included.

  You can capture similar data by using the Work Load Management event monitors by defining a work load for a group and capturing the activity details. You should be aware that the mapping to workloads can involve attributes in addition to just the authorization ID, which can cause you to not achieve the desired granularity in auditing, or if those other attributes are modified, connections may map to different (possibly unmonitored) workloads. The auditing solution provides a guarantee that a user, group or role will be audited.
- Authorities (SYSADM, SECADM, DBADM, SYSCTRL, SYSMAINT, SYSMON)

  All auditable events that are initiated by a user that holds the specified authority even if that authority is unnecessary for the event are audited according to the audit policy. If an audit policy is associated with DBADM authority, any user with SYSADM authority is also audited according to this policy, because they are considered to have DBADM authority.

The security administrator can create multiple audit policies. For example, your company might want a policy for auditing sensitive data and a policy for auditing the activity of users holding DBADM authority. If multiple audit policies are in effect for a statement, all events required to be audited by each of the audit policies are audited (but audited only once). For example, if the database's audit policy requires auditing successful EXECUTE events for a particular table and the user's audit policy requires auditing failures of EXECUTE events for that same table, both successful and failed attempts at accessing that table are audited.

For a specific object, there can only be one audit policy in effect. For example, you cannot have multiple audit policies associated with the same table at the same time.

An audit policy cannot be associated with a view or a typed table. Views that access a table that has an associated audit policy are audited according to the underlying table's policy.

The audit policy that applies to a table does not automatically apply to a MQT based on that table. If you associate an audit policy with a table, associate the same policy with any MQT based on that table.

Auditing performed during a transaction is done based on the audit policies and their associations at the start of the transaction. For example, if the security administrator associates an audit policy with a user and that user is in a transaction at the time, the audit policy does not affect any remaining statements performed within that transaction. Also, changes to an audit policy do not take effect until they are committed. If the security administrator issues an ALTER AUDIT POLICY statement, it does not take effect until the statement is committed.

The security administrator uses the CREATE AUDIT POLICY statement to create an audit policy, and the ALTER AUDIT POLICY statement to modify an audit policy. These statements can specify:

- The status values for events to be audited: None, Success, Failure, or Both. Only auditable events that match the specified status value are audited.
- The server behavior when errors occur during auditing.

The security administrator uses the AUDIT statement to associate an audit policy with the current database or with a database object, at the current server. Any time the object is in use, it is audited according to this audit policy.

To delete an audit policy, the security administrator uses the DROP statement. You cannot drop an audit policy if it is associated with any object. Use the AUDIT REMOVE statement to remove any remaining association with an object. To add metadata to an audit policy, the security administrator uses the COMMENT statement.

Events generated before a full connection has been established

For some events generated during connect and a switch user operation, the only audit policy information available is the policy that is associated with the database. These events are shown in the following table:

Table 4. Connection events

Event	Audit category	Comment
CONNECT	CONTEXT	
CONNECT_RESET	CONTEXT	
AUTHENTICATION	VALIDATE	This includes authentication during both connect and switch user within a trusted connection.
CHECKING_FUNC	CHECKING	The access attempted is SWITCH_USER.

These events are audited based only on the audit policy associated with the database and not with audit policies associated with any other object such as a user, their groups, or authorities. For the CONNECT and AUTHENTICATION events that occur during connect, the instance-level audit settings are used until the database is activated. The database is activated either during the first connection or when the ACTIVATE DATABASE command is issued.

Effect of switching user

If a user is switched within a trusted connection, no remnants of the original user are left behind. In this case, the audit policies associated with the original user are

no longer considered, and the applicable audit policies are re-evaluated according to the new user. Any audit policy associated with the trusted connection is still in effect.

If a SET SESSION USER statement is used, only the session authorization ID is switched. The audit policy of the authorization ID of the original user (the system authorization ID) remains in effect and the audit policy of the new user is used as well. If multiple SET SESSION USER statements are issued within a session, only the audit policies associated with the original user (the system authorization ID) and the current user (the session authorization ID) are considered.

Data definition language restrictions

The following data definition language (DDL) statements are called AUDIT exclusive SQL statements:

- AUDIT
- CREATE AUDIT POLICY, ALTER AUDIT POLICY, and DROP AUDIT POLICY
- DROP ROLE and DROP TRUSTED CONTEXT, if the role or trusted context being dropped is associated with an audit policy

AUDIT exclusive SQL statements have some restrictions in their use:

- Each statement must be followed by a COMMIT or ROLLBACK.
- These statements cannot be issued within a global transaction, for example an XA transaction.

Only one uncommitted AUDIT exclusive DDL statement is allowed at a time across all partitions. If an uncommitted AUDIT exclusive DDL statement is executing, subsequent AUDIT exclusive DDL statements wait until the current AUDIT exclusive DDL statement commits or rolls back.

Note: Changes are written to the catalog, but do not take effect until COMMIT, even for the connection that issues the statement.

Example of auditing any access to a specific table

Consider a company where the EMPLOYEE table contains extremely sensitive information and the company wants to audit any and all SQL access to the data in that table. The EXECUTE category can be used to track all access to a table; it audits the SQL statement, and optionally the input data value provided at execution time for that statement.

There are two steps to track activity on the table. First, the security administrator creates an audit policy that specifies the EXECUTE category, and then the security administrator associates that policy with the table:

```
CREATE AUDIT POLICY SENSITIVEDATAPOLICY
    CATEGORIES EXECUTE STATUS BOTH ERROR TYPE AUDIT
COMMIT

AUDIT TABLE EMPLOYEE USING POLICY SENSITIVEDATAPOLICY
COMMIT
```

Example of auditing any actions by SYSADM or DBADM

In order to complete their security compliance certification, a company must show that any and all activities within the database by those people holding system administration (SYSADM) or database administrative (DBADM) authority can be monitored.

To capture all actions within the database, both the EXECUTE and SYSADMIN categories should be audited. The security administrator creates an audit policy that audits these two categories. The security administrator can use the AUDIT statement to associate this audit policy with the SYSADM and DBADM authorities. Any user that holds either SYSADM or DBADM authority will then have any auditable events logged. The following example shows how to create such an audit policy and associate it with the SYSADM and DBADM authorities:

```
CREATE AUDIT POLICY ADMINSPOLICY CATEGORIES EXECUTE STATUS BOTH,
   SYSADMIN STATUS BOTH ERROR TYPE AUDIT
COMMIT
AUDIT SYSADM, DBADM USING POLICY ADMINSPOLICY
COMMIT
```

Example of auditing any access by a specific role

A company has allowed its web applications access to their corporate database. The exact individuals using the web applications are unknown. Only the role that is used is known and that role is used to manage the database authorizations. The company wants to monitor the actions of anyone who is a member of that role in order to examine the requests they are submitting to the database and to ensure that they only access the database through the web applications.

The EXECUTE category contains the necessary level of auditing to track the activity of the users for this situation. The first step is to create the appropriate audit policy and associate it with the roles that are used by the web applications (in this example, the roles are TELLER and CLERK):

```
CREATE AUDIT POLICY WEBAPPPOLICY CATEGORIES EXECUTE WITH DATA
   STATUS BOTH ERROR TYPE AUDIT
COMMIT
AUDIT ROLE TELLER, ROLE CLERK USING POLICY WEBAPPPOLICY
COMMIT
```

Storage and analysis of audit logs

The system administrator can configure the path for the active audit log and the archived audit log using the db2audit configure command. Archiving the audit log moves the active audit log to an archive directory while the server begins writing to a new, active audit log. This allows the audit log to be stored offline without having to extract data from it until necessary. After the security administrator or system administrator has archived a log, they can extract data from the log into delimited files. The data in the delimited files can be loaded into DB2 database tables for analysis.

Configuring the location of the audit logs allows you to place the audit logs on a large, high-speed disk, with the option of having separate disks for each node in a database partitioning feature (DPF) installation. In a DPF environment, the path for the active audit log can be a directory that is unique to each node. Having a unique directory for each node helps to avoid file contention, because each node is writing to a different disk.

The default path for the audit logs on Windows operating systems is *instance*\security\auditdata and on Linux and UNIX operating systems is *instance*/security/auditdata. If you do not want to use the default location, you can choose different directories (you can create new directories on your system to use as alternative locations, if they do not already exist). To set the path for the active audit log location and the archived audit log location, use the db2audit configure command with the datapath and archivepath parameters, as shown in this example:

```
db2audit configure datapath /auditlog archivepath /auditarchive
```

The audit log storage locations you set using db2audit apply to all databases in the instance.

Note: If there are multiple instance on the server, then each instance should each have separate data and archive paths.

The path for active audit logs (datapath) in a DPF environment

In a DPF environment, the same active audit log location (set by the **datapath** parameter) must be used on each partition. There are two ways to accomplish this:

1. Use database partition expressions when you specify the **datapath** parameter. Using database partition expressions allows the partition number to be included in the path of the audit log files and results in a different path on each database partition.
2. Use a shared drive that is the same on all nodes.

You can use database partition expressions anywhere within the value you specify for the **datapath** parameter. For example, on a three node system, where the database partition number is 10, the following command:

```
db2audit configure datapath '/pathForNode $N'
```

creates the following files:

- /pathForNode10
- /pathForNode20
- /pathForNode30

Note: You cannot use database partition expressions to specify the archive log file path (**archivepath** parameter).

Archiving active audit logs

The system administrator can use the db2audit tool to archive both instance and database audit logs as well as to extract audit data from archived logs of either type. To archive the active audit log, the security administrator can use the SYSPROC.AUDIT_ARCHIVE stored procedure. To extract data from the log and load it into delimited files, the security administrator can use the SYSPROC.AUDIT_DELIM_EXTRACT stored procedure.

These are the steps a security administrator needs to follow to archive and extract the audit logs:

1. Schedule an application to perform regular archives of the active audit log using the stored procedure SYSPROC.AUDIT_ARCHIVE.

2. Determine which archived log files are of interest. Use the SYSPROC.AUDIT_LIST_LOGS table function to list all of the archived audit logs.
3. Pass the file name as a parameter to the SYSPROC.AUDIT_DELIM_EXTRACT stored procedure to extract data from the log and load it into delimited files.
4. Load the audit data into DB2 database tables for analysis.

The archived log files do not need to be immediately loaded into tables for analysis; they can be saved for future analysis. For example, they may only need to be looked at when a corporate audit is taking place.

If a problem occurs during archive, such as running out of disk space in the archive path, or the archive path does not exist, the archive process fails and an interim log file with the file extension .bk is generated in the audit log data path, for example, db2audit.instance.log.0.20070508172043640941.bk. After the problem is resolved (by allocating sufficient disk space in the archive path, or by creating the archive path) you must move this interim log to the archive path. Then, you can treat it in the same way as a successfully archived log.

Archiving active audit logs in a DPF environment

In a DPF environment, if the archive command is issued while the instance is running, the archive process automatically runs on every node. The same timestamp is used in the archived log file name on all nodes. For example, on a three node system, where the database partition number is 10, the following command:

```
db2audit archive to /auditarchive
```

creates the following files:

- /auditarchive/db2audit.log.10.*timestamp*
- /auditarchive/db2audit.log.20.*timestamp*
- /auditarchive/db2audit.log.30.*timestamp*

If the archive command is issued while the instance is not running, you can control on which node the archive is run by one of the following methods:

- Use the node option with the db2audit command to perform the archive for the current node only.
- Use the db2_all command to run the archive on all nodes.

  For example:

  ```
  db2_all db2audit archive node to /auditarchive
  ```

  This sets the DB2NODE environment variable to indicate on which nodes the command is invoked.

Alternatively, you can issue an individual archive command on each node separately. For example:

- On node 10:

  ```
  db2audit archive node 10 to /auditarchive
  ```
- On node 20:

  ```
  db2audit archive node 20 to /auditarchive
  ```
- On node 30:

  ```
  db2audit archive node 30 to /auditarchive
  ```

Note: When the instance is not running, the timestamps in the archived audit log file names are not the same on each node.

Note: It is recommended that the archive path is shared across all nodes, but it is not required.

Note: The AUDIT_DELIM_EXTRACT stored procedure and AUDIT_LIST_LOGS table function can only access the archived log files that are visible from the current (coordinator) node.

Example of archiving a log and extracting data to a table

To ensure their audit data is captured and stored for future use, a company needs to create a new audit log every six hours and archive the current audit log to a WORM drive. The company schedules the following call to the SYSPROC.AUDIT_ARCHIVE stored procedure to be issued by the security administrator every six hours. The path to the archived log is the default archive path, /auditarchive, and the archive runs on all nodes:

```
CALL SYSPROC.AUDIT_ARCHIVE( '/auditarchive', -2 )
```

As part of their security procedures, the company has identified and defined a number of suspicious behaviors or disallowed activities that it needs to watch for in the audit data. They want to extract all the data from the one or more audit logs, place it in a relational table, and then use SQL queries to look for these activities. The company has decided on appropriate categories to audit and has associated the necessary audit policies with the database or other database objects.

For example, they can call the SYSPROC.AUDIT_DELIM_EXTRACT stored procedure to extract the archived audit logs for all categories from all nodes that were created with a timestamp in April 2006, using the default delimiter:

```
CALL SYSPROC.AUDIT_DELIM_EXTRACT(
     '', '', '/auditarchive',  'db2audit.%.200604%', '' )
```

In another example, they can call the SYSPROC.AUDIT_DELIM_EXTRACT stored procedure to extract the archived audit records with success events from the EXECUTE category and failure events from the CHECKING category, from a file with the timestamp they are interested in:

```
CALL SYSPROC.AUDIT_DELIM_EXTRACT( '', '', '/auditarchive',
    'db2audit.%.20060419034937', 'categories
     execute status success, checking status failure );
```

Audit log file names:

The audit log files have names that distinguish whether they are instance-level or database-level logs and which partition they originate from in a database partitioning feature (DPF) environment. Archived audit logs have the timestamp of when the archive command was run appended to their file name.

Active audit log file names

In a DPF environment, the path for the active audit log can be a directory that is unique to each partition so that each partition writes to an individual file. In order to accurately track the origin of audit records, the partition number is included as part of the audit log file name. For example, on partition 20, the instance level audit log file name is db2audit.instance.log.20. For a database called testdb in this instance, the audit log file is db2audit.db.testdb.log.20.

In a non-DPF environment the partition number is considered to be 0 (zero). In this case, the instance level audit log file name is db2audit.instance.log.0. For a database called testdb in this instance, the audit log file is db2audit.db.testdb.log.0.

Archived audit log file names

When the active audit log is archived, the current timestamp in the following format is appended to the filename: *YYYYMMDDHHMMSS* (where YYYY is the year, MM is the month, DD is the day, HH is the hour, MM is the minutes, and SS is the seconds.

The file name format for an archive audit log depends on the level of the audit log:

instance-level archived audit log
The file name of the instance-level archived audit log is: db2audit.instance.log.*partition*.*YYYYMMDDHHMMSS*.

database-level archived audit log
The file name of the database-level archived audit log is: db2audit.db*database*.log.*partition*.*YYYYMMDDHHMMSS*.

In a non-DPF environment, the value for *partition* is 0 (zero).

The timestamp represents the time that the archive command was run, therefore it does not always precisely reflect the time of the last record in the log. The archived audit log file may contain records with timestamps a few seconds later than the timestamp in the log file name because:

- When the archive command is issued, the audit facility waits for the writing of any in-process records to complete before creating the archived log file.
- In a multi-machine environment, the system time on a remote machine may not be synchronized with the machine where the archive command is issued.

In a DPF environment, if the server is running when archive is run, the timestamp is consistent across partitions and reflects the timestamp generated at the partition at which the archive was performed.

Creating tables to hold the DB2 audit data:

Before you can work with audit data in database tables, you need to create the tables to hold the data. You should consider creating these tables in a separate schema to isolate the data in the tables from unauthorized users.

- See the CREATE SCHEMA statement for the authorities and privileges that you require to create a schema.
- See the CREATE TABLE statement for the authorities and privileges that you require to create a table.
- Decide which table space you want to use to hold the tables. (This topic does not describe how to create table spaces.)

Note: The format of the tables you need to create to hold the audit data may change from release to release. New columns may be added or the size of an existing column may change. The script, db2audit.ddl, creates tables of the correct format to contain the audit records.

The examples that follow show how to create the tables to hold the records from the delimited files. If you want, you can create a separate schema to contain these tables.

If you do not want to use all of the data that is contained in the files, you can omit columns from the table definitions, or bypass creating certain tables, as required. If you omit columns from the table definitions, you must modify the commands that you use to load data into these tables.

1. Issue the db2 command to open a DB2 command window.
2. Optional. Create a schema to hold the tables. For this example, the schema is called AUDIT:

   ```
   CREATE SCHEMA AUDIT
   ```

3. Optional. If you created the AUDIT schema, switch to the schema before creating any tables:

   ```
   SET CURRENT SCHEMA = 'AUDIT'
   ```

4. Run the script, db2audit.ddl, to create the tables that will contain the audit records.

   The script db2audit.ddl is located in the sqllib/misc directory (sqllib\misc on Windows). The script assumes that a connection to the database exists and that an 8K table space is available. The command to run the script is: db2 +o -tf sqllib/misc/db2audit.ddl The tables that the script creates are: AUDIT, CHECKING, OBJMAINT, SECMAINT, SYSADMIN, VALIDATE, CONTEXT and EXECUTE.

5. After you have created the tables, the security administrator can use the SYSPROC.AUDIT_DELIM_EXTRACT stored procedure, or the system administrator can use the db2audit extract command, to extract the audit records from the archived audit log files into delimited files. You can load the audit data from the delimited files into the database tables you just created.

Loading DB2 audit data into tables:

After you have archived and extracted the audit log file into delimited files, and you have created the database tables to hold the audit data, you can load the audit data from the delimited files into the database tables for analysis.

You use the load utility to load the audit data into the tables. Issue a separate load command for each table. If you omitted one or more columns from the table definitions, you must modify the version of the LOAD command that you use to successfully load the data. Also, if you specified a delimiter character other than the default when you extracted the audit data, you must also modify the version of the LOAD command that you use.

1. Issue the db2 command to open a DB2 command window.
2. To load the AUDIT table, issue the following command:

   ```
   LOAD FROM audit.del OF DEL MODIFIED BY DELPRIORITYCHAR LOBSINFILE
             INSERT INTO schema.AUDIT
   ```

   Note: Specify the DELPRIORITYCHAR modifier to ensure proper parsing of binary data.

   Note: Specify the LOBSINFILE option of the LOAD command (due to the restriction that any inline data for large objects must be limited to 32K). In some situations, you might also need to use the LOBS FROM option.

   Note: When specifying the file name, use the fully qualified path name. For example, if you have the DB2 database system installed on the C: drive of a Windows-based computer, you would specify `C:\Program Files\IBM\SQLLIB\instance\security\audit.del` as the fully qualified file name for the audit.del file.

3. To load the CHECKING table, issue the following command:
   ```
   LOAD FROM checking.del OF DEL MODIFIED BY DELPRIORITYCHAR LOBSINFILE
            INSERT INTO schema.CHECKING
   ```
4. To load the OBJMAINT table, issue the following command:
   ```
   LOAD FROM objmaint.del OF DEL MODIFIED BY DELPRIORITYCHAR LOBSINFILE
            INSERT INTO schema.OBJMAINT
   ```
5. To load the SECMAINT table, issue the following command:
   ```
   LOAD FROM secmaint.del OF DEL MODIFIED BY DELPRIORITYCHAR LOBSINFILE
            INSERT INTO schema.SECMAINT
   ```
6. To load the SYSADMIN table, issue the following command:
   ```
   LOAD FROM sysadmin.del OF DEL MODIFIED BY DELPRIORITYCHAR LOBSINFILE
            INSERT INTO schema.SYSADMIN
   ```
7. To load the VALIDATE table, issue the following command:
   ```
   LOAD FROM validate.del OF DEL MODIFIED BY DELPRIORITYCHAR LOBSINFILE
            INSERT INTO schema.VALIDATE
   ```
8. To load the CONTEXT table, issue the following command:
   ```
   LOAD FROM context.del OF DEL MODIFIED BY DELPRIORITYCHAR LOBSINFILE
            INSERT INTO schema.CONTEXT
   ```
9. To load the EXECUTE table, issue the following command:
   ```
   LOAD FROM execute.del OF DEL MODIFIED BY DELPRIORITYCHAR LOBSINFILE
            INSERT INTO schema.EXECUTE
   ```
10. After you finish loading the data into the tables, delete the .del files from the security/auditdata subdirectory of the sqllib directory.
11. When you have loaded the audit data into the tables, you are ready to select data from these tables for analysis.

If you have already populated the tables a first time, and want to do so again, use the INSERT option to have the new table data added to the existing table data. If you want to have the records from the previous db2audit extract operation removed from the tables, load the tables again using the REPLACE option.

Audit archive and extract stored procedures:

The security administrator can use the SYSPROC.AUDIT_ARCHIVE and SYSPROC.AUDIT_DELIM_EXTRACT stored procedures, and the SYSPROC.AUDIT_LIST_LOGS table function to archive audit logs and extract data to delimited files for the database to which the security administrator is currently connected.

The security administrator must be connected to a database in order to use these stored procedures and table function to archive or list that database's audit logs.

If you copy the archived files to another database system, and you want to use the stored procedures to access them, ensure that the database name is the same, or rename the files to include the same database name.

These stored procedures and table function do not archive or list the instance level audit log. The system administrator must use the db2audit command to archive and extract the instance level audit log.

The security administrator can use these stored procedures and table function to perform the following operations:

Table 5. Audit system stored procedures

Stored procedure and table function	Operation	Comments
AUDIT_ARCHIVE	Archives the current audit log.	Takes the archive path as input. If the archive path is not supplied, this stored procedure takes the archive path from the audit configuration file. The archive is run on each node, and a synchronized timestamp is appended to the name of the audit log file.
AUDIT_LIST_LOGS	Returns a list of the archived audit logs at the specified path, for the current database.	
AUDIT_ DELIM_EXTRACT	Extracts data from the binary archived logs and loads it into delimited files.	The extracted audit records are placed in a delimited format suitable for loading into DB2 database tables. The output is placed in separate files, one for each category. In addition, the file **auditlobs** is created to hold any large objects that are included in the audit data. The file names are: • **audit.del** • **checking.del** • **objmaint.del** • **secmaint.del** • **sysadmin.del** • **validate.del** • **context.del** • **execute.del** • **auditlobs** If the files already exist, the output is appended to them. The **auditlobs** file is created if the CONTEXT or EXECUTE categories are extracted. Only archived audit logs for the current database can be extracted. Only files that are visible to the coordinator node are extracted. Only the instance owner can delete archived audit logs.

The EXECUTE category for auditing SQL statements

The EXECUTE category allows you to accurately track the SQL statements a user issues (prior to Version 9.5, you had to use the CONTEXT category to find this information).

This EXECUTE category captures the SQL statement text as well as the compilation environment and other values that are needed to replay the statement at a later date. For example, replaying the statement can show you exactly which rows a SELECT statement returned. In order to re-run a statement, the database tables must first be restored to their state when the statement was issued.

When you audit using the EXECUTE category, the statement text for both static and dynamic SQL is recorded, as are input parameter markers and host variables. You can configure the EXECUTE category to be audited with or without input values.

Note: Global variables are not audited.

The auditing of EXECUTE events takes place at the completion of the event (for SELECT statements this is on cursor close). The status that the event completed with is also stored. Because EXECUTE events are audited at completion, long-running queries do not immediately appear in the audit log.

Note: The preparation of a statement is not considered part of the execution. Most authorization checks are performed at prepare time (for example, SELECT privilege). This means that statements that fail during prepare due to authorization errors do not generate EXECUTE events.

Statement Value Index, Statement Value Type and Statement Value Data fields may be repeated for a given execute record. For the report format generated by the extraction, each record lists multiple values. For the delimited file format, multiple rows are used. The first row has an event type of STATEMENT and no values. Following rows have an event type of DATA, with one row for each data value associated with the SQL statement. You can use the event correlator and application ID fields to link STATEMENT and DATA rows together. The columns Statement Text, Statement Isolation Level, and Compilation Environment Description are not present in the DATA events.

The statement text and input data values that are audited are converted into the database code page when they are stored on disk (all audited fields are stored in the database code page). No error is returned if the code page of the input data is not compatible with the database code page; the unconverted data will be logged instead. Because each database has it's own audit log, databases having different code pages does not cause a problem.

ROLLBACK and COMMIT are audited when executed by the application, and also when issued implicitly as part of another command, such as BIND.

After an EXECUTE event has been audited due to access to an audited table, all statements that affect which other statements are executed within a unit of work, are audited. These statements are COMMIT, ROLLBACK, ROLLBACK TO SAVEPOINT and SAVEPOINT.

Savepoint ID field

You can use the Savepoint ID field to track which statements were affected by a ROLLBACK TO SAVEPOINT statement. An ordinary DML statement (such as SELECT, INSERT, and so on) has the current savepoint ID audited. However, for the ROLLBACK TO SAVEPOINT statement, the savepoint ID that is rolled back to will be audited instead. Therefore, every statement with a savepoint ID greater than or equal to that ID will be rolled back, as demonstrated by the following

example. The table shows the sequence of statements run; all events with a Savepoint ID greater than or equal to 2 will be rolled back. Only the value of 3 (from the first INSERT statement) is inserted into the table T1.

Table 6. Sequence of statements to demonstrate effect of ROLLBACK TO SAVEPOINT statement

Statement	Savepoint ID
INSERT INTO T1 VALUES (3)	1
SAVEPOINT A	2
INSERT INTO T1 VALUES (5)	2
SAVEPOINT B	3
INSERT INTO T1 VALUES (6)	3
ROLLBACK TO SAVEPOINT A	2
COMMIT	

WITH DATA option

Not all input values are audited when you specify the WITH DATA option. LOB, LONG, XML and structured type parameters appear as NULL.

Date, time, and timestamp fields are recorded in ISO format.

If WITH DATA is specified in one policy, but WITHOUT DATA is specified in another policy associated with objects involved in the execution of the SQL statement, then WITH DATA takes precedence and data is audited for that particular statement. For example, if the audit policy associated with a user specifies WITHOUT DATA, but the policy associated with a table specifies WITH DATA, when that user accesses that table, the input data used for the statement is audited.

You are not able to determine which rows were modified on a positioned-update or positioned-delete statement. Only the execution of the underlying SELECT statement is logged, not the individual FETCH. It is not possible from the EXECUTE record to determine which row the cursor is on when the statement is issued. When replaying the statement at a later time, it is only possible to issue the SELECT statement to see what range of rows may have been affected.

Example of replaying past activities

Consider in this example that as part of their comprehensive security policy, a company requires that they retain the ability to retroactively go back up to seven years to analyze the effects of any particular request against certain tables in their database. To do this, they institute a policy of archiving their weekly backups and associated log files such that they can reconstitute the database for any chosen moment in time. They require that the database audit capture sufficient information about every request made against the database to allow the replay and analysis of any request against the relevant, restored database. This requirement covers both static and dynamic SQL statements.

This example shows the audit policy that must be in place at the time the SQL statement is issued, and the steps to archive the audit logs and later to extract and analyze them.

1. Create an audit policy that audits the EXECUTE category and apply this policy to the database:

```
CREATE AUDIT POLICY STATEMENTS CATEGORIES EXECUTE WITH DATA
   STATUS BOTH ERROR TYPE AUDIT
COMMIT

AUDIT DATABASE USING POLICY STATEMENTS
COMMIT
```

2. Regularly archive the audit log to create an archive copy.

   The following statement should be run by the security administrator on a regular basis, for example, once a week or once a day, depending on the amount of data logged. These archived files can be kept for whatever period is required. The procedure, SYSPROC.AUDIT_ARCHIVE, is called with two input parameters: the path to the archive directory and -2, to indicate that the archive should be run on all nodes:

```
CALL SYSPROC.AUDIT_ARCHIVE( '/auditarchive', -2 )
```

3. The security administrator uses the SYSPROC.AUDIT_LIST_LOGS table function to examine all of the available audit logs from April 2006, to determine which logs may contain the necessary data:

```
SELECT FILE FROM TABLE(SYSPROC.AUDIT_LIST_LOGS('/auditarchive'))
   AS T WHERE FILE LIKE 'db2audit.dbname.log.0.200604%'
FILE
------------------------------------
...
db2audit.dbname.log.0.20060418235612
db2audit.dbname.log.0.20060419234937
db2audit.dbname.log.0.20060420235128
```

4. From this output, the security administrator observes that the necessary logs should be in one file: db2audit.dbname.log.20060419234937. The timestamp shows this file was archived at the end of the day for the day the auditors want to see.

   The security administrator uses this filename as input to the SYSPROC.AUDIT_DELIM_EXTRACT stored procedure to extract the audit data into delimited files. The audit data in these files can be loaded into DB2 database tables, where it can be analyzed to find the particular statement the auditors are interested in. Even though the auditors are only interested in a single SQL statement, multiple statements from the unit of work may need to be examined in case they have any impact on the statement of interest.

5. In order to replay the statement, the security administrator must take the following actions:
   - Determine the exact statement to be issued from the audit record.
   - Determine the user who issued the statement from the audit record.
   - Recreate the exact permissions of the user at the time they issued the statement, including any LBAC protection.
   - Reproduce the compilation environment, by using the compilation environment column in the audit record in combination with the SET COMPILATION ENVIRONMENT statement.
   - Restore the database to its exact state at the time the statement was issued.

   To avoid disturbing the production system, any restore of the database and replay of the statement should be done on a second database system. The security administrator, running as the user who issued the statement, can reissue the statement as found in the statement text with any input variables that are provided in the statement value data elements.

Audit facility management

Audit facility behavior

This topic provides background information to help you understand how the timing of writing audit records to the log can affect database performance; how to manage errors that occur within the audit facility; and how audit records are generated in different situations.

Controlling the timing of writing audit records to the active log

The writing of the audit records to the active log can take place synchronously or asynchronously with the occurrence of the events causing the generation of those records. The value of the *audit_buf_sz* database manager configuration parameter determines when the writing of audit records is done.

If the value of *audit_buf_sz* is zero (0), the writing is done synchronously. The event generating the audit record waits until the record is written to disk. The wait associated with each record causes the performance of the DB2 database to decrease.

If the value of *audit_buf_sz* is greater than zero, the record writing is done asynchronously. The value of the *audit_buf_sz* when it is greater than zero is the number of 4 KB pages used to create an internal buffer. The internal buffer is used to keep a number of audit records before writing a group of them out to disk. The statement generating the audit record as a result of an audit event will not wait until the record is written to disk, and can continue its operation.

In the asynchronous case, it could be possible for audit records to remain in an unfilled buffer for some time. To prevent this from happening for an extended period, the database manager forces the writing of the audit records regularly. An authorized user of the audit facility may also flush the audit buffer with an explicit request. Also, the buffers are automatically flushed during an archive operation.

There are differences when an error occurs dependent on whether there is synchronous or asynchronous record writing. In asynchronous mode there may be some records lost because the audit records are buffered before being written to disk. In synchronous mode there may be one record lost because the error could only prevent at most one audit record from being written.

Managing audit facility errors

The setting of the ERRORTYPE audit facility parameter controls how errors are managed between the DB2 database system and the audit facility. When the audit facility is active, and the setting of the ERRORTYPE audit facility parameter is `AUDIT`, then the audit facility is treated in the same way as any other part of DB2 database. An audit record must be written (to disk in synchronous mode; or to the audit buffer in asynchronous mode) for an audit event associated with a statement to be considered successful. Whenever an error is encountered when running in this mode, a negative SQLCODE is returned to the application for the statement generating an audit record.

If the error type is set to `NORMAL`, then any error from db2audit is ignored and the operation's SQLCODE is returned.

Audit records generated in different situations

Depending on the API or query statement and the audit settings, none, one, or several audit records may be generated for a particular event. For example, an SQL UPDATE statement with a SELECT subquery may result in one audit record containing the results of the authorization check for UPDATE privilege on a table and another record containing the results of the authorization check for SELECT privilege on a table.

For dynamic data manipulation language (DML) statements, audit records are generated for all authorization checking at the time that the statement is prepared. Reuse of those statements by the same user will not be audited again since no authorization checking takes place at that time. However, if a change has been made to one of the catalog tables containing privilege information, then in the next unit of work, the statement privileges for the cached dynamic SQL or XQuery statements are checked again and one or more new audit records created.

For a package containing only static DML statements, the only auditable event that could generate an audit record is the authorization check to see if a user has the privilege to execute that package. The authorization checking and possible audit record creation required for the static SQL or XQuery statements in the package is carried out at the time the package is precompiled or bound. The execution of the static SQL or XQuery statements within the package is auditable using the EXECUTE category. When a package is bound again either explicitly by the user, or implicitly by the system, audit records are generated for the authorization checks required by the static SQL or XQuery statements.

For statements where authorization checking is performed at statement execution time (for example, data definition language (DDL), GRANT, and REVOKE statements), audit records are generated whenever these statements are used.

Note: When executing DDL, the section number recorded for all events (except the context events) in the audit record will be zero (0) no matter what the actual section number of the statement might have been.

Audit facility tips and techniques

Archiving the audit log

You should archive the audit log on a regular basis. Archiving the audit log moves the current audit log to an archive directory while the server begins writing to a new, active audit log. The name of each archived log file includes a timestamp that helps you identify log files of interest for later analysis.

For long term storage, you may want to zip up groups of archived files.

For archived audit logs that you are no longer interested in, the instance owner can simply delete the files from the operating system.

Error handling

When you create an audit policy, you should use the error type AUDIT, unless you are just creating a test audit policy. For example, if the error type is set to AUDIT, and an error occurs, such as running out of disk space, then an error is returned. The error condition must be corrected before any more auditable actions can

continue. However, if the error type had been set to NORMAL, the logging would simply fail and no error is returned to the user. Operation continues as if the error did not happen.

If a problem occurs during archive, such as running out of disk space in the archive path, or the archive path does not exist, the archive process fails and an interim log file with the file extension .bk is generated in the audit log data path, for example, db2audit.instance.log.0.20070508172043640941.bk. After the problem is resolved (by allocating sufficient disk space in the archive path, or by creating the archive path) you must move this interim log to the archive path. Then, you can treat it in the same way as a successfully archived log.

DDL statement restrictions

Some data definition language (DDL) statements, called AUDIT exclusive SQL statements, do not take effect until the next unit of work. Therefore, you are advised to use a COMMIT statement immediately after each of these statements.

The AUDIT exclusive SQL statements are:

- AUDIT
- CREATE AUDIT POLICY, ALTER AUDIT POLICY, and DROP AUDIT POLICY
- DROP ROLE and DROP TRUSTED CONTEXT, if the role or trusted context being dropped is associated with an audit policy

Table format for holding archived data may change

The security administrator can use the SYSPROC.AUDIT_DEL_EXTRACT stored procedure, or the system administrator can use the db2audit extract command, to extract audit records from the archived audit log files into delimited files. You can load the audit data from the delimited files into DB2 database tables for analysis. The format of the tables you need to create to hold the audit data may change from release to release.

Important: The script, db2audit.ddl, creates tables of the correct format to contain the audit records. You should expect to run db2audit.ddl for each release, as columns may be added or the size of an existing column may change.

Using CHECKING events

In most cases, when working with CHECKING events, the object type field in the audit record is the object being checked to see if the required privilege or authority is held by the user ID attempting to access the object. For example, if a user attempts to ALTER a table by adding a column, then the CHECKING event audit record indicates the access attempted was “ALTER” and the object type being checked was “TABLE” (not the column, because it is table privileges that are checked).

However, when the checking involves verifying if a database authority exists to allow a user ID to CREATE or BIND an object, or to delete an object, then although there is a check against the database, the object type field will specify the object being created, bound, or dropped (rather than the database itself).

When creating an index on a table, the privilege to create an index is required, therefore the CHECKING event audit record has an access attempt type of “index” rather than “create”.

Audit records created for binding a package

When binding a package that already exists, then an OBJMAINT event audit record is created for the DROP of the package and then another OBJMAINT event audit record is created for the CREATE of the new copy of the package.

Using CONTEXT event information after ROLLBACK

Data Definition Language (DDL) may generate OBJMAINT or SECMAINT events that are logged as successful. It is possible however that following the logging of the event, a subsequent error may cause a ROLLBACK to occur. This would leave the object as not created; or the GRANT or REVOKE actions as incomplete. The use of CONTEXT events becomes important in this case. Such CONTEXT event audit records, especially the statement that ends the event, indicates the nature of the completion of the attempted operation.

The load delimiter

When extracting audit records in a delimited format suitable for loading into a DB2 database table, you should be clear regarding the delimiter used within the statement text field. This can be done when extracting the delimited file, using:

```
db2audit extract delasc delimiter <load delimiter>
```

The *load delimiter* can be a single character (such as ″) or a four-byte string representing a hexadecimal value (such as “0xff”). Examples of valid commands are:

```
db2audit extract delasc
db2audit extract delasc delimiter !
db2audit extract delasc delimiter 0xff
```

If you have used anything other than the default load delimiter as the delimiter when extracting, you should use the MODIFIED BY option on the LOAD command. A partial example of the LOAD command with “0xff” used as the delimiter follows:

```
db2 load from context.del of del modified by chardel0xff replace into ...
```

This will override the default load character string delimiter which is ″ (double quote).

Chapter 2. Roles

Roles simplify the administration and management of privileges by offering an equivalent capability as groups but without the same restrictions. A role is a database object that groups together one or more privileges and can be assigned to users, groups, PUBLIC, or other roles by using a GRANT statement, or can be assigned to a trusted context by using a CREATE TRUSTED CONTEXT or ALTER TRUSTED CONTEXT statement. A role can be specified for the SESSION_USER ROLE connection attribute in a workload definition.

Roles provide several advantages that make it easier to manage privileges in a database system:

- Security administrators can control access to their databases in a way that mirrors the structure of their organizations (they can create roles in the database that map directly to the job functions in their organizations).
- Users are granted membership in the roles that reflect their job responsibilities. As their job responsibilities change, their membership in roles can be easily granted and revoked.
- The assignment of privileges is simplified. Instead of granting the same set of privileges to each individual user in a particular job function, the administrator can grant this set of privileges to a role representing that job function and then grant that role to each user in that job function.
- A role's privileges can be updated and all users who have been granted that role receive the update; the administrator does not need to update the privileges for every user on an individual basis.
- The privileges and authorities granted to roles are always used when you create views, triggers, materialized query tables (MQTs), static SQL and SQL routines, whereas privileges and authorities granted to groups (directly or indirectly) are not used.

  This is because the DB2 database system cannot determine when membership in a group changes, as the group is managed by third-party software (for example, the operating system or an LDAP directory). Because roles are managed inside the database, the DB2 database system can determine when authorization changes and act accordingly. Roles granted to groups are not considered, due to the same reason groups are not considered.
- All the roles assigned to a user are enabled when that user establishes a connection, so all privileges and authorities granted to roles are taken into account when a user connects. Roles cannot be explicitly enabled or disabled.
- The security administrator can delegate management of a role to others.

All DB2 privileges and authorities that can be granted within a database can be granted to a role, with the exception of security administrator (SECADM) authority. For example, a role can be granted any of the following authorities and privileges:

- DBADM, LOAD, and IMPLICIT_SCHEMA database authorities
- CONNECT, CREATETAB, CREATE_NOT_FENCED, BINDADD CREATE_EXTERNAL_ROUTINE, or QUIESCE_CONNECT database authorities
- Any database object privilege (including CONTROL)

A user's roles are automatically enabled and considered for authorization when a user connects to a database; you do not need to activate a role by using the SET ROLE statement. For example, when you create a view, a materialized query table (MQT), a trigger, a package, or an SQL routine, the privileges that you gain through roles apply. However, privileges that you gain through roles granted to groups of which you are a member do not apply.

A role does not have an owner. The security administrator can use the WITH ADMIN OPTION clause of the GRANT statement to delegate management of the role to another user, so that the other user can control the role membership.

Restrictions

There are a few restrictions in the use of roles:

- A role cannot own database objects.
- A role cannot be granted security administrator (SECADM) authority.
- Permissions and roles granted to groups are not considered when you create the following database objects:
  - Packages containing static SQL
  - Views
  - Materialized query tables (MQT)
  - Triggers
  - SQL Routines

  Only roles granted to the user creating the object or to PUBLIC, directly or indirectly (such as through a role hierarchy), are considered when creating these objects.

Creating and granting membership in roles

The security administrator holds the authority to create, drop, grant, revoke, and comment on a role. The security administrator uses the GRANT (Role) statement to grant membership in a role to an authorization ID and uses the REVOKE (Role) statement to revoke membership in a role from an authorization ID.

The security administrator can delegate the management of membership in a role to an authorization ID by granting the authorization ID membership in the role with the WITH ADMIN OPTION. The WITH ADMIN OPTION clause of the GRANT (Role) statement gives another user the ability to:

- Grant roles to others.
- Revoke roles from others.
- Comment on the role.

The WITH ADMIN OPTION clause does not give the ability to:

- Drop the role.
- Revoke the WITH ADMIN OPTION for a role from an authorization ID.
- Grant WITH ADMIN OPTION to someone else (if you do not hold SECADM authority).

After the security administrator has created a role, the database administrator can use the GRANT statement to assign authorities and privileges to the role. All DB2 privileges and authorities that can be granted within a database can be granted to

a role with the exception of SECADM authority. Instance level authorities, such as SYSADM authority, cannot be assigned to a role.

The security administrator, or any user who the security administrator has granted membership in a role with WITH ADMIN OPTION can use the GRANT (Role) statement to grant membership in that role to other users, groups, PUBLIC or roles. A user may have been granted membership in a role with WITH ADMIN OPTION either directly, or indirectly through PUBLIC, a group or a role.

All the roles assigned to a user are enabled when that user establishes a session. All the privileges and authorities associated with a user's roles are taken into account when the DB2 database system checks for authorization. Some database systems use the SET ROLE statement to activate a particular role. The DB2 database system supports SET ROLE to provide compatibility with other products using the SET ROLE statement. In a DB2 database system, the SET ROLE statement checks whether the session user is a member of the role and returns an error if they are not.

To revoke a user's membership in a role, the security administrator, or a user who holds WITH ADMIN OPTION privilege on the role, uses the REVOKE (Role) statement.

Example

A role has a certain set of privileges and a user who is granted membership in this role inherits those privileges. This inheritance of privileges eliminates managing individual privileges when reassigning the privileges of one user to another user. The only operations required when using roles is to revoke membership in the role from one user and grant membership in the role to the other user.

For example, the employees BOB and ALICE, working in department DEV, have the privilege to SELECT on the tables SERVER, CLIENT and TOOLS. One day, management decides to move them to a new department, QA, and the database administrator has to revoke their privilege to select on tables SERVER, CLIENT and TOOLS. Department DEV later hires a new employee, TOM, and the database administrator has to grant SELECT privilege on tables SERVER, CLIENT and TOOLS to TOM.

When using roles, the following steps occur:

1. The security administrator creates a role, DEVELOPER:
   ```
   CREATE ROLE DEVELOPER
   ```
2. The database administrator (who holds DBADM authority) grants SELECT on tables SERVER, CLIENT, and TOOLS to role DEVELOPER:
   ```
   GRANT  SELECT ON TABLE SERVER TO ROLE DEVELOPER
   GRANT  SELECT ON TABLE CLIENT TO ROLE DEVELOPER
   GRANT  SELECT ON TABLE TOOLS TO ROLE DEVELOPER
   ```
3. The security administrator grants the role DEVELOPER to the users in department DEV, BOB and ALICE:
   ```
   GRANT ROLE DEVELOPER TO USER BOB, USER ALICE
   ```
4. When BOB and ALICE leave department DEV, the security administrator revokes the role DEVELOPER from users BOB and ALICE:
   ```
   REVOKE ROLE DEVELOPER FROM USER BOB, USER ALICE
   ```
5. When TOM is hired in department DEV, the security administrator grants the role DEVELOPER to user TOM:

```
GRANT ROLE DEVELOPER TO USER TOM
```

Role hierarchies

A role hierarchy is formed when one role is granted membership in another role.

A role *contains* another role when the other role is granted to the first role. The other role inherits all of the privileges of the first role. For example, if the role DOCTOR is granted to the role SURGEON, then SURGEON is said to contain DOCTOR. The role SURGEON inherits all the privileges of role DOCTOR.

Cycles in role hierarchies are not allowed. A *cycle* occurs if a role is granted in circular way such that one role is granted to another role and that other role is granted to the original role. For example, the role DOCTOR is granted to role SURGEON, and then the role SURGEON is granted back to the role DOCTOR. If you create a cycle in a role hierarchy, an error is returned (SQLSTATE 428GF).

Example of building a role hierarchy

The following example shows how to build a role hierarchy to represent the medical levels in a hospital.

Consider the following roles: DOCTOR, SPECIALIST, and SURGEON. A role hierarchy is built by granting a role to another role, but without creating cycles. The role DOCTOR is granted to role SPECIALIST, and role SPECIALIST is granted to role SURGEON.

Granting role SURGEON to role DOCTOR would create a cycle and is not allowed.

The security administrator runs the following SQL statements to build the role hierarchy:

```
CREATE ROLE DOCTOR
CREATE ROLE SPECIALIST
CREATE ROLE SURGEON

GRANT ROLE DOCTOR TO ROLE SPECIALIST

GRANT ROLE SPECIALIST TO ROLE SURGEON
```

Effect of revoking privileges from roles

When privileges are revoked, this can sometimes cause dependent database objects, such as views, packages or triggers, to become invalid or inoperative.

The following examples show what happens to a database object when some privileges are revoked from an authorization identifier and privileges are held through a role or through different means.

Example of revoking privileges from roles

1. The security administrator creates the role DEVELOPER and grants the user BOB membership in this role:

   ```
   CREATE ROLE DEVELOPER
   GRANT ROLE DEVELOPER TO USER BOB
   ```

2. User ALICE creates a table, WORKITEM:

   ```
   CREATE TABLE WORKITEM (x int)
   ```

3. The database administrator grants SELECT and INSERT privileges on table WORKITEM to PUBLIC and also to the role DEVELOPER:

```
GRANT SELECT ON TABLE ALICE.WORKITEM TO PUBLIC
GRANT INSERT ON TABLE ALICE.WORKITEM TO PUBLIC
GRANT SELECT ON TABLE ALICE.WORKITEM TO ROLE DEVELOPER
GRANT INSERT ON TABLE ALICE.WORKITEM TO ROLE DEVELOPER
```

4. User BOB creates a view, PROJECT, that uses the table WORKITEM, and a package, PKG1, that depends on the table WORKITEM:

```
CREATE VIEW PROJECT AS SELECT * FROM ALICE.WORKITEM
PREP emb001.sqc BINDFILE PACKAGE USING PKG1 VERSION 1
```

5. If the database administrator revokes SELECT privilege on table ALICE.WORKITEM from PUBLIC, then the view BOB.PROJECT remains operative and package PKG1 remains valid because the view definer, BOB, still holds the privileges required through his membership in the role DEVELOPER:

```
REVOKE SELECT ON TABLE ALICE.WORKITEM FROM PUBLIC
```

6. If the database administrator revokes SELECT privilege on table ALICE.WORKITEM from the role DEVELOPER, the view BOB.PROJECT becomes inoperative and package PKG1 becomes invalid because the view and package definer, BOB, does not hold the required privileges through other means:

```
REVOKE SELECT ON TABLE ALICE.WORKITEM FROM ROLE DEVELOPER
```

Example of revoking DBADM authority

In this example, the role DEVELOPER holds DBADM authority and is granted to user BOB.

1. The security administrator creates the role DEVELOPER:

```
CREATE ROLE DEVELOPER
```

2. The system administrator grants DBADM authority to the role DEVELOPER:

```
GRANT DBADM ON DATABASE TO ROLE DEVELOPER
```

3. The security administrator grants user BOB membership in this role:

```
GRANT ROLE DEVELOPER TO USER BOB
```

4. User ALICE creates a table, WORKITEM:

```
CREATE TABLE WORKITEM (x int)
```

5. User BOB creates a view PROJECT that uses table WORKITEM, a package PKG1 that depends on table WORKITEM, and a trigger, TRG1, that also depends on table WORKITEM:

```
CREATE VIEW PROJECT AS SELECT * FROM ALICE.WORKITEM
PREP emb001.sqc BINDFILE PACKAGE USING PKG1 VERSION 1
CREATE TRIGGER TRG1 AFTER DELETE ON ALICE.WORKITEM
              FOR EACH STATEMENT MODE DB2SQL
       INSERT INTO ALICE.WORKITEM VALUES (1)
```

6. The security administrator revokes the role DEVELOPER from user BOB:

```
REVOKE ROLE DEVELOPER FROM USER BOB
```

   Revoking the role DEVELOPER causes the user BOB to lose DBADM authority because the role that held that authority was revoked. The view, package, and trigger are affected as follows:

   - View BOB. PROJECT is still valid.
   - Package PKG1 becomes invalid.
   - Trigger BOB.TRG1 is still valid.

View BOB.PROJECT and trigger BOB.TRG1 are usable while package PKG1 is not usable. View and trigger objects created by an authorization ID holding DBADM authority are not affected when DBADM authority is lost.

Delegating role maintenance by using the WITH ADMIN OPTION clause

Using the WITH ADMIN OPTION clause of the GRANT (Role) SQL statement, a security administrator can delegate the management and control of membership in a role to someone else. The WITH ADMIN OPTION clause gives another user the authority to grant membership in the role to other users, to revoke membership in the role from other members of the role, and to comment on a role, but not to drop the role.

The WITH ADMIN OPTION clause does not give another user the authority to grant WITH ADMIN OPTION on a role to another user. It also does not give the authority to revoke WITH ADMIN OPTION for a role from another authorization ID.

Example demonstrating use of the WITH ADMIN OPTION clause

1. A security administrator creates the role, DEVELOPER, and grants the new role to user BOB using the WITH ADMIN OPTION clause:

   ```
   CREATE ROLE DEVELOPER
   GRANT ROLE DEVELOPER TO USER BOB WITH ADMIN OPTION
   ```

2. User BOB can grant membership in the role to and revoke membership from the role from other users, for example, ALICE:

   ```
   GRANT ROLE DEVELOPER TO USER ALICE
   REVOKE ROLE DEVELOPER FROM USER ALICE
   ```

3. User BOB cannot drop the role or grant WITH ADMIN OPTION to another user (only a security administrator can perform these two operations). These commands issued by BOB will fail:

   ```
   DROP ROLE DEVELOPER  - FAILURE!
                        - only a security administrator is allowed to drop the role
   GRANT ROLE DEVELOPER TO USER ALICE WITH ADMIN OPTION - FAILURE!
                        - only a security administrator can grant WITH ADMIN OPTION
   ```

4. User BOB cannot revoke role administration privileges (conferred by WITH ADMIN OPTION) from users for role DEVELOPER, because he does not have security administrator (SECADM) authority. When BOB issues the following command, it fails:

   ```
   REVOKE ADMIN OPTION FOR ROLE DEVELOPER FROM USER SANJAY - FAILURE!
   ```

5. A security administrator is allowed to revoke the role administration privileges for role DEVELOPER (conferred by WITH ADMIN OPTION) from user BOB , and user BOB still has the role DEVELOPER granted:

   ```
   REVOKE ADMIN OPTION FOR ROLE DEVELOPER FROM USER BOB
   ```

   Alternatively, if a security administrator simply revokes the role DEVELOPER from user BOB, then BOB loses all the privileges he received by being a member of the role DEVELOPER and the authority on the role he received through the WITH ADMIN OPTION clause:

   ```
   REVOKE ROLE DEVELOPER FROM USER BOB
   ```

Roles compared to groups

Privileges and authorities granted to groups are not considered when creating views, materialized query tables (MQTs), SQL routines, triggers, and packages containing static SQL. Avoid this restriction by using roles instead of groups.

Roles allow users to create database objects using their privileges acquired through roles, which are controlled by the DB2 database system. Groups and users are controlled externally from the DB2 database system, for example, by an operating system or an LDAP server.

Example of replacing the use of groups with roles

This example shows how you can replace groups by using roles.

Assume there are three groups, DEVELOPER_G, TESTER_G and SALES_G. The users BOB, ALICE, and TOM are members of these groups, as shown in the following table:

Table 7. Example groups and users

Group	Users belonging to this group
DEVELOPER_G	BOB
TESTER_G	ALICE, TOM
SALES_G	ALICE, BOB

1. The security administrator creates the roles DEVELOPER, TESTER and SALES to be used instead of the groups.

   ```
   CREATE ROLE DEVELOPER
   CREATE ROLE TESTER
   CREATE ROLE SALES
   ```

2. The security administrator grants membership in these roles to users (setting the membership of users in groups was the system administrator's responsibility):

   ```
   GRANT  ROLE DEVELOPER TO USER BOB
   GRANT  ROLE TESTER TO USER ALICE, USER TOM
   GRANT  ROLE SALES TO USER BOB, USER ALICE
   ```

3. The database administrator can grant to the roles similar privileges or authorities as were held by the groups, for example:

   ```
   GRANT <privilege> ON <object> TO ROLE DEVELOPER
   ```

   The database administrator can then revoke these privileges from the groups, as well as ask the system administrator to remove the groups from the system.

Example of creating a trigger using privileges acquired through a role

This example shows that user BOB can successfully create a trigger, TRG1, when he holds the necessary privilege through the role DEVELOPER.

1. First, user ALICE creates the table, WORKITEM:

   ```
   CREATE TABLE WORKITEM (x int)
   ```

2. Then, the privilege to alter ALICE's table is granted to role DEVELOPER by the database administrator.

   ```
   GRANT ALTER ON ALICE.WORKITEM TO ROLE DEVELOPER
   ```

3. User BOB successfully creates the trigger, TRG1, because he is a member of the role, DEVELOPER.

   ```
   CREATE TRIGGER TRG1 AFTER DELETE ON ALICE.WORKITEM
              FOR EACH STATEMENT MODE DB2SQL INSERT INTO ALICE.WORKITEM VALUES (1)
   ```

Using roles after migrating from IBM Informix Dynamic Server

If you have migrated from IBM Informix® Dynamic Server to the DB2 database system and are using roles there are a few things you need to be aware of.

The Informix Dynamic Server (IDS) SQL statement, GRANT ROLE, provides the clause WITH GRANT OPTION. The DB2 database system GRANT ROLE statement provides the clause WITH ADMIN OPTION (this conforms to the SQL standard) that provides the same functionality. During an IDS to DB2 database system migration, after the dbschema tool generates CREATE ROLE and GRANT ROLE statements, the dbschema tool replaces any occurrences of WITH GRANT OPTION with WITH ADMIN OPTION.

In an IDS database system, the SET ROLE statement activates a particular role. The DB2 database system supports the SET ROLE statement, but only to provide compatibility with other products using that SQL statement. The SET ROLE statement checks whether the session user is a member of the role and returns an error if they are not.

Example dbschema output

Assume that an IDS database contains the roles DEVELOPER, TESTER and SALES. Users BOB, ALICE, and TOM have different roles granted to each of them; the role DEVELOPER is granted to BOB, the role TESTER granted to ALICE, and the roles TESTER and SALES granted to TOM. To migrate to the DB2 database system, use the dbschema tool to generate the CREATE ROLE and GRANT ROLE statements for the database:

```
CREATE ROLE DEVELOPER
CREATE ROLE TESTER
CREATE ROLE SALES

GRANT  DEVELOPER TO BOB
GRANT  TESTER TO ALICE, TOM
GRANT  SALES TO TOM
```

You must create the database in the DB2 database system, and then you can run the above statements in that database to recreate the roles and assignment of the roles.

Chapter 3. Using trusted contexts and trusted connections

You can establish an explicit trusted connection by making an explicit request within an application when a connection to DB2 is established. In order to be successful, the security administrator must have defined a trusted context, using the CREATE TRUSTED CONTEXT SQL statement, with attributes matching those of the connection you are establishing (see Step 1, later).

The API you use to request an explicit trusted connection when you establish a connection depends on the type of application you are using (see the table in Step 2). After you have established an explicit trusted connection, the application can switch the user for the connection, subject to the constraints imposed on such switching by the trusted context definition, using the appropriate API for the type of application (see the table in Step 3).

1. The security administrator defines a trusted context in the server by using the CREATE TRUSTED CONTEXT statement. For example:

```
CREATE TRUSTED CONTEXT MYTCX
       BASED UPON CONNECTION USING SYSTEM AUTHID NEWTON
       ATTRIBUTES (ADDRESS   '192.0.2.1')
       WITH USE FOR PUBLIC WITHOUT AUTHENTICATION
       ENABLE
```

2. To establish a trusted connection, use one of the following APIs in your application:

Option	Description
Application	**API**
CLI/ODBC	SQLConnect, SQLSetConnectAttr
XA CLI/ODBC	Xa_open
JAVA	getDB2TrustedPooledConnection, getDB2TrustedXAConnection

3. To switch to a different user, with or without authentication, use one of the following APIs in your application:

Option	Description
Application	**API**
CLI/ODBC	SQLSetConnectAttr
XA CLI/ODBC	SQLSetConnectAttr
JAVA	getDB2Connection, reuseDB2Connection

Example of establishing an explicit trusted connection and switching the user

In the following example, a middle-tier server needs to issue some database requests on behalf of an end-user, but does not have access to the end-user's credentials to establish a database connection on behalf of that end-user.

You can create a trusted context object on the database server that allows the middle-tier server to establish an explicit trusted connection to the database. After establishing an explicit trusted connection, the middle-tier server can switch the

current user ID of the connection to a new user ID without the need to authenticate the new user ID at the database server. The following CLI code snippet demonstrates how to establish a trusted connection using the trusted context, MYTCX, defined in Step 1, earlier, and how to switch the user on the trusted connection without authentication.

```
int main(int argc, char *argv[])
{
 SQLHANDLE henv;         /* environment handle */
 SQLHANDLE hdbc1;       /* connection handle */
 char origUserid[10] = "newton";
  char password[10] = "test";
   char switchUserid[10] = "zurbie";
 char dbName[10] = "testdb";

// Allocate the handles
SQLAllocHandle( SQL_HANDLE_ENV, &henv );
SQLAllocHandle( SQL_HANDLE_DBC, &hdbc1 );

// Set the trusted connection attribute
SQLSetConnectAttr( hdbc1, SQL_ATTR_USE_TRUSTED_CONTEXT,
SQL_TRUE, SQL_IS_INTEGER );

// Establish a trusted connection
SQLConnect( hdbc1, dbName, SQL_NTS, origUserid, SQL_NTS,
password, SQL_NTS );

//Perform some work under user ID "newton"
. . . . . .

 // Commit the work
SQLEndTran(SQL_HANDLE_DBC, hdbc1, SQL_COMMIT);


// Switch the user ID on the trusted connection
SQLSetConnectAttr( hdbc1,
SQL_ATTR_TRUSTED_CONTEXT_USERID, switchUserid,
SQL_IS_POINTER
);

//Perform new work using user ID "zurbie"
. . . . . .

//Commit the work
SQLEndTranSQL_HANDLE_DBC, hdbc1, SQL_COMMIT);

// Disconnect from database
SQLDisconnect( hdbc1 );

 return 0;

}  /* end of main */
```

Problem determination:

An explicit trusted connection is a connection that is successfully established by a specific, explicit request for a trusted connection. When you request an explicit trusted connection and you do not qualify for one, you get a regular connection and a warning (+20360). To determine why a user could not establish a trusted connection, the security administrator needs to look at the trusted context definition in the system catalogs and at the connection attributes. In particular, the IP address from which the connection is established, the encryption level of the

data stream or network, and the system authorization ID making the connection. The -application option of the db2pd utility returns this information, as well as the following information:

- Connection Trust Type: Indicates whether the connection is trusted or not. When the connection is trusted, this also indicates whether this is an explicit trusted connection or an implicit trusted connection.
- Trusted Context name: The name of the trusted context associated with the trusted connection.
- Role Inherited: The role inherited through the trusted connection.

Note:

- TCP/IP is the only supported protocol for a client application to communicate with the DB2 server that can be used to establish a trusted connection.
- When the database server authentication type is set to CLIENT, an explicit trusted connection cannot be established.
- Once established, a trusted connection retains its trusted nature for the duration of the connection.

Trusted contexts and trusted connections

A trusted context is a database object that defines a trust relationship for a connection between the database and an external entity such as an application server.

The trust relationship is based upon the following set of attributes:

- System authorization ID: Represents the user that establishes a database connection
- IP address (or domain name): Represents the host from which a database connection is established
- Data stream encryption: Represents the encryption setting (if any) for the data communication between the database server and the database client

When a user establishes a database connection, the DB2 database system checks whether the connection matches the definition of a trusted context object in the database. When a match occurs, the database connection is said to be trusted.

A trusted connection allows the initiator of this trusted connection to acquire additional capabilities that may not be available outside the scope of the trusted connection. The additional capabilities vary depending on whether the trusted connection is explicit or implicit.

The initiator of an explicit trusted connection has the ability to:

- Switch the current user ID on the connection to a different user ID with or without authentication
- Acquire additional privileges via the role inheritance feature of trusted contexts

An implicit trusted connection is a trusted connection that is not explicitly requested; the implicit trusted connection results from a normal connection request rather than an explicit trusted connection request. No application code changes are needed to obtain an implicit connection. Also, whether you obtain an implicit trusted connection or not has no effect on the connect return code (when you request an explicit trusted connection, the connect return code indicates whether

the request succeeds or not). The initiator of an implicit trusted connection can only acquire additional privileges via the role inheritance feature of trusted contexts; they cannot switch the user ID.

How using trusted contexts enhances security

The three-tiered application model extends the standard two-tiered client and server model by placing a middle tier between the client application and the database server. It has gained great popularity in recent years particularly with the emergence of web-based technologies and the Java™ 2 Enterprise Edition (J2EE) platform. An example of a software product that supports the three-tier application model is IBM WebSphere Application Server (WAS).

In a three-tiered application model, the middle tier is responsible for authenticating the users running the client applications and for managing the interactions with the database server. Traditionally, all the interactions with the database server occur through a database connection established by the middle tier using a combination of a user ID and a credential that identify that middle tier to the database server. In other words, the database server uses the database privileges associated with the middle tier's user ID for all authorization checking and auditing that must occur for any database access, including access performed by the middle tier on behalf of a user.

While the three-tiered application model has many benefits, having all interactions with the database server (for example, a user request) occur under the middle tier's authorization ID raises several security concerns, which can be summarized as follows:

- Loss of user identity

  Some enterprises prefer to know the identity of the actual user accessing the database for access control purposes.
- Diminished user accountability

  Accountability through auditing is a basic principle in database security. Not knowing the user's identity makes it difficult to distinguish the transactions performed by the middle tier for its own purpose from those performed by the middle tier on behalf of a user.
- Over granting of privileges to the middle tier's authorization ID

  The middle tier's authorization ID must have all the privileges necessary to execute all the requests from all the users. This has the security issue of enabling users who do not need access to certain information to obtain access anyway.
- Weakened security

  In addition to the privilege issue raised in the previous point, the current approach requires that the authorization ID used by the middle tier to connect must be granted privileges on all resources that might be accessed by user requests. If that middle-tier authorization ID is ever compromised, then all those resources will be exposed.
- "Spill over" between users of the same connection

  Changes by a previous user can affect the current user.

Clearly, there is a need for a mechanism whereby the actual user's identity and database privileges are used for database requests performed by the middle tier on behalf of that user. The most straightforward approach of achieving this goal would be for the middle-tier to establish a new connection using the user's ID and password, and then direct the user's requests through that connection. Although simple, this approach suffers from several drawbacks which include the following:

- Inapplicability for certain middle tiers. Many middle-tier servers do not have the user authentication credentials needed to establish a connection.
- Performance overhead. There is an obvious performance overhead associated with creating a new physical connection and re-authenticating the user at the database server.
- Maintenance overhead. In situations where you are not using a centralized security set up or are not using single sign-on, there is maintenance overhead in having two user definitions (one on the middle tier and one at the server). This requires changing passwords at different places.

The trusted contexts capability addresses this problem. The security administrator can create a trusted context object in the database that defines a trust relationship between the database and the middle-tier. The middle-tier can then establish an explicit trusted connection to the database, which gives the middle tier the ability to switch the current user ID on the connection to a different user ID, with or without authentication. In addition to solving the end-user identity assertion problem, trusted contexts offer another advantage. This is the ability to control when a privilege is made available to a database user. The lack of control on when privileges are available to a user can weaken overall security. For example, privileges may be used for purposes other than they were originally intended. The security administrator can assign one or more privileges to a role and assign that role to a trusted context object. Only trusted database connections (explicit or implicit) that match the definition of that trusted context can take advantage of the privileges associated with that role.

Enhancing performance

When you use trusted connections, you can maximize performance because of the following advantages:

- No new connection is established when the current user ID of the connection is switched.
- If the trusted context definition does not require authentication of the user ID to switch to, then the overhead associated with authenticating a new user at the database server is not incurred.

Example of creating a trusted context

Suppose that the security administrator creates the following trusted context object:

```
CREATE TRUSTED CONTEXT CTX1
  BASED UPON CONNECTION USING SYSTEM AUTHID USER2
  ATTRIBUTES (ADDRESS   '192.0.2.1')
  DEFAULT ROLE managerRole
   ENABLE
```

If user *user1* requests a trusted connection from IP address 192.0.2.1, the DB2 database system returns a warning (SQLSTATE 01679, SQLCODE +20360) to indicate that a trusted connection could not be established, and that user *user1* simply got a non-trusted connection. However, if user *user2* requests a trusted connection from IP address 192.0.2.1, the request is honored because the connection attributes are satisfied by the trusted context CTX1. Now that use *user2* has established a trusted connection, he or she can now acquire all the privileges and authorities associated with the trusted context role managerRole. These privileges and authorities may not be available to user *user2* outside the scope of this trusted connection

Role membership inheritance through a trusted context

The current user of a trusted connection can acquire additional privileges through the automatic inheritance of a role through the trusted context, if this was specified by the security administrator as part of the relevant trusted context definition.

A role can be inherited by all users of the trusted connection by default. The security administrator can also use the trusted context definition to specify a role for specific users to inherit.

The active roles that a session authorization ID can hold while on a trusted connection are:

- The roles of which the session authorization ID is normally considered a member, plus
- Either the trusted context default role or the trusted context user-specific role, if they are defined

Note:

- If you configure user authentication using a custom security plugin that is built such that the system authorization ID and the session authorization ID produced by this security plugin upon a successful connection are different from each other, then a trusted contexts role cannot be inherited through that connection, even if it is a trusted connection.
- Trusted context privileges acquired through a role are effective only for dynamic DML operations. They are not effective for:
  - DDL operations
  - Non-dynamic SQL (operations involving static SQL statements such as BIND, REBIND, implicit rebind, incremental bind, and so on)

Acquiring trusted context user-specific privileges

The security administrator can use the trusted context definition to associate roles with a trusted context so that:

- All users of the trusted connection can inherit a specified role by default
- Specific users of the trusted connection can inherit a specified role

When the user on a trusted connection is switched to a new authorization ID and a trusted context user-specific role exists for this new authorization ID, the user-specific role overrides the trusted context default role, if one exists, as demonstrated in the example.

Example of creating a trusted context that assigns a default role and a user-specific role

Suppose that the security administrator creates the following trusted context object:

```
CREATE TRUSTED CONTEXT CTX1
  BASED UPON CONNECTION USING SYSTEM AUTHID USER1
 ATTRIBUTES (ADDRESS   '192.0.2.1')
  WITH USE FOR USER2 WITH AUTHENTICATION,
               USER3 WITHOUT AUTHENTICATION
  DEFAULT ROLE AUDITOR
  ENABLE
```

When USER1 establishes a trusted connection, the privileges granted to the role AUDITOR are inherited by this authorization ID. Similarly, these same privileges

are also inherited by USER3 when the current authorization ID on the trusted connection is switched to his or her user ID. (If the user ID of the connection is switched to USER2 at some point, then USER2 would also inherit the trusted context default role, AUDITOR.) The security administrator may choose to have USER3 inherit a different role than the trusted context default role. They can do so by assigning a specific role to this user as follows:

```
CREATE TRUSTED CONTEXT CTX1
 BASED UPON CONNECTION USING SYSTEM AUTHID USER1
 ATTRIBUTES (ADDRESS   '192.0.2.1')
  WITH USE FOR USER2 WITH AUTHENTICATION,
               USER3 WITHOUT AUTHENTICATION ROLE OTHER_ROLE
  DEFAULT ROLE AUDITOR
  ENABLE
```

When the current user ID on the trusted connection is switched to USER3, this user no longer inherits the trusted context default role. Rather, they inherit the specific role, OTHER_ROLE, assigned to him or her by the security administrator.

User ID switching on an explicit trusted connection

On an explicit trusted connection, you can switch the user ID of the connection to a different user ID, either with or without authenticating depending on the definition of the trusted context object associated with the explicit trusted connection.

To switch user on a trusted connection, the request must be made explicitly, using one of these APIs within the application:

Table 8. Application APIs for switch user on a trusted connection request

Application	API
CLI/ODBC	SQLSetConnectAttr
JAVA	getDB2Connection, reuseDB2Connection
XA	SQLSetConnectAttr

Rules for switching the user ID

1. If the switch request is not made from an explicit trusted connection, and the switch request is sent to the server for processing, the connection is shut down and an error message is returned (SQLSTATE 08001, SQLCODE -30082 with reason code 41).
2. If the switch request is not made on a transaction boundary, the transaction is rolled back, and the switch request is sent to the server for processing, the connection is put into an unconnected state and an error message is returned (SQLSTATE 58009, SQLCODE -30020).
3. If the switch request is made from within a stored procedure, an error message is returned (SQLCODE -30090, reason code 29), indicating this is an illegal operation in this environment. The connection state is maintained and the connection is not placed into an unconnected state. Subsequent requests may be processed.
4. If the switch request is delivered to the server on an instance attach (rather than a database connection), the attachment is shut down and an error message is returned (SQLCODE -30005).
5. If the switch request is made with an authorization ID that is not allowed on the trusted connection, error (SQLSTATE 42517, SQLCODE -20361) is returned, and the connection is put in an unconnected state.

6. If the switch request is made with an authorization ID that is allowed on the trusted connection WITH AUTHENTICATION, but the appropriate authentication token is not provided, error (SQLSTATE 42517, SQLCODE -20361) is returned, and the connection is put in an unconnected state.
7. If the trusted context object associated with the trusted connection is disabled, and a switch request for that trusted connection is made, error (SQLSTATE 42517, SQLCODE -20361) is returned, and the connection is put in an unconnected state.

   In this case, the only switch user request that is accepted is one that specifies the user ID that established the trusted connection or the NULL user ID. If a switch to the user ID that established the trusted connection is made, this user ID does not inherit any trusted context role (neither the trusted context default role nor the trusted context user-specific role).
8. If the system authorization ID attribute of the trusted context object associated with the trusted connection is changed, and a switch request for that trusted connection is made, error (SQLSTATE 42517, SQLCODE -20361) is returned, and the connection is put in an unconnected state.

   In this case, the only switch user request that is accepted is one that specifies the user ID that established the trusted connection or the NULL user ID. If a switch to the user ID that established the trusted connection is made, this user ID does not inherit any trusted context role (neither the trusted context default role nor the trusted context user-specific role).
9. If the trusted context object associated with the trusted connection is dropped, and a switch request for that trusted connection is made, error (SQLSTATE 42517, SQLCODE -20361) is returned, and the connection is put in an unconnected state.

   In this case, the only switch user request that is accepted is one that specifies the user ID that established the trusted connection or the NULL user ID. If a switch to the user ID that established the trusted connection is made, this user ID does not inherit any trusted context role (neither the trusted context default role nor the trusted context user-specific role).
10. If the switch request is made with a user ID allowed on the trusted connection, but that user ID does not hold CONNECT privilege on the database, the connection is put in an unconnected state and an error message is returned (SQLSTATE 08004, SQLCODE -1060).
11. If the trusted context system authorization ID appears in the WITH USE FOR clause, the DB2 database system honors the authentication setting for the system authorization ID on switch user request to switch back to the system authorization ID. If the trusted context system authorization ID does not appear in the WITH USE FOR clause, then a switch user request to switch back to the system authorization ID is always allowed even without authentication.

Note: When the connection is put in the unconnected state, the only requests that are accepted and do not result in returning the error ″The application state is in error. A database connection does not exist.″ (SQLCODE -900) are:

- A switch user request
- A COMMIT or ROLLBACK statement
- A DISCONNECT, CONNECT RESET or CONNECT request

Note: When the user ID on the trusted connection is switched to a new user ID, all traces of the connection environment under the old user are gone. In other words, the switching of user IDs results in an environment that is identical to a

new connection environment. For example, if the old user ID on the connection had any temporary tables or WITH HOLD cursors open, these objects are completely lost when the user ID on that connection is switched to a new user ID.

Example of creating a trusted connection that specifies authentication for certain users

Suppose that the security administrator creates the following trusted context object:

```
CREATE TRUSTED CONTEXT CTX1
 BASED UPON CONNECTION USING SYSTEM AUTHID USER1
 ATTRIBUTES (ADDRESS   '192.0.2.1')
  WITH USE FOR USER2 WITH AUTHENTICATION,
               USER3 WITHOUT AUTHENTICATION
  ENABLE
```

Further, suppose that an explicit trusted connection is established. A request to switch the user ID on the trusted connection to USER3 without providing authentication information is allowed because USER3 is defined as a user of trusted context CTX1 for whom authentication is not required. However, a request to switch the user ID on the trusted connection to USER2 without providing authentication information will fail because USER2 is defined as a user of trusted context CTX1 for whom authentication information must be provided.

Chapter 4. Label-based access control (LBAC)

Label-based access control (LBAC) greatly increases the control you have over who can access your data. LBAC lets you decide exactly who has write access and who has read access to individual rows and individual columns.

What LBAC does

The LBAC capability is very configurable and can be tailored to match your particular security environment. All LBAC configuration is performed by a *security administrator*, which is a user that has been granted the SECADM authority by the system administrator.

A security administrator configures the LBAC system by creating security policies. A *security policy* describes the criteria that will be used to decide who has access to what data. Only one security policy can be used to protect any one table but different tables can be protected by different security policies.

After creating a security policy, a security administrator creates objects, called *security labels* that are part of that policy. Exactly what makes up a security label is determined by the security policy and can be configured to represent the criteria that your organization uses to decide who should have access to particular data items. If you decide, for instance, that you want to look at a person's position in the company and what projects they are part of to decide what data they should see, then you can configure your security labels so that each label can include that information. LBAC is flexible enough to let you set up anything from very complicated criteria, to a very simple system where each label represents either a ″high″ or a ″low″ level of trust.

Once created, a security label can be associated with individual columns and rows in a table to protect the data held there. Data that is protected by a security label is called *protected data*. A security administrator allows users access to protected data by granting them security labels. When a user tries to access protected data, that user's security label is compared to the security label protecting the data. The protecting label will block some security labels and not block others.

A user, a role, or a group is allowed to hold security labels for multiple security policies at once. For any given security policy, however, a use, a role, or a group can hold at most one label for read access and one label for write access.

A security administrator can also grant exemptions to users. An *exemption* allows you to access protected data that your security labels might otherwise prevent you from accessing. Together your security labels and exemptions are called your *LBAC credentials*.

If you try to access a protected column that your LBAC credentials do not allow you to access then the access will fail and you will get an error message.

If you try to read protected rows that your LBAC credentials do not allow you to read then DB2 acts as if those rows do not exist. Those rows cannot be selected as part of any SQL statement that you run, including SELECT, UPDATE, or DELETE. Even the aggregate functions ignore rows that your LBAC credentials do not allow

you to read. The COUNT(*) function, for example, will return a count only of the rows that you have read access to.

Views and LBAC

You can define a view on a protected table the same way you can define one on a non-protected table. When such a view is accessed the LBAC protection on the underlying table is enforced. The LBAC credentials used are those of the session authorization ID. Two users accessing the same view might see different rows depending on their LBAC credentials.

Referential integrity constraints and LBAC

The following rules explain how LBAC rules are enforced in the presence of referential integrity constraints:

- **Rule 1**: The LBAC read access rules are NOT applied for internally generated scans of child tables. This is to avoid having orphan children.
- **Rule 2**: The LBAC read access rules are NOT applied for internally generated scans of parent tables
- **Rule 3**: The LBAC write rules are applied when a CASCADE operation is performed on child tables. For example, If a user deletes a parent, but cannot delete any of the children because of an LBAC write rule violation, then the delete should be rolled-back and an error raised.

Storage overhead when using LBAC

When you use LBAC to protect a table at the row level, the additional storage cost is the cost of the row security label column. This cost depends on the type of security label chosen. For example, if you create a security policy with two components to protect a table, a security label from that security policy will occupy 16 bytes (8 bytes for each component). Because the row security label column is treated as a not nullable VARCHAR column, the total cost in this case would be 20 bytes per row. In general, the total cost per row is (*N**8 + 4) bytes where *N* is the number of components in the security policy protecting the table.

When you use LBAC to protect a table at the column level, the column security label is meta-data (that is, it is stored together with the column's meta-data in the SYSCOLUMNS catalog table). This meta-data is simply the ID of the security label protecting the column. The user table does not incur any storage overhead in this case.

What LBAC does not do

- LBAC will never allow access to data that is forbidden by discretionary access control.

  Example: If you do not have permission to read from a table then you will not be allowed to read data from that table--even the rows and columns to which LBAC would otherwise allow you access.
- Your LBAC credentials only limit your access to protected data. They have no effect on your access to unprotected data.
- LBAC credentials are not checked when you drop a table or a database, even if the table or database contains protected data.
- LBAC credentials are not checked when you back up your data. If you can run a backup on a table, which rows are backed up is not limited in any way by the

LBAC protection on the data. Also, data on the backup media is not protected by LBAC. Only data in the database is protected.

- LBAC cannot be used to protect any of the following types of tables:
  - A materialized query table (MQT)
  - A table that a materialized query table (MQT) depends on
  - A staging table
  - A table that a staging table depends on
  - A typed table
- LBAC protection cannot be applied to a nickname.

LBAC tutorial

A tutorial leading you through the basics of using LBAC is available online. The tutorial is part of the IBM developerWorks® web site (http://www.ibm.com/developerworks/db2) and is called DB2 Label-Based Access Control, a practical guide.

LBAC security policies

The security administrator uses a security policy to define criteria that determine who has write access and who has read access to individual rows and individual columns of tables.

A security policy includes this information:

- What security label components are used in the security labels that are part of the policy
- What rules are used when comparing those security label components
- Which of certain optional behaviors are used when accessing data protected by the policy
- What additional security labels and exemptions are to be considered when enforcing access to data protected by the security policy. For example, the option to consider or not to consider security labels granted to roles and groups is controlled through the security policy.

Every protected table must have one and only one security policy associated with it. Rows and columns in that table can only be protected with security labels that are part of that security policy and all access of protected data follows the rules of that policy. You can have multiple security policies in a single database but you cannot have more than one security policy protecting any given table.

Creating a security policy

You must be a security administrator to create a security policy. You create a security policy with the SQL statement CREATE SECURITY POLICY. The security label components listed in a security policy must be created before the CREATE SECURITY POLICY statement is executed. The order in which the components are listed when a security policy is created does not indicate any sort of precedence or other relationship among the components but it is important to know the order when creating security labels with built-in functions like SECLABEL.

Altering a security policy

A security administrator can use the ALTER SECURITY POLICY statement to modify a security policy.

Dropping a security policy

You must be a security administrator to drop a security policy. You drop a security policy using the SQL statement DROP.

You cannot drop a security policy if it is associated with (added to) any table.

LBAC security label components overview

A *security label component* is a database object that is part of label-based access control (LBAC). You use security label components to model your organization's security structure.

A security label component can represent any criteria that you might use to decide if a user should have access to a given piece of data. Typical examples of such criteria include:

- How well trusted the user is
- What department the user is in
- Whether the user is involved in a particular project

Example: If you want the department that a user is in to affect which data they can access, you could create a component named dept and define elements for that component that name the various departments in your company. You would then include the component dept in your security policy.

An *element* of a security label component is one particular "setting" that is allowed for that component.

Example: A security label component that represents a level of trust might have the four elements: Top Secret, Secret, Classified, and Unclassified.

Creating a security label component

You must be a security administrator to create a security label component. You create security label components with the SQL statement CREATE SECURITY LABEL COMPONENT.

When you create a security label component you must provide:

- A name for the component
- What type of component it is (ARRAY, TREE, or SET)
- A complete list of allowed elements
- For types ARRAY and TREE you must describe how each element fits into the structure of the component

Types of components

There are three types of security label components:

- TREE: Each element represents a node in a tree structure
- ARRAY: Each element represents a point on a linear scale

- SET: Each element represents one member of a set

The types are used to model the different ways in which elements can relate to each other. For example, if you are creating a component to describe one or more departments in a company you would probably want to use a component type of TREE because most business structures are in the form of a tree. If you are creating a component to represent the level of trust that a person has, you would probably use a component of type ARRAY because for any two levels of trust, one will always be higher than the other.

The details of each type, including detailed descriptions of the relationships that the elements can have with each other, are described in their own section.

Altering security label components

The security administrator can use the ALTER SECURITY LABEL COMPONENT statement to modify a security label component.

Dropping a security label component

You must be a security administrator to drop a security label component. You drop a security label component with the SQL statement DROP.

LBAC security label component type: SET

SET is one type of security label component that can be used in a label-based access control (LBAC) security policy.

Components of type SET are unordered lists of elements. The only comparison that can be made for elements of this type of component is whether or not a given element is in the list.

LBAC security label component type: ARRAY

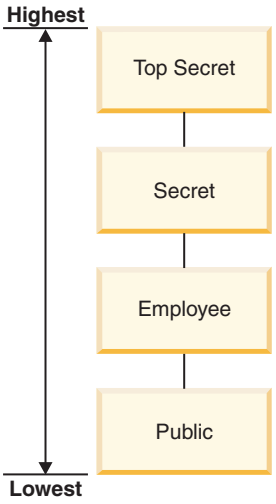
ARRAY is one type of security label component.

In the ARRAY type of component the order in which the elements are listed when the component is created defines a scale with the first element listed being the highest value and the last being the lowest.

Example: If the component mycomp is defined in this way:

```
CREATE SECURITY LABEL COMPONENT mycomp
  ARRAY [ 'Top Secret', 'Secret', 'Employee', 'Public' ]
```

Then the elements are treated as if they are organized in a structure like this:



In a component of type ARRAY, the elements can have these sorts of relationships to each other:

Higher than
Element A is higher than element B if element A is listed earlier in the ARRAY clause than element B.

Lower than
Element A is lower than element B if element A is listed later in the ARRAY clause than element B

LBAC security label component type: TREE

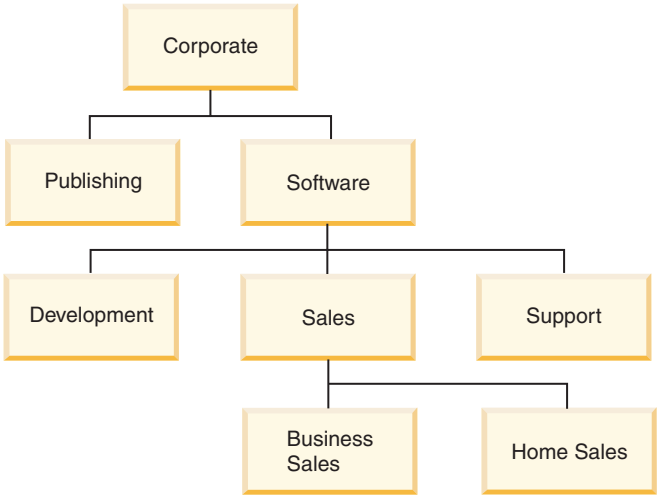
TREE is one type of security label component that can be used in a label-based access control (LBAC) security policy.

In the TREE type of component the elements are treated as if they are arranged in a tree structure. When you specify an element that is part of a component of type TREE you must also specify which other element it is under. The one exception is the first element which must be specified as being the ROOT of the tree. This allows you to organize the elements in a tree structure.

Example: If the component mycomp is defined this way:

```
CREATE SECURITY LABEL COMPONENT mycomp
TREE (
    'Corporate'      ROOT,
    'Publishing'     UNDER 'Corporate',
    'Software'       UNDER 'Corporate',
    'Development'    UNDER 'Software',
    'Sales'          UNDER 'Software',
    'Support'        UNDER 'Software'
    'Business Sales' UNDER 'Sales'
    'Home Sales'     UNDER 'Sales'
)
```

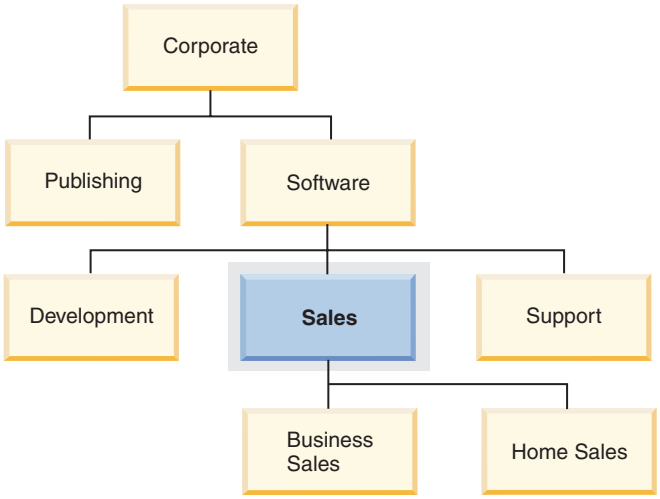
Then the elements are treated as if they are organized in a tree structure like this:



In a component of type TREE, the elements can have these types of relationships to each other:

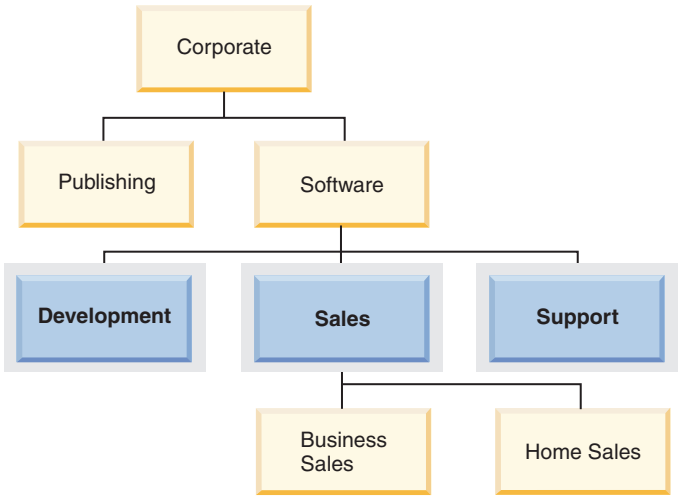
Parent Element A is a parent of element B if element B is UNDER element A.

Example: This diagram shows the parent of the Business Sales element:



Child Element A is a child of element B if element A is UNDER element B.

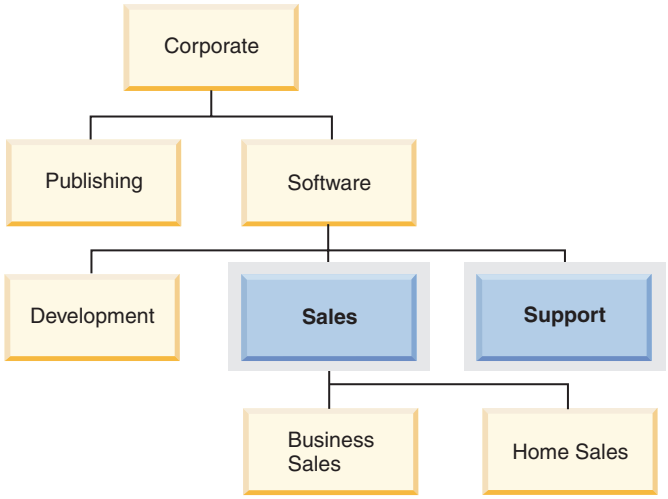
Example: This diagram shows the children of the Software element:



Sibling

Two elements are siblings of each other if they have the same parent.

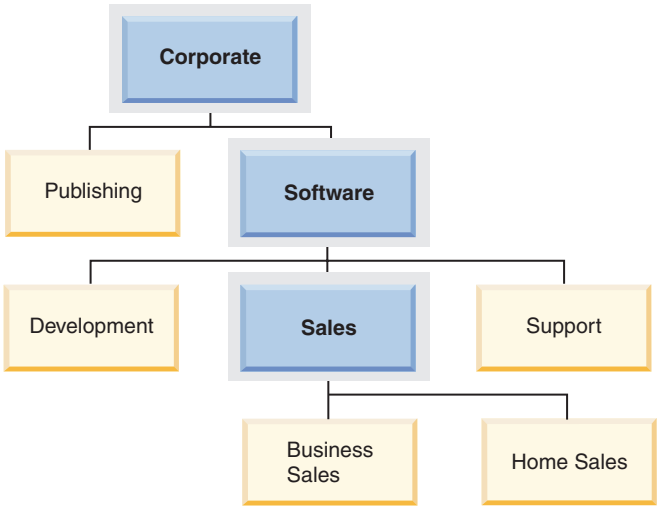
Example: This diagram shows the siblings of the Development element:



Ancestor

Element A is an ancestor of element B if it is the parent of B, or if it is the parent of the parent of B, and so on. The root element is an ancestor of all other elements in the tree.

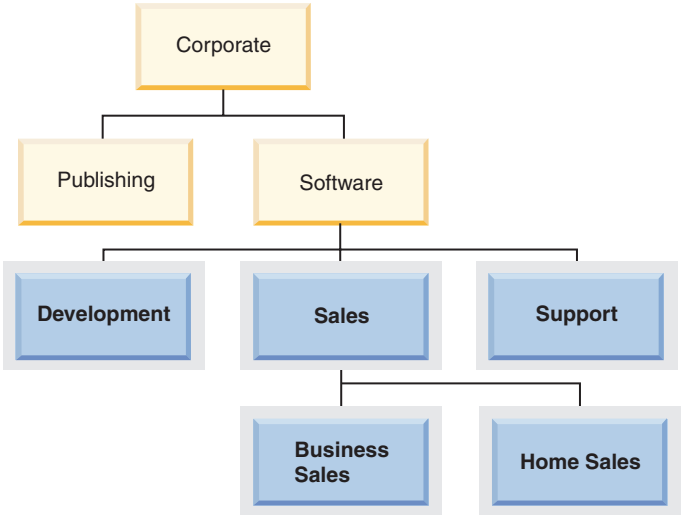
Example: This diagram shows the ancestors of the Home Sales element:



Descendent

Element A is a descendent of element B if it is the child of B, or if it is the child of a child of B, and so on.

Example: This diagram shows the descendents of the Software element:



LBAC security labels

In label-based access control (LBAC) a *security label* is a database object that describes a certain set of security criteria. Security labels are applied to data in order to protect the data. They are granted to users to allow them to access protected data.

When a user tries to access protected data, their security label is compared to the security label that is protecting the data. The protecting security label will block some security labels and not block others. If a user's security label is blocked then the user cannot access the data.

Every security label is part of exactly one security policy and includes one value for each component in that security policy. A *value* in the context of a security label component is a list of zero or more of the elements allowed by that component.

Values for ARRAY type components can contain zero or one element, values for other types can have zero or more elements. A value that does not include any elements is called an *empty value*.

Example: If a TREE type component has the three elements Human Resources, Sales, and Shipping then these are some of the valid values for that component:

- Human Resources (or any of the elements by itself)
- Human Resources, Shipping (or any other combination of the elements as long as no element is included more than once)
- *An empty value*

Whether a particular security label will block another is determined by the values of each component in the labels and the LBAC rule set that is specified in the security policy of the table. The details of how the comparison is made are given in the topic that discusses how LBAC security labels are compared.

When security labels are converted to a text string they use the format described in the topic that discusses the format for security label values.

Creating security labels

You must be a security administrator to create a security label. You create a security label with the SQL statement CREATE SECURITY LABEL. When you create a security label you provide:

- A name for the label
- The security policy that the label is part of
- Values for one or more of the components included in the security policy

Any components for which a value is not specified is assumed to have an empty value. A security label must have at least one non-empty value.

Altering security labels

Security labels cannot be altered. The only way to change a security label is to drop it and re-create it. However, the *components* of a security label can be modified by a security administrator (using the ALTER SECURITY LABEL COMPONENT statement).

Dropping security labels

You must be a security administrator to drop a security label. You drop a security label with the SQL statement DROP. You cannot drop a security label that is being used to protect data anywhere in the database or that is currently held by one or more users.

Granting security labels

You must be a security administrator to grant a security label to a user, a group, or a role. You grant a security label with the SQL statement GRANT SECURITY LABEL. When you grant a security label you can grant it for read access, for write access, or for both read and write access. A user, a group, or a role cannot hold more than one security label from the same security policy for the same type of access.

Revoking security labels

You must be a security administrator to revoke a security label from a user, group, or role. To revoke a security label, use the SQL statement REVOKE SECURITY LABEL.

Data types compatible with security labels

Security labels have a data type of SYSPROC.DB2SECURITYLABEL. Data conversion is supported between SYSPROC.DB2SECURITYLABEL and VARCHAR(128) FOR BIT DATA.

Format for security label values

Sometimes the values in a security label are represented in the form of a character string, for example when using the built-in function SECLABEL. When representing the values in a security label as a string this format is used.

- The values of the components are listed from left to right in the same order that the components are listed in the CREATE SECURITY POLICY statement for the security policy
- An element is represented by the name of that element
- Elements for different components are separated by a colon (:)
- If more than one element are given for the same component the elements are enclosed in parentheses (()) and are separated by a comma (,)
- Empty values are represented by a set of empty parentheses (())

Example: A security label is part of a security policy that has these three components in this order: Level, Department, and Projects. The security label has these values:

Table 9.

Component	Values
Level	Secret
Department	*Empty value*
Projects	• Epsilon 37 • Megaphone • Cloverleaf

This security label values look like this as a string:

```
'Secret:():(Epsilon 37,Megaphone,Cloverleaf)'
```

How LBAC security labels are compared

When you try to access data protected by label-based access control (LBAC), your LBAC credentials are compared to one or more security labels to see if the access is blocked. Your LBAC credentials are any security labels you hold plus any exemptions that you hold.

There are only two types of comparison that can be made. Your LBAC credentials can be compared to a single security label for read access or your LBAC credentials compared to a single security label for write access. Updating and deleting are

treated as being a read followed by a write. When an operation requires multiple comparisons to be made, each is made separately.

Which of your security labels is used

Even though you might hold multiple security labels only one is compared to the protecting security label. The label used is the one that meets these criteria:

- It is part of the security policy that is protecting the table being accessed.
- It was granted for the type of access (read or write).

If you do not have a security label that meets these criteria then a default security label is assumed that has empty values for all components.

How the comparison is made

Security labels are compared component by component. If a security label does not have a value for one of the components then an empty value is assumed. As each component is examined, the appropriate rules of the LBAC rule set are used to decide if the elements in your value for that component should be blocked by the elements in the value for the same component in the protecting label. If any of your values are blocked then your LBAC credentials are blocked by the protecting security label.

The LBAC rule set used in the comparison is designated in the security policy. To find out what the rules are and when each one is used, see the description of that rule set.

How exemptions affect comparisons

If you hold an exemption for the rule that is being used to compare two values then that comparison is not done and the protecting value is assumed not to block the value in your security label.

Example: The LBAC rule set is DB2LBACRULES and the security policy has two components. One component is of type ARRAY and the other is of type TREE. The user has been granted an exemption on the rule DB2LBACREADTREE, which is the rule used for read access when comparing values of components of type TREE. If the user attempts to read protected data then whatever value the user has for the TREE component, even if it is an empty value, will not block access because that rule is not used. Whether the user can read the data depends entirely on the values of the ARRAY component of the labels.

LBAC rule sets overview

An LBAC rule set is a predefined set of rules that are used when comparing security labels. When the values of a two security labels are being compared, one or more of the rules in the rule set will be used to determine if one value blocks another.

Each LBAC rule set is identified by a unique name. When you create a security policy you must specify the LBAC rule set that will be used with that policy. Any comparison of security labels that are part of that policy will use that LBAC rule set.

Each rule in a rule set is also identified by a unique name. You use the name of a rule when you are granting an exemption on that rule.

How many rules are in a set and when each rule is used can vary from rule set to rule set.

There is currently only one supported LBAC rule set. The name of that rule set is DB2LBACRULES.

LBAC rule set: DB2LBACRULES

The DB2LBACRULES LBAC rule set provides a traditional set of rules for comparing the values of security label components. It protects from both write-up and write-down.

What are write-up and write down?

Write-up and write-down apply only to components of type ARRAY and only to write access. Write up occurs when the value protecting data that you are writing to is higher than your value. Write-down is when the value protecting the data is lower than yours. By default neither write-up nor write-down is allowed, meaning that you can only write data that is protected by the same value that you have.

When comparing two values for the same component, which rules are used depends on the type of the component (ARRAY, SET, or TREE) and what type of access is being attempted (read, or write). This table lists the rules, tells when each is used, and describes how the rule determines if access is blocked.

Table 10. Summary of the DB2LBACRULES rules

Rule name	Used when comparing the values of this type of component	Used when attempting this type of access	Access is blocked when this condition is met
DB2LBACREADARRAY	ARRAY	Read	The user's value is lower than the protecting value.
DB2LBACREADSET	SET	Read	There are one or more protecting values that the user does not hold.
DB2LBACREADTREE	TREE	Read	None of the user's values is equal to or an ancestor of one of the protecting values.
DB2LBACWRITEARRAY	ARRAY	Write	The user's value is higher than the protecting value or lower than the protecting value.[1]
DB2LBACWRITESET	SET	Write	There are one or more protecting values that the user does not hold.
DB2LBACWRITETREE	TREE	Write	None of the user's values is equal to or an ancestor of one of the protecting values.

Note:

1. The DB2LBACWRITEARRAY rule can be thought of as being two different rules combined. One prevents writing to data that is higher than your level (write-up) and the other prevents writing to data that is lower than your level (write-down). When granting an exemption to this rule you can exempt the user from either of these rules or from both.

How the rules handle empty values

All rules treat empty values the same way. An empty value blocks no other values and is blocked by any non-empty value.

DB2LBACREADSET and DB2LBACWRITESET examples

These examples are valid for a user trying to read or trying to write protected data. They assume that the values are for a component of type SET that has these elements: one two three four

Table 11. Examples of applying the DB2LBACREADSET and DB2LBACWRITESET rules.

User's value	Protecting value	Access blocked?
'one'	'one'	Not blocked. The values are the same.
'(one,two,three)'	'one'	Not blocked. The user's value contains the element 'one'.
'(one,two)'	'(one,two,four)'	Blocked. The element 'four' is in the protecting value but not in the user's value.
'()'	'one'	Blocked. An empty value is blocked by any non-empty value.
'one'	'()'	Not blocked. No value is blocked by an empty value.
'()'	'()'	Not blocked. No value is blocked by an empty value.

DB2LBACREADTREE and DB2LBACWRITETREE

These examples are valid for both read access and write access. They assume that the values are for a component of type TREE that was defined in this way:

```
CREATE SECURITY LABEL COMPONENT mycomp
TREE (
    'Corporate'      ROOT,
    'Publishing'     UNDER 'Corporate',
    'Software'       UNDER 'Corporate',
    'Development'    UNDER 'Software',
    'Sales'          UNDER 'Software',
    'Support'        UNDER 'Software'
    'Business Sales' UNDER 'Sales'
    'Home Sales'     UNDER 'Sales'
)
```

This means the elements are in this arrangement:

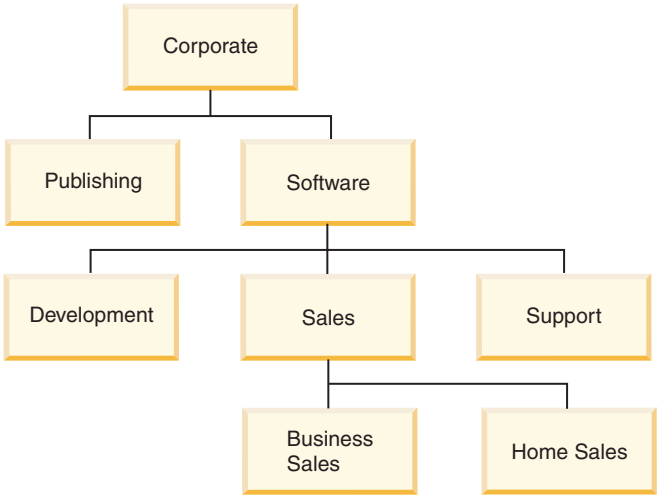


Table 12. Examples of applying the DB2LBACREADTREE and DB2LBACWRITETREE rules.

User's value	Protecting value	Access blocked?
'(Support,Sales)'	'Development'	Blocked. The element 'Development' is not one of the user's values and neither 'Support' nor 'Sales' is an ancestor of 'Development'.
'(Development,Software)'	'(Business Sales,Publishing)'	Not blocked. The element 'Software' is an ancestor of 'Business Sales'.
'(Publishing,Sales)'	'(Publishing,Support)'	Not blocked. The element 'Publishing' is in both sets of values.
'Corporate'	'Development'	Not blocked. The root value is an ancestor of all other values.
'()'	'Sales'	Blocked. An empty value is blocked by any non-empty value.
'Home Sales'	'()'	Not blocked. No value is blocked by an empty value.
'()'	'()'	Not blocked. No value is blocked by an empty value.

DB2LBACREADARRAY examples

These examples are for read access only. They assume that the values are for a component of type ARRAY that includes these elements in this arrangement:

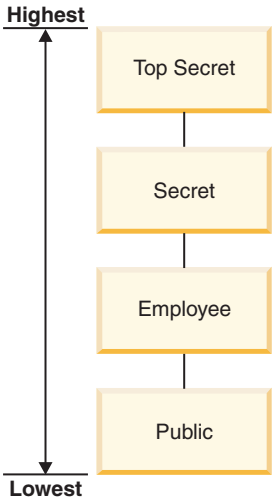


Table 13. Examples of applying the DB2LBACREADARRAY rule.

User's value	Protecting value	Read access blocked?
'Secret'	'Employee'	Not blocked. The element 'Secret' is higher than the element 'Employee'.
'Secret'	'Secret'	Not blocked. The values are the same.
'Secret'	'Top Secret'	Blocked. The element 'Top Secret' is higher than the element 'Secret'.
'()'	'Public'	Blocked. An empty value is blocked by any non-empty value.
'Public'	'()'	Not blocked. No value is blocked by an empty value.
'()'	'()'	Not blocked. No value is blocked by an empty value.

DB2LBACWRITEARRAY examples

These examples are for write access only. They assume that the values are for a component of type ARRAY that includes these elements in this arrangement:

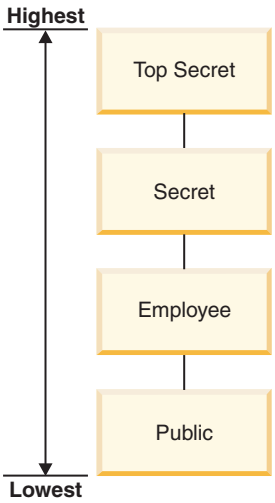


Table 14. Examples of applying the DB2LBACWRITEARRAY rule.

User's value	Protecting value	Write access blocked?
'Secret'	'Employee'	Blocked. The element 'Employee' is lower than the element 'Secret'.
'Secret'	'Secret'	Not blocked. The values are the same.
'Secret'	'Top Secret'	Blocked. The element 'Top Secret' is higher than the element 'Secret'.
'()'	'Public'	Blocked. An empty value is blocked by any non-empty value.
'Public'	'()'	Not blocked. No value is blocked by an empty value.
'()'	'()'	Not blocked. No value is blocked by an empty value.

LBAC rule exemptions

When you hold an LBAC rule exemption on a particular rule of a particular security policy, that rule is not enforced when you try to access data protected by that security policy.

An exemption has no effect when comparing security labels of any security policy other than the one for which it was granted.

Example:

There are two tables: T1 and T2. T1 is protected by security policy P1 and T2 is protected by security policy P2. Both security policies have one component. The component of each is of type ARRAY. T1 and T2 each contain only one row of data. The security label that you hold for read access under security policy P1 does not allow you access to the row in T1. The security label that you hold for read access under security policy P2 does not allow you read access to the row in T2.

Now you are granted an exemption on DB2LBACREADARRAY under P1. You can now read the row from T1 but not the row from T2 because T2 is protected by a different security policy and you do not hold an exemption to the DB2LBACREADARRAY rule in that policy.

You can hold multiple exemptions. If you hold an exemption to every rule used by a security policy then you will have complete access to all data protected by that security policy.

Granting LBAC rule exemptions

You must be a security administrator to grant an LBAC rule exemption. To grant an LBAC rule exemption, use the SQL statement GRANT EXEMPTION ON RULE.

When you grant an LBAC rule exemption you provide this information:

- The rule or rules that the exemption is for
- The security policy that the exemption is for
- The user, group, or role to which you are granting the exemption

Important: LBAC rule exemptions provide very powerful access. Do not grant them without careful consideration.

Revoking LBAC rule exemptions

You must be a security administrator to revoke an LBAC rule exemption. To revoke an LBAC rule exemption, use the SQL statement REVOKE EXEMPTION ON RULE.

Built-in functions for managing LBAC security labels

The built-in functions SECLABEL, SECLABEL_BY_NAME, and SECLABEL_TO_CHAR are provided for managing label-based access control (LBAC) security labels.

Each is described briefly here and in detail in the *SQL Reference*

SECLABEL

This built-in function is used to build a security label by specifying a security policy and values for each of the components in the label. The returned value has a data type of DB2SECURITYLABEL and is a security label that is part of the indicated security policy and has the indicated values for the components. It is not necessary that a security label with the indicated values already exists.

Example: Table T1 has two columns, the first has a data type of DB2SECURITYLABEL and the second has a data type of INTEGER. T1 is protected by security policy P1, which has three security label components: level, departments, and groups. If UNCLASSIFIED is an element of the component level, ALPHA and SIGMA are both elements of the component departments, and G2 is an element of the component groups then a security label could be inserted like this:

```
INSERT INTO T1 VALUES
   ( SECLABEL( 'P1', 'UNCLASSIFIED:(ALPHA,SIGMA):G2' ), 22 )
```

SECLABEL_BY_NAME

This built-in function accepts the name of a security policy and the name of a security label that is part of that security policy. It then returns the indicated security label as a DB2SECURITYLABEL. You must use this function when inserting an existing security label into a column that has a data type of DB2SECURITYLABEL.

Example: Table T1 has two columns, the first has a data type of DB2SECURITYLABEL and the second has a data type of INTEGER. The security label named L1 is part of security policy P1. This SQL inserts the security label:

```
INSERT INTO T1 VALUES ( SECLABEL_BY_NAME( 'P1', 'L1' ), 22 )
```

This SQL statement does not work:

```
INSERT INTO T1 VALUES ( P1.L1, 22 )     // Syntax Error!
```

SECLABEL_TO_CHAR

This built-in function returns a string representation of the values that make up a security label.

Example: Column C1 in table T1 has a data type of DB2SECURITYLABEL. T1 is protected by security policy P1, which has three security label components: level, departments, and groups. There is one row in T1 and the value in column C1 that has these elements for each of the components:

Component	Elements
level	SECRET
departments	DELTA and SIGMA
groups	G3

A user that has LBAC credentials that allow reading the row executes this SQL statement:

```
SELECT SECLABEL_TO_CHAR( 'P1', C1 ) AS C1 FROM T1
```

The output looks like this:

```
C1

'SECRET:(DELTA,SIGMA):G3'
```

Protection of data using LBAC

Label-based access control (LBAC) can be used to protect rows of data, columns of data, or both. Data in a table can only be protected by security labels that are part of the security policy protecting the table. Data protection, including adding a security policy, can be done when creating the table or later by altering the table.

You can add a security policy to a table and protect data in that table as part of the same CREATE TABLE or ALTER TABLE statement.

As a general rule you are not allowed to protect data in such a way that your current LBAC credentials do not allow you to write to that data.

Adding a security policy to a table

You can add a security policy to a table when you create the table by using the SECURITY POLICY clause of the CREATE TABLE statement. You can add a security policy to an existing table by using the ADD SECURITY POLICY clause of the ALTER TABLE statement. You do not need to have SECADM authority or have LBAC credentials to add a security policy to a table.

Security policies cannot be added to types of tables that cannot be protected by LBAC. See the overview of LBAC for a list of table types that cannot be protected by LBAC.

No more than one security policy can be added to any table.

Protecting rows

You can allow protected rows in a new table by including a column with a data type of DB2SECURITYLABEL when you create the table. The CREATE TABLE statement must also add a security policy to the table. You do not need to have SECADM authority or have any LBAC credentials to create such a table.

You can allow protected rows in an existing table by adding a column that has a data type of DB2SECURITYLABEL. To add such a column, either the table must already be protected by a security policy or the ALTER TABLE statement that adds the column must also add a security policy to the table. When the column is added, the security label you hold for write access is used to protect all existing rows. If you do not hold a security label for write access that is part of the security policy protecting the table then you cannot add a column that has a data type of DB2SECURITYLABEL.

After a table has a column of type DB2SECURITYLABEL you protect each new row of data by storing a security label in that column. The details of how this works are described in the topics about inserting and updating LBAC protected data. You must have LBAC credentials to insert rows into a table that has a column of type DB2SECURITYLABEL.

A column that has a data type of DB2SECURITYLABEL cannot be dropped and cannot be changed to any other data type.

Protecting columns

You can protect a column when you create the table by using the SECURED WITH column option of the CREATE TABLE statement. You can add protection to an existing column by using the SECURED WITH option in an ALTER TABLE statement.

To protect a column with a particular security label you must have LBAC credentials that allow you to write to data protected by that security label. You do not have to have SECADM authority.

Columns can only be protected by security labels that are part of the security policy protecting the table. You cannot protect columns in a table that has no security policy. You are allowed to protect a table with a security policy and protect one or more columns in the same statement.

You can protect any number of the columns in a table but a column can be protected by no more than one security label.

Reading of LBAC protected data

When you try to read data protected by label-based access control (LBAC), your LBAC credentials for reading are compared to the security label that is protecting the data. If the protecting label does not block your credentials you are allowed to read the data.

In the case of a protected column the protecting security label is defined in the schema of the table. The protecting security label for that column is the same for every row in the table. In the case of a protected row the protecting security label is stored in the row in a column of type DB2SECURITYLABEL. It can be different for every row in the table.

The details of how your LBAC credentials are compared to a security label are given in the topic about how LBAC security labels are compared.

Reading protected columns

When you try to read from a protected column your LBAC credentials are compared with the security label protecting the column. Based on this comparison access will either be blocked or allowed. If access is blocked then an error is returned and the statement fails. Otherwise, the statement proceeds as usual.

Trying to read a column that your LBAC credentials do not allow you to read, causes the entire statement to fail.

Example:

Table T1 has two protected columns. The column C1 is protected by the security label L1. The column C2 is protected by the security label L2.

Assume that user Jyoti has LBAC credentials for reading that allow access to security label L1 but not to L2. If Jyoti issues the following SQL statement, the statement will fail:

```
SELECT * FROM T1
```

The statement fails because column C2 is included in the SELECT clause as part of the wildcard (*).

If Jyoti issues the following SQL statement it will succeed:

```
SELECT C1 FROM T1
```

The only protected column in the SELECT clause is C1, and Jyoti's LBAC credentials allow her to read that column.

Reading protected rows

If you do not have LBAC credentials that allow you to read a row it is as if that row does not exist for you.

When you read protected rows, only those rows to which your LBAC credentials allow read access are returned. This is true even if the column of type DB2SECURITYLABEL is not part of the SELECT clause.

Depending on their LBAC credentials, different users might see different rows in a table that has protected rows. For example, two users executing the statement `SELECT COUNT(*) FROM T1` may get different results if T1 has protected rows and the users have different LBAC credentials.

Your LBAC credentials affect not only SELECT statements but also other SQL statements like UPDATE, and DELETE. If you do not have LBAC credentials that allow you to read a row, you cannot affect that row.

Example:

Table T1 has these rows and columns. The column ROWSECURITYLABEL has a data type of DB2SECURITYLABEL.

Table 15.

LASTNAME	DEPTNO	ROWSECURITYLABEL
Rjaibi	55	L2

Table 15. (continued)

LASTNAME	DEPTNO	ROWSECURITYLABEL
Miller	77	L1
Fielding	11	L3
Bird	55	L2

Assume that user Dan has LBAC credentials that allow him to read data that is protected by security label L1 but not data protected by L2 or L3.

Dan issues the following SQL statement:

```
SELECT * FROM T1
```

The SELECT statement returns only the row for Miller. No error messages or warning are returned.

Dan's view of table T1 is this:

Table 16.

LASTNAME	DEPTNO	ROWSECURITYLABEL
Miller	77	L1

The rows for Rjaibi, Fielding, and Bird are not returned because read access is blocked by their security labels. Dan cannot delete or update these rows. They will also not be included in any aggregate functions. For Dan it is as if those rows do not exist.

Dan issues this SQL statement:

```
SELECT COUNT(*) FROM T1
```

The statement returns a value of 1 because only the row for Miller can be read by the user Dan.

Reading protected rows that contain protected columns

Column access is checked before row access. If your LBAC credentials for read access are blocked by the security label protecting one of the columns you are selecting then the entire statement fails. If not, the statement continues and only the rows protected by security labels to which your LBAC credentials allow read access are returned.

Example

The column LASTNAME of table T1 is protected with the security label L1. The column DEPTNO is protected with security label L2. The column ROWSECURITYLABEL has a data type of DB2SECURITYLABEL. T1, including the data, looks like this:

Table 17.

LASTNAME *Protected by L1*	DEPTNO *Protected by L2*	ROWSECURITYLABEL
Rjaibi	55	L2

Table 17. (continued)

LASTNAME *Protected by L1*	**DEPTNO** *Protected by L2*	**ROWSECURITYLABEL**
Miller	77	L1
Fielding	11	L3

Assume that user Sakari has LBAC credentials that allow reading data protected by security label L1 but not L2 or L3.

Sakari issues this SQL statement:

```
SELECT * FROM T1
```

The statement fails because the SELECT clause uses the wildcard (*) which includes the column DEPTNO. The column DEPTNO is protected by security label L2, which Sakari's LBAC credentials do not allow her to read.

Sakari next issues this SQL statement:

```
SELECT LASTNAME, ROWSECURITYLABEL FROM T1
```

The select clause does not include any columns that Sakari is not able to read so the statement continues. Only one row is returned, however, because each of the other rows is protected by security label L2 or L3.

Table 18.

LASTNAME	**ROWSECURITYLABEL**
Miller	L1

Inserting of LBAC protected data

Inserting to protected columns

When you try to explicitly insert data to a protected column your LBAC credentials for writing are compared with the security label protecting that column. Based on this comparison access will either be blocked or allowed.

The details of how two security labels are compared are given in the topic about how LBAC security labels are compared.

If access is allowed, the statement proceeds as usual. If access is blocked, then the insert fails and an error is returned.

If you are inserting a row but do not provide a value for a protected column then a default value is inserted if one is available. This happens even if your LBAC credentials do not allow write access to that column. A default is available in the following cases:

- The column was declared with the WITH DEFAULT option
- The column is a generated column
- The column has a default value that is given through a BEFORE trigger
- The column has a data type of DB2SECURITYLABEL, in which case security label that you hold for write access is the default value

Inserting to protected rows

When you insert a new row into a table with protected rows, you do not have to provide a value for the column that is of type DB2SECURITYLABEL. If you do not provide a value for that column, the column is automatically populated with the security label you have been granted for write access. If you have not been granted a security label for write access, an error is returned and the insert fails.

By using built-in functions like SECLABEL, you can explicitly provide a security label to be inserted in a column of type DB2SECURITYLABEL. The provided security label is only used, however, if your LBAC credentials would allow you to write to data that is protected with the security label you are trying to insert.

If you provide a security label that you would not be able to write, then what happens depends on the security policy that is protecting the table. If the security policy has the RESTRICT NOT AUTHORIZED WRITE SECURITY LABEL option, then the insert fails and an error is returned. If the security policy does not have the RESTRICT NOT AUTHORIZED WRITE SECURITY LABEL option or if it instead has the OVERRIDE NOT AUTHORIZED WRITE SECURITY LABEL option, then the security label you provide is ignored and if you hold a security label for write access, it is used instead. If you do not hold a security label for write access, an error is returned.

Examples

Table T1 is protected by a security policy named P1 that was created without the RESTRICT NOT AUTHORIZED WRITE SECURITY LABEL option. Table T1 has two columns but no rows. The columns are LASTNAME and LABEL. The column LABEL has a data type of DB2SECURITYLABEL.

User Joe holds a security label L2 for write access. Assume that the security label L2 allows him to write to data protected by security label L2 but not to data protected by security labels L1 or L3.

Joe issues the following SQL statement:

```
INSERT INTO T1 (LASTNAME, DEPTNO) VALUES ('Rjaibi', 11)
```

Because no security label was included in the INSERT statement, Joe's security label for write access is inserted into the LABEL row.

Table T1 now looks like this:

Table 19.

LASTNAME	LABEL
Rjaibi	L2

Joe issues the following SQL statement, in which he explicitly provides the security label to be inserted into the column LABEL:

```
INSERT INTO T1 VALUES ('Miller', SECLABEL_BY_NAME('P1', 'L1') )
```

The SECLABEL_BY_NAME function in the statement returns a security label that is part of security policy P1 and is named L1. Joe is not allowed to write to data that is protected with L1 so he is not allowed to insert L1 into the column LABEL.

Because the security policy protecting T1 was created without the RESTRICT NOT AUTHORIZED WRITE SECURITY LABEL option the security label that Joe holds for writing is inserted instead. No error or message is returned.

The table now looks like this:

Table 20.

LASTNAME	LABEL
Rjaibi	L2
Miller	L2

If the security policy protecting the table had been created with the RESTRICT NOT AUTHORIZED WRITE SECURITY LABEL option then the insert would have failed and an error would have been returned.

Next Joe is granted an exemption to one of the LBAC rules. Assume that his new LBAC credentials allow him to write to data that is protected with security labels L1 and L2. The security label granted to Joe for write access does not change, it is still L2.

Joe issues the following SQL statement:

```
INSERT INTO T1 VALUES ('Bird', SECLABEL_BY_NAME('P1', 'L1') )
```

Because of his new LBAC credentials Joe is able to write to data that is protected by the security label L1. The insertion of L1 is therefore allowed. The table now looks like this:

Table 21.

LASTNAME	LABEL
Rjaibi	L2
Miller	L2
Bird	L1

Updating of LBAC protected data

Your LBAC credentials must allow you write access to data before you can update it. In the case of updating a protected row, your LBAC credentials must also allow read access to the row.

Updating protected columns

When you try to update data in a protected column, your LBAC credentials are compared to the security label protecting the column. The comparison made is for write access. If write access is blocked then an error is returned and the statement fails, otherwise the update continues.

The details of how your LBAC credentials are compared to a security label are given in the topic about how LBAC security labels are compared.

Example:

Assume there is a table T1 in which column DEPTNO is protected by a security label L2 and column PAYSCALE is protected by a security label L3. T1, including its data, looks like this:

Table 22. Table T1

EMPNO	**LASTNAME**	**DEPTNO** *Protected by* **L2**	**PAYSCALE** *Protected by* **L3**
1	Rjaibi	11	4
2	Miller	11	7
3	Bird	11	9

User Lhakpa has no LBAC credentials. He issues this SQL statement:

```
UPDATE T1 SET EMPNO = 4
   WHERE LASTNAME = "Bird"
```

This statement executes without error because it does not update any protected columns. T1 now looks like this:

Table 23. Table T1 After Update

EMPNO	**LASTNAME**	**DEPTNO** *Protected by* **L2**	**PAYSCALE** *Protected by* **L3**
1	Rjaibi	11	4
2	Miller	11	7
4	Bird	11	9

Lhakpa next issues this SQL statement:

```
UPDATE T1 SET DEPTNO = 55
   WHERE LASTNAME = "Miller"
```

This statement fails and an error is returned because DEPTNO is protected and Lhakpa has no LBAC credentials.

Assume Lhakpa is granted LBAC credentials and that allow the access summarized in the following table. The details of what those credentials are and what elements are in the security labels are not important for this example.

Security label protecting the data	**Can read?**	**Can Write?**
L2	No	Yes
L3	No	No

Lhakpa issues this SQL statement again:

```
UPDATE T1 SET DEPTNO = 55
   WHERE LASTNAME = "Miller"
```

This time the statement executes without error because Lhakpa's LBAC credentials allow him to write to data protected by the security label that is protecting the column DEPTNO. It does not matter that he is not able to read from that same

column. The data in T1 now looks like this:

Table 24. Table T1 After Second Update

EMPNO	**LASTNAME**	**DEPTNO** *Protected by* **L2**	**PAYSCALE** *Protected by* **L3**
1	Rjaibi	11	4
2	Miller	55	7
4	Bird	11	9

Next Lhakpa issues this SQL statement:

```
UPDATE T1 SET DEPTNO = 55, PAYSCALE = 4
   WHERE LASTNAME = "Bird"
```

The column PAYSCALE is protected by the security label L3 and Lhakpa's LBAC credentials do not allow him to write to it. Because Lhakpa is unable to write to the column, the update fails and no data is changed.

Updating protected rows

If your LBAC credentials do not allow you to read a row, then it is as if that row does not exist for you so there is no way for you to update that row. For rows that you are able to read, you must also be able to write to the row in order to update it.

When you try to update a row, your LBAC credentials for writing are compared to the security label protecting the row. If write access is blocked, the update fails and an error is returned. If write access is not blocked, then the update continues.

The update that is performed is done the same way as an update to a non-protected row except for the treatment of the column that has a data type of DB2SECURITYLABEL. If you do not explicitly set the value of that column, it is automatically set to the security label that you hold for write access. If you do not have a security label for write access, an error is returned and the statement fails.

If the update explicitly sets the column that has a data type of DB2SECURITYLABEL, then your LBAC credentials are checked again. If the update you are trying to perform would create a row that your current LBAC credentials would not allow you to write to, then what happens depends on the security policy that is protecting the table. If the security policy has the RESTRICT NOT AUTHORIZED WRITE SECURITY LABEL option, then the update fails and an error is returned. If the security policy does not have the RESTRICT NOT AUTHORIZED WRITE SECURITY LABEL option or if it instead has the OVERRIDE NOT AUTHORIZED WRITE SECURITY LABEL option, then the security label you provide is ignored and if you hold a security label for write access, it is used instead. If you do not hold a security label for write access, an error is returned.

Example:

Assume that table T1 is protected by a security policy named P1 and has a column named LABEL that has a data type of DB2SECURITYLABEL.

T1, including its data, looks like this:

Table 25. Table T1

EMPNO	LASTNAME	DEPTNO	LABEL
1	Rjaibi	11	L1
2	Miller	11	L2
3	Bird	11	L3

Assume that user Jenni has LBAC credentials that allow her to read and write data protected by the security labels L0 and L1 but not data protected by any other security labels. The security label she holds for both read and write is L0. The details of her full credentials and of what elements are in the labels are not important for this example.

Jenni issues this SQL statement:

```
SELECT * FROM T1
```

Jenni sees only one row in the table:

Table 26. Jenni's SELECT Query Result

EMPNO	LASTNAME	DEPTNO	LABEL
1	Rjaibi	11	L1

The rows protected by labels L2 and L3 are not included in the result set because Jenni's LBAC credentials do not allow her to read those rows. For Jenni it is as if those rows do not exist.

Jenni issues these SQL statements:

```
UPDATE T1 SET DEPTNO = 44 WHERE DEPTNO = 11;
SELECT * FROM T1;
```

The result set returned by the query looks like this:

Table 27. Jenni's UPDATE & SELECT Query Result

EMPNO	LASTNAME	DEPTNO	LABEL
1	Rjaibi	44	L0

The actual data in the table looks like this:

Table 28. Table T1

EMPNO	LASTNAME	DEPTNO	LABEL
1	Rjaibi	44	L0
2	Miller	11	L2
3	Bird	11	L3

The statement executed without error but affected only the first row. The second and third rows are not readable by Jenni so they are not selected for update by the statement even though they meet the condition in the WHERE clause.

Notice that the value of the LABEL column in the updated row has changed even though that column was not explicitly set in the UPDATE statement. The column was set to the security label that Jenni held for writing.

Now Jenni is granted LBAC credentials that allow her to read data protected by any security label. Her LBAC credentials for writing do not change. She is still only able to write to data protected by L0 and L1.

Jenni again issues this SQL statement:

```
UPDATE T1 SET DEPTNO = 44 WHERE DEPTNO = 11
```

This time the update fails because of the second and third rows. Jenni is able to read those rows, so they are selected for update by the statement. She is not, however, able to write to them because they are protected by security labels L2 and L3. The update does not occur and an error is returned.

Jenni now issues this SQL statement:

```
UPDATE T1
 SET DEPTNO = 55, LABEL = SECLABEL_BY_NAME( 'P1', 'L2' )
 WHERE LASTNAME = "Rjaibi"
```

The SECLABEL_BY_NAME function in the statement returns the security label named L2. Jenni is trying to explicitly set the security label protecting the first row. Jenni's LBAC credentials allow her to read the first row, so it is selected for update. Her LBAC credentials allow her to write to rows protected by the security label L0 so she is allowed to update the row. Her LBAC credentials would not, however, allow her to write to a row protected by the security label L2, so she is not allowed to set the column LABEL to that value. The statement fails and an error is returned. No columns in the row are updated.

Jenni now issues this SQL statement:

```
UPDATE T1 SET LABEL = SECLABEL_BY_NAME( 'P1', 'L1' ) WHERE LASTNAME = "Rjaibi"
```

The statement succeeds because she would be able to write to a row protected by the security label L1.

T1 now looks like this:

Table 29. Table T1

EMPNO	LASTNAME	DEPTNO	LABEL
1	Rjaibi	44	L1
2	Miller	11	L2
3	Bird	11	L3

Updating protected rows that contain protected columns

If you try to update protected columns in a table with protected rows then your LBAC credentials must allow writing to of all of the protected columns affected by the update, otherwise the update fails and an error is returned. This is as described in section about updating protected columns, earlier. If you are allowed to update all of the protected columns affected by the update you will still only be able to update rows that your LBAC credentials allow you to both read from and write to. This is as described in the section about updating protected rows, earlier. The

handling of a column with a data type of DB2SECURITYLABEL is the same whether the update affects protected columns or not.

If the column that has a data type of DB2SECURITYLABEL is itself a protected column then your LBAC credentials must allow you to write to that column or you cannot update any of the rows in the table.

Deleting or dropping of LBAC protected data

If your LBAC credentials do not allow you to read a row then it is as if that row does not exist for you so there is no way for you to delete it. To delete a row that you are able to read, your LBAC credentials must also allow you to write to the row. To delete any row in a table that has protected columns you must have LBAC credentials that allow you to write to all protected columns in the table.

Deleting protected rows

When you try to delete a row, your LBAC credentials for writing are compared to the security label protecting the row. If the protecting security label blocks write access by your LBAC credentials, the DELETE statement fails, an error is returned, and no rows are deleted.

Example

Protected table T1 has these rows:

LASTNAME	DEPTNO	LABEL
Rjaibi	55	L2
Miller	77	L1
Bird	55	L2
Fielding	77	L3

Assume that user Pat has LBAC credentials such that her access is as summarized in this table:

Security label	Read access?	Write access?
L1	Yes	Yes
L2	Yes	No
L3	No	No

The exact details of her LBAC credentials and of the security labels are unimportant for this example.

Pat issues the following SQL statement:

```
SELECT * FROM T1 WHERE DEPTNO != 999
```

The statement executes and returns this result set:

LASTNAME	DEPTNO	LABEL
Rjaibi	55	L2
Miller	77	L1

LASTNAME	DEPTNO	LABEL
Bird	55	L2

The last row of T1 is not included in the results because Pat does not have read access to that row. It is as if that row does not exist for Pat.

Pat issues this SQL statement:

```
DELETE FROM T1 WHERE DEPTNO != 999
```

Pat does not have write access to the first or third row, both of which are protected by L2. So even though she can read the rows she cannot delete them. The DELETE statement fails and no rows are deleted.

Pat issues this SQL statement:

```
DELETE FROM T1 WHERE DEPTNO = 77;
```

This statement succeeds because Pat is able to write to the row with Miller in the LASTNAME column. That is the only row selected by the statement. The row with Fielding in the LASTNAME column is not selected because Pat's LBAC credentials do not allow her to read that row. That row is never considered for the delete so no error occurs.

The actual rows of the table now look like this:

LASTNAME	DEPTNO	LABEL
Rjaibi	55	L2
Bird	55	L2
Fielding	77	L3

Deleting rows that have protected columns

To delete any row in a table that has protected columns you must have LBAC credentials that allow you to write to all protected columns in the table. If there is any row in the table that your LBAC credentials do not allow you to write to then the delete will fail and an error will be returned.

If the table has both protected columns and protected rows then to delete a particular row you must have LBAC credentials that allow you to write to every protected column in the table and also to read from and write to the row that you want to delete.

Example

In protected table T1, the column DEPTNO is protected by the security label L2. T1 contains these rows:

LASTNAME	DEPTNO *Protected by* **L2**	LABEL
Rjaibi	55	L2
Miller	77	L1

LASTNAME	DEPTNO *Protected by* **L2**	LABEL
Bird	55	L2
Fielding	77	L3

Assume that user Benny has LBAC credentials that allow him the access summarized in this table:

Security label	Read access?	Write access?
L1	Yes	Yes
L2	Yes	No
L3	No	No

The exact details of his LBAC credentials and of the security labels are unimportant for this example.

Benny issues the following SQL statement:

```
DELETE FROM T1 WHERE DEPTNO = 77
```

The statement fails because Benny does not have write access to the column DEPTNO.

Now Benny's LBAC credentials are changed so that he has access as summarized in this table:

Security label	Read access?	Write access?
L1	Yes	Yes
L2	Yes	Yes
L3	Yes	No

Benny issues this SQL statement again:

```
DELETE FROM T1 WHERE DEPTNO = 77
```

This time Benny has write access to the column DEPTNO so the delete continues. The delete statement selects only the row that has a value of Miller in the LASTNAME column. The row that has a value of Fielding in the LASTNAME column is not selected because Benny's LBAC credentials do not allow him to read that row. Because the row is not selected for deletion by the statement it does not matter that Benny is unable to write to the row.

The one row selected is protected by the security label L1. Benny's LBAC credentials allow him to write to data protected by L1 so the delete is successful.

The actual rows in table T1 now look like this:

LASTNAME	DEPTNO *Protected by* **L2**	LABEL
Rjaibi	55	L2

LASTNAME	DEPTNO *Protected by* **L2**	LABEL
Bird	55	L2
Fielding	77	L3

Dropping protected data

You cannot drop a column that is protected by a security label unless your LBAC credentials allow you to write to that column.

A column with a data type of DB2SECURITYLABEL cannot be dropped from a table. To remove it you must first drop the security policy from the table. When you drop the security policy the table is no longer protected with LBAC and the data type of the column is automatically changed from DB2SECURITYLABEL to VARCHAR(128) FOR BIT DATA. The column can then be dropped.

Your LBAC credentials do not prevent you from dropping entire tables or databases that contain protected data. If you would normally have permission to drop a table or a database you do not need any LBAC credentials to do so, even if the database contains protected data.

Removal of LBAC protection from data

You must have SECADM authority to remove the security policy from a table. To remove the security policy from a table you use the DROP SECURITY POLICY clause of the ALTER TABLE statement. This also automatically removes protection from all rows and all columns of the table.

Removing protection from rows

In a table that has protected rows every row must be protected by a security label. There is no way to remove LBAC protection from individual rows.

A column of type DB2SECURITYLABEL cannot be altered or removed except by removing the security policy from the table.

Removing protection from columns

Protection of a column can be removed using the DROP COLUMN SECURITY clause of the SQL statement ALTER TABLE. To remove the protection from a column you must have LBAC credentials that allow you to read from and write to that column in addition to the normal privileges and authorities needed to alter a table.

Chapter 5. Using the system catalog for security information

Information about each database is automatically maintained in a set of views called the system catalog, which is created when the database is created. This system catalog describes tables, columns, indexes, programs, privileges, and other objects.

The following views and table functions list information about privileges held by users, identities of users granting privileges, and object ownership:

SYSCAT.DBAUTH
Lists the database privileges

SYSCAT.TABAUTH
Lists the table and view privileges

SYSCAT.COLAUTH
Lists the column privileges

SYSCAT.PACKAGEAUTH
Lists the package privileges

SYSCAT.INDEXAUTH
Lists the index privileges

SYSCAT.SCHEMAAUTH
Lists the schema privileges

SYSCAT.PASSTHRUAUTH
Lists the server privilege

SYSCAT.ROUTINEAUTH
Lists the routine (functions, methods, and stored procedures) privileges

SYSCAT.SURROGATEAUTHIDS
Lists the authorization IDs for which another authorization ID can act as a surrogate.

Privileges granted to users by the system will have SYSIBM as the grantor. SYSADM, SYSMAINT SYSCTRL, and SYSMON are not listed in the system catalog.

The CREATE and GRANT statements place privileges in the system catalog. Users with SYSADM and DBADM authorities can grant and revoke SELECT privilege on the system catalog views.

Retrieving authorization names with granted privileges

You can use the PRIVILEGES and other administrative views to retrieve information about the authorization names that have been granted privileges in a database.

For example, the following query retrieves all explicit privileges and the authorization IDs to which they were granted, plus other information, from the PRIVILEGES administrative view:

```
SELECT AUTHID, PRIVILEGE, OBJECTNAME, OBJECTSCHEMA, OBJECTTYPE FROM SYSIBMADM.PRIVILEGES
```

The following query uses the AUTHORIZATIONIDS administrative view to find all the authorization IDs that have been granted privileges or authorities, and to show their types:

```
SELECT AUTHID, AUTHIDTYPE FROM SYSIBMADM.AUTHORIZATIONIDS
```

You can also use the SYSIBMADM.OBJECTOWNERS administrative view and the SYSPROC.AUTH_LIST_GROUPS_FOR_AUTHID table function to find security-related information.

Prior to Version 9.1, no single system catalog view contained information about all privileges. For releases earlier than version 9.1, the following statement retrieves all authorization names with privileges:

```
 SELECT DISTINCT GRANTEE, GRANTEETYPE, 'DATABASE' FROM SYSCAT.DBAUTH
UNION
SELECT DISTINCT GRANTEE, GRANTEETYPE, 'TABLE   ' FROM SYSCAT.TABAUTH
UNION
SELECT DISTINCT GRANTEE, GRANTEETYPE, 'PACKAGE ' FROM SYSCAT.PACKAGEAUTH
UNION
SELECT DISTINCT GRANTEE, GRANTEETYPE, 'INDEX   ' FROM SYSCAT.INDEXAUTH
UNION
SELECT DISTINCT GRANTEE, GRANTEETYPE, 'COLUMN  ' FROM SYSCAT.COLAUTH
UNION
SELECT DISTINCT GRANTEE, GRANTEETYPE, 'SCHEMA  ' FROM SYSCAT.SCHEMAAUTH
UNION
SELECT DISTINCT GRANTEE, GRANTEETYPE, 'SERVER  ' FROM SYSCAT.PASSTHRUAUTH
ORDER BY GRANTEE, GRANTEETYPE, 3
```

Periodically, the list retrieved by this statement should be compared with lists of user and group names defined in the system security facility. You can then identify those authorization names that are no longer valid.

Note: If you are supporting remote database clients, it is possible that the authorization name is defined at the remote client only and not on your database server machine.

Retrieving all names with DBADM authority

The following statement retrieves all authorization names that have been directly granted DBADM authority:

```
SELECT DISTINCT GRANTEE, GRANTEETYPE FROM SYSCAT.DBAUTH
   WHERE DBADMAUTH = 'Y'
```

Note: This query does not return information about authorization names that acquired DBADM authority implicitly by having SYSADM authority.

Retrieving names authorized to access a table

You can use the PRIVILEGES and other administrative views to retrieve information about the authorization names that have been granted privileges in a database.

The following statement retrieves all authorization names (and their types) that are directly authorized to access the table EMPLOYEE with the qualifier JAMES:

```
SELECT DISTINCT AUTHID, AUTHIDTYPE FROM SYSIBMADM.PRIVILEGES
    WHERE OBJECTNAME = 'EMPLOYEE' AND OBJECTSCHEMA = 'JAMES'
```

For releases earlier than Version 9.1, the following query retrieves the same information:

```
SELECT DISTINCT GRANTEETYPE, GRANTEE FROM SYSCAT.TABAUTH
   WHERE TABNAME = 'EMPLOYEE'
     AND TABSCHEMA = 'JAMES'
UNION
SELECT DISTINCT GRANTEETYPE, GRANTEE FROM SYSCAT.COLAUTH
   WHERE TABNAME = 'EMPLOYEE'
     AND TABSCHEMA = 'JAMES'
```

To find out who can update the table EMPLOYEE with the qualifier JAMES, issue the following statement:

```
 SELECT DISTINCT GRANTEETYPE, GRANTEE FROM SYSCAT.TABAUTH
   WHERE TABNAME = 'EMPLOYEE' AND TABSCHEMA = 'JAMES' AND
      (CONTROLAUTH  = 'Y' OR
       UPDATEAUTH IN ('G','Y'))
UNION
SELECT DISTINCT GRANTEETYPE, GRANTEE FROM SYSCAT.DBAUTH
   WHERE DBADMAUTH = 'Y'
UNION
SELECT DISTINCT GRANTEETYPE, GRANTEE FROM SYSCAT.COLAUTH
   WHERE TABNAME = 'EMPLOYEE' AND TABSCHEMA = 'JAMES' AND
   PRIVTYPE = 'U'
```

This retrieves any authorization names with DBADM authority, as well as those names to which CONTROL or UPDATE privileges have been directly granted. However, it will not return the authorization names of users who only hold SYSADM authority.

Remember that some of the authorization names may be groups, not just individual users.

Retrieving all privileges granted to users

By making queries on the system catalog views, users can retrieve a list of the privileges they hold and a list of the privileges they have granted to other users.

You can use the PRIVILEGES and other administrative views to retrieve information about the authorization names that have been granted privileges in a database. For example, the following query retrieves all the privileges granted to the current session authorization ID:

```
SELECT * FROM SYSIBMADM.PRIVILEGES
 WHERE AUTHID = SESSION_USER AND AUTHIDTYPE = 'U'
```

The keyword SESSION_USER in this statement is a special register that is equal to the value of the current user's authorization name.

For releases earlier than Version 9.1, the following examples provide similar information. For example, the following statement retrieves a list of the database privileges that have been directly granted to the individual authorization name JAMES:

```
SELECT * FROM SYSCAT.DBAUTH
 WHERE GRANTEE = 'JAMES' AND GRANTEETYPE = 'U'
```

The following statement retrieves a list of the table privileges that were directly granted by the user JAMES:

```
SELECT * FROM SYSCAT.TABAUTH
 WHERE GRANTOR  = 'JAMES'
```

The following statement retrieves a list of the individual column privileges that were directly granted by the user JAMES:

```
SELECT * FROM SYSCAT.COLAUTH
   WHERE GRANTOR  = 'JAMES'
```

Securing the system catalog view

Because the system catalog views describe every object in the database, if you have sensitive data, you might want to restrict their access.

You can use the CREATE DATABASE ... RESTRICTIVE command to create a database in which no privileges are automatically granted to PUBLIC. In this case, none of the following normal default grant actions occur:

- CREATETAB
- BINDADD
- CONNECT
- IMPLSCHEMA
- EXECUTE with GRANT on all procedures in schema SQLJ
- EXECUTE with GRANT on all functions and procedures in schema SYSPROC
- BIND on all packages created in the NULLID schema
- EXECUTE on all packages created in the NULLID schema
- CREATEIN on schema SQLJ
- CREATEIN on schema NULLID
- USE on table space USERSPACE1
- SELECT access to the SYSIBM catalog tables
- SELECT access to the SYSCAT catalog views
- SELECT access to the SYSIBMADM administrative views
- SELECT access to the SYSSTAT catalog views
- UPDATE access to the SYSSTAT catalog views

If you have created a database using the RESTRICTIVE option, and you want to check that the permissions granted to PUBLIC are limited, you can issue the following query to verify which schemas PUBLIC can access:

```
SELECT DISTINCT OBJECTSCHEMA FROM SYSIBMADM.PRIVILEGES WHERE AUTHID='PUBLIC'

OBJECTSCHEMA
------------
SYSFUN
SYSIBM
SYSPROC
```

To see what access PUBLIC still has to SYSIBM, you can issue the following query to check what privileges are granted on SYSIBM. The results show that only EXECUTE on certain procedures and functions is granted.

```
SELECT * FROM SYSIBMADM.PRIVILEGES WHERE OBJECTSCHEMA = 'SYSIBM'

AUTHID       AUTHIDTYPE  PRIVILEGE   GRANTABLE  OBJECTNAME         OBJECTSCHEMA    OBJECTTYPE
---------... ----------  ----------  ---------  ---------------... ------------... ----------
PUBLIC       G           EXECUTE     N          SQL060207192129400 SYSPROC         FUNCTION
PUBLIC       G           EXECUTE     N          SQL060207192129700 SYSPROC         FUNCTION
PUBLIC       G           EXECUTE     N          SQL060207192129701 SYSPROC
...
PUBLIC       G           EXECUTE     Y          TABLES             SYSIBM          PROCEDURE
PUBLIC       G           EXECUTE     Y          TABLEPRIVILEGES    SYSIBM          PROCEDURE
PUBLIC       G           EXECUTE     Y          STATISTICS         SYSIBM          PROCEDURE
```

```
PUBLIC      G          EXECUTE     Y          SPECIALCOLUMNS     SYSIBM          PROCEDURE
PUBLIC      G          EXECUTE     Y          PROCEDURES         SYSIBM          PROCEDURE
PUBLIC      G          EXECUTE     Y          PROCEDURECOLS      SYSIBM          PROCEDURE
PUBLIC      G          EXECUTE     Y          PRIMARYKEYS        SYSIBM          PROCEDURE
PUBLIC      G          EXECUTE     Y          FOREIGNKEYS        SYSIBM          PROCEDURE
PUBLIC      G          EXECUTE     Y          COLUMNS            SYSIBM          PROCEDURE
PUBLIC      G          EXECUTE     Y          COLPRIVILEGES      SYSIBM          PROCEDURE
PUBLIC      G          EXECUTE     Y          UDTS               SYSIBM          PROCEDURE
PUBLIC      G          EXECUTE     Y          GETTYPEINFO        SYSIBM          PROCEDURE
PUBLIC      G          EXECUTE     Y          SQLCAMESSAGE       SYSIBM          PROCEDURE
PUBLIC      G          EXECUTE     Y          SQLCAMESSAGECCSID  SYSIBM          PROCEDURE
```

Note: The SYSIBMADM.PRIVILEGES administrative view is available starting with Version 9.1 of the DB2 database manager.

For releases earlier than Version 9.1 of the DB2 database manager, during database creation, SELECT privilege on the system catalog views is granted to PUBLIC. In most cases, this does not present any security problems. For very sensitive data, however, it may be inappropriate, as these tables describe every object in the database. If this is the case, consider revoking the SELECT privilege from PUBLIC; then grant the SELECT privilege as required to specific users. Granting and revoking SELECT on the system catalog views is done in the same way as for any view, but you must have either SYSADM or DBADM authority to do this.

At a minimum, if you don't want any user to be able to know what objects other users have access to, you should consider restricting access to the following catalog and administrative views:

- SYSCAT.COLAUTH
- SYSCAT.DBAUTH
- SYSCAT.INDEXAUTH
- SYSCAT.PACKAGEAUTH
- SYSCAT.PASSTHRUAUTH
- SYSCAT.ROUTINEAUTH
- SYSCAT.SCHEMAAUTH
- SYSCAT.SECURITYLABELACCESS
- SYSCAT.SECURITYPOLICYEXEMPTIONS
- SYSCAT.SEQUENCEAUTH
- SYSCAT.SURROGATEAUTHIDS
- SYSCAT.TABAUTH
- SYSCAT.TBSPACEAUTH
- SYSCAT.XSROBJECTAUTH
- SYSIBMADM.AUTHORIZATIONIDS
- SYSIBMADM.OBJECTOWNERS
- SYSIBMADM.PRIVILEGES

This would prevent information on user privileges from becoming available to everyone with access to the database.

You should also examine the columns for which statistics are gathered. Some of the statistics recorded in the system catalog contain data values which could be sensitive information in your environment. If these statistics contain sensitive data, you may wish to revoke SELECT privilege from PUBLIC for the SYSCAT.COLUMNS and SYSCAT.COLDIST catalog views.

If you wish to limit access to the system catalog views, you could define views to let each authorization name retrieve information about its own privileges.

For example, the following view MYSELECTS includes the owner and name of every table on which a user's authorization name has been directly granted SELECT privilege:

```
CREATE VIEW MYSELECTS AS
   SELECT TABSCHEMA, TABNAME FROM SYSCAT.TABAUTH
   WHERE GRANTEETYPE = 'U'
     AND GRANTEE = USER
     AND SELECTAUTH = 'Y'
```

The keyword USER in this statement is equal to the value of the current session authorization name.

The following statement makes the view available to every authorization name:

```
GRANT SELECT ON TABLE MYSELECTS TO PUBLIC
```

And finally, remember to revoke SELECT privilege on the view and base table by issuing the following two statements:

```
REVOKE SELECT ON TABLE SYSCAT.TABAUTH FROM PUBLIC
REVOKE SELECT ON TABLE SYSIBM.SYSTABAUTH FROM PUBLIC
```

Security considerations

To successfully manage security, you need to be aware of indirect ways that users can gain access to data. Also, you need to be aware of the default privileges to certain system tables that are granted when a database is created.

Gaining access to data through indirect means

The following are indirect means through which users can gain access to data they might not be authorized for:

- **Catalog views:** The DB2 database system catalog views store metadata and statistics about database objects. Users with SELECT access to the catalog views can gain some knowledge about data that they might not be qualified for. For better security, make sure that only qualified users have access to the catalog views.

  Note: In DB2 Universal Database™ Version 8, or earlier, SELECT access on the catalog views was granted to PUBLIC by default. In DB2 Version 9.1, or later, database systems, users can choose whether SELECT access to the catalog views is granted to PUBLIC or not by using the new RESTRICTIVE option on the CREATE DATABASE command.
- **Visual explain:** Visual explain shows the access plan chosen by the query optimizer for a particular query. The visual explain information also includes statistics about columns referenced in the query. These statistics can reveal information about a table's contents.
- **Explain snapshot:** The explain snapshot is compressed information that is collected when an SQL or XQuery statement is explained. It is stored as a binary large object (BLOB) in the EXPLAIN_STATEMENT table, and contains column statistics that can reveal information about table data. For better security, access to the explain tables should be granted to qualified users only.
- **Log reader functions:** A user authorized to run a function that reads the logs can gain access to data they might not be authorized for if they are able to

understand the format of a log record. These functions read the logs:

Function	Authority needed in order to execute the function
db2ReadLog	SYSADM or DBADM
db2ReadLogNoConn	None.

- **Replication:** When you replicate data, even the protected data is reproduced at the target location. For better security, make sure that the target location is at least as secure as the source location.
- **Exception tables:** When you specify an exception table while loading data into a table, users with access to the exception table can gain information that they might not be authorized for. For better security, only grant access to the exception table to authorized users and drop the exception table as soon as you are done with it.
- **Backup table space or database:** Users with the authority to run the backup command can take a backup of a database or a table space, including any protected data, and restore the data somewhere else. The backup can include data that the user might not otherwise have access to.

  The backup command can be executed by users with SYSADM, SYSCTRL, or SYSMAINT authority.
- **Set session authorization:** In DB2 Universal Database Version 8, or earlier, a user with DBADM authority could use the SET SESSION AUTHORIZATION SQL statement to set the session authorization ID to any database user. In DB2 Version 9.1, or later, database systems a user must be explicitly authorized through the GRANT SETSESSIONUSER statement before they can set the session authorization ID.

  When migrating an existing Version 8 database to a DB2 Version 9.1, or later, database system, however, a user with existing explicit DBADM authority (for example, granted in SYSCAT.DBAUTH) will keep the ability to set the session authorization to any database user. This is allowed so that existing applications will continue to work. Being able to set the session authorization potentially allows access to all protected data. For more restrictive security, you can override this setting by executing the REVOKE SETSESSIONUSER SQL statement.
- **Statement and deadlock monitoring:** As part of the deadlock monitoring activity of DB2 database management systems, values associated with parameter markers are written to the monitoring output when the WITH VALUES clause is specified. A user with access to the monitoring output can gain access to information for which they might not be authorized.
- **Traces:** A trace can contain table data. A user with access to such a trace can gain access to information that they might not be authorized for.
- **Dump files:** To help in debugging certain problems, DB2 database products might generate memory dump files in the `sqllib\db2dump` directory. These memory dump files might contain table data. If they do, users with access to the files can gain access to information that they might not be authorized for. For better security you should limit access to the `sqllib\db2dump` directory.
- **db2dart:** The db2dart tool examines a database and reports any architectural errors that it finds. The tool can access table data and DB2 does not enforce access control for that access. A user with the authority to run the db2dart tool or with access to the db2dart output can gain access to information that they might not be authorized for.

- **REOPT bind option:** When the REOPT bind option is specified, explain snapshot information for each reoptimizable incremental bind SQL statement is placed in the explain tables at run time. The explain will also show input data values.
- **db2cat:** The db2cat tool is used to dump a table's packed descriptor. The table's packed descriptor contains statistics that can reveal information about a table's contents. A user who runs the db2cat tool or has access to the output can gain access to information that they might not be authorized for.

Default privileges granted upon creating a database

The following are the default privileges to certain system tables that are granted when a database is created:

1. SYSIBM.SYSDBAUTH
   - The database creator is granted the following privileges:
     - DBADM
     - CREATETAB
     - CREATEROLE
     - BINDADD
     - CONNECT
     - NOFENCE
     - IMPLSCHEMA
     - LOAD
     - EXTERNALROUTINE
     - QUIESCECONNECT
   - The special group PUBLIC is granted the following privileges:
     - CREATETAB
     - BINDADD
     - CONNECT
     - IMPLSCHEMA
2. SYSIBM.SYSTABAUTH
   - The special group PUBLIC is granted the following privileges:
     - SELECT on all SYSCAT and SYSIBM tables
     - SELECT and UPDATE on all SYSSTAT tables
3. SYSIBM.SYSROUTINEAUTH
   - The special group PUBLIC is granted the following privileges:
     - EXECUTE with GRANT on all procedures in schema
     - SQLJ EXECUTE with GRANT on all functions and procedures in schema SYSFUN
     - EXECUTE with GRANT on all functions and procedures in schema SYSPROC
     - EXECUTE on all table functions in schema SYSIBM
     - EXECUTE on all other procedures in schema SYSIBM
4. SYSIBM.SYSPACKAGEAUTH
   - The database creator is granted the following privileges:
     - CONTROL on all packages created in the NULLID schema
     - BIND with GRANT on all packages created in the NULLID schema

- EXECUTE with GRANT on all packages created in the NULLID schema
-
- The special group PUBLIC is granted the following privileges:
  - BIND on all packages created in the NULLID schema
  - EXECUTE on all packages created in the NULLID schema

5. SYSIBM.SCHEMAAUTH
   - The special group PUBLIC is granted the following privileges:
     - CREATEIN on schema SQLJ
     - CREATE IN on schema NULLID
6. SYSIBM.TBSPACEAUTH
   - The special group PUBLIC is granted the following privileges:
     - USE on table space USERSPACE1

Chapter 6. Firewall support

A *firewall* is a set of related programs, located at a network gateway server, that are used to prevent unauthorized access to a system or network.

There are four types of firewalls:

1. Network level, packet-filter, or screening router firewalls
2. Classical application level proxy firewalls
3. Circuit level or transparent proxy firewalls
4. Stateful multi-layer inspection (SMLI) firewalls

There are existing firewall products that incorporate one of the firewall types listed above. There are many other firewall products that incorporate some combination of the above types.

Screening router firewalls

The screening router firewall is also known as a network level or packet-filter firewall. Such a firewall works by screening incoming packets by protocol attributes. The protocol attributes screened may include source or destination address, type of protocol, source or destination port, or some other protocol-specific attributes.

For all firewall solutions (except SOCKS), you need to ensure that all the ports used by DB2 database are open for incoming and outgoing packets. DB2 database uses port 523 for the DB2 Administration Server (DAS), which is used by the DB2 database tools. Determine the ports used by all your server instances by using the services file to map the service name in the server database manager configuration file to its port number.

Application proxy firewalls

A proxy or proxy server is a technique that acts as an intermediary between a Web client and a Web server. A proxy firewall acts as a gateway for requests arriving from clients.

When client requests are received at the firewall, the final server destination address is determined by the proxy software. The application proxy translates the address, performs additional access control checking and logging as necessary, and connects to the server on behalf of the client.

The DB2 Connect product on a firewall machine can act as a proxy to the destination server. Also, a DB2 database server on the firewall, acting as a hop server to the final destination server, acts like an application proxy.

Circuit level firewalls

The circuit level firewall is also known as a transparent proxy firewall.

A transparent proxy firewall does not modify the request or response beyond what is required for proxy authentication and identification. An example of a transparent proxy firewall is SOCKS.

The DB2 database system supports SOCKS Version 4.

Stateful multi-layer inspection (SMLI) firewalls

The stateful multi-layer inspection (SMLI) firewall uses a sophisticated form of packet-filtering that examines all seven layers of the Open System Interconnection (OSI) model.

Each packet is examined and compared against known states of friendly packets. While screening router firewalls only examine the packet header, SMLI firewalls examine the entire packet including the data.

Chapter 7. Security plug-ins

Authentication for the DB2 database system is done using *security plug-ins*. A security plug-in is a dynamically-loadable library that provides authentication security services.

The DB2 database system provides the following types of plug-ins:

- Group retrieval plug-in: retrieves group membership information for a given user.
- Client authentication plug-in: manages authentication on a DB2 client.
- Server authentication plug-in: manages authentication on a DB2 server.

DB2 supports two mechanisms for plug-in authentication:

User ID/password authentication
This involves authentication using a user ID and password. The following authentication types are implemented using user ID/password authentication plug-ins:

- CLIENT
- SERVER
- SERVER_ENCRYPT
- DATA_ENCRYPT
- DATA_ENCRYPT_CMP

These authentication types determine how and where authentication of a user occurs. The authentication type used depends on the authentication type specified by the *authentication* database manager configuration parameter. If the SRVCON_AUTH parameter is specified it takes precedence over AUTHENTICATION when dealing with connect or attach operations.

GSS-API authentication
GSS-API is formally known as *Generic Security Service Application Program Interface, Version 2* (IETF RFC2743) and *Generic Security Service API Version 2: C-Bindings* (IETF RFC2744). Kerberos authentication is also implemented using GSS-API. The following authentication types are implemented using GSS-API authentication plug-ins:

- KERBEROS
- GSSPLUGIN
- KRB_SERVER_ENCRYPT
- GSS_SERVER_ENCRYPT

KRB_SERVER_ENCRYPT and GSS_SERVER_ENCRYPT support both GSS-API authentication and user ID/password authentication; however, GSS-API authentication is the preferred authentication type.

Note: Authentication types determine how and where a user is authenticated. To use a particular authentication type, update the authentication database manager configuration parameter.

Each of the plug-ins can be used independently or in conjunction with one or more of the other plug-ins. For example, you might only use a server authentication

plug-in and assume the DB2 defaults for client and group authentication. Alternatively, you might have only a group or client authentication plug-in. The only situation where both a client and server plug-in are required is for GSS-API authentication plug-ins.

The default behavior is to use a user ID/password plug-in that implements an operating-system-level mechanism for authentication. In previous releases, the default behavior is to directly use operating-system-level authentication without a plug-in implementation. Client-side Kerberos support is available on Solaris, AIX, Windows, and Linux operating systems. For Windows platforms, Kerberos support is enabled by default.

DB2 database systems include sets of plug-ins for group retrieval, user ID/password authentication, and for Kerberos authentication. With the security plug-in architecture you can customize DB2 client and server authentication behavior by either developing your own plug-ins, or buying plug-ins from a third party.

Deployment of security plug-ins on DB2 clients

DB2 clients can support one group plug-in, one user ID/password authentication plug-in, and will negotiate with the DB2 server for a particular GSS-API plug-in. This negotiation consists of a scan by the client of the DB2 server's list of implemented GSS-API plug-ins for the first authentication plug-in name that matches an authentication plug-in implemented on the client. The server's list of plug-ins is specified in the *srvcon_gssplugin_list* database manager configuration parameter value, which contains the names of all of the plug-ins that are implemented on the server. The following figure portrays the security plug-in infrastructure on a DB2 client.

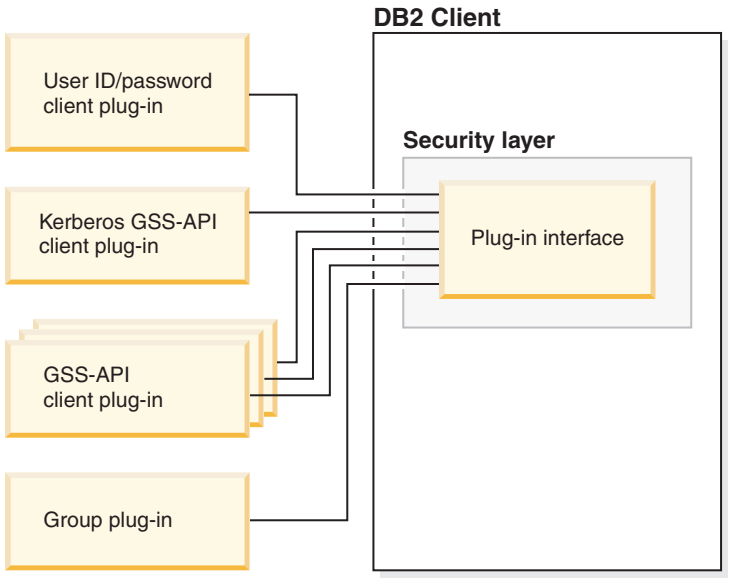


Figure 3. Deploying Security Plug-ins on DB2 Clients

Deployment of security plug-ins on DB2 servers

DB2 servers can support one group plug-in, one user ID/password authentication plug-in, and multiple GSS-API plug-ins. The multiple GSS-API plug-ins are

specified in the *srvcon_gssplugin_list* database manager configuration parameter value as a list. Only one GSS-API plug-in in this list can be a Kerberos plug-in.

In addition to server-side security plug-ins, you might also need to deploy client authorization plug-ins on your database server. When you run instance-level operations like db2start and db2trc, the DB2 database manager performs authorization checking for these operations using client authentication plug-ins. Therefore, you should install the client authentication plug-in that corresponds to the server plug-in that is specified by the *authentication* database manager configuration parameter. There is a main distinction between *authentication* and *srvcon_auth*. Specifically, they could be set to different values to cause one mechanism to be used to authenticate database connections and another mechanism to be used for local authorization. The most common usage is *srvcon_auth* set as GSSPLUGIN and *authentication* set as SERVER. If you do not use client authentication plug-ins on the database server, instance level operations such as db2start will fail. For example, if the authentication type is SERVER and no user-supplied client plug-in is used, the DB2 database system will use the IBM-shipped default client operating-system plug-in. The following figure portrays the security plug-in infrastructure on a DB2 server.

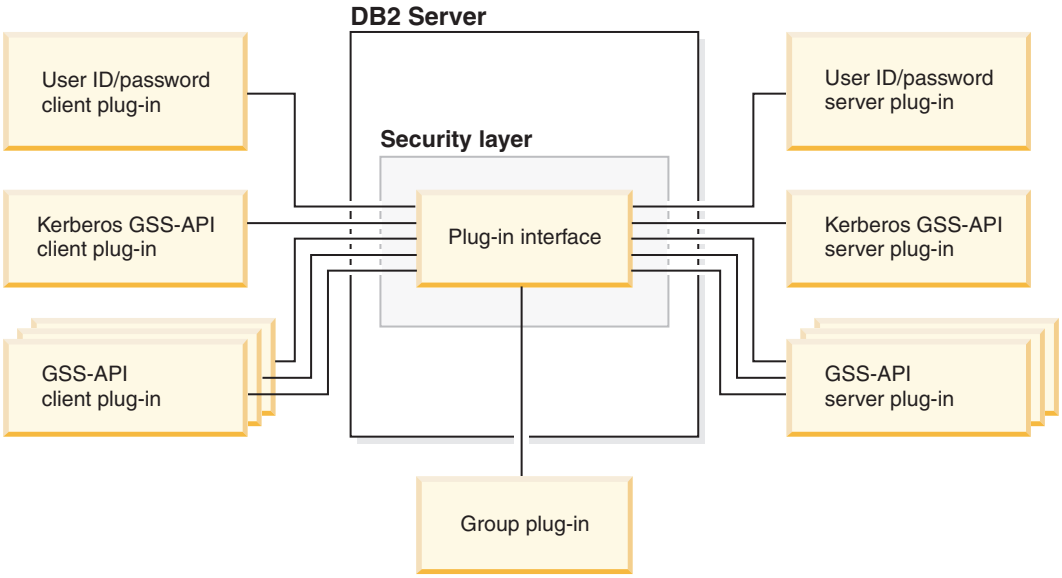


Figure 4. Deploying Security Plug-ins on DB2 Servers

Note: The integrity of your DB2 database system installation can be compromised if the deployment of security plug-ins are not adequately coded, reviewed, and tested. The DB2 database system takes precaution against many common types of failures, but it cannot guarantee complete integrity when user-written security plug-ins are deployed.

Enabling security plug-ins

The system administrator can specify the names of the plug-ins to use for each authentication mechanism by updating certain plug-in-related database manager configuration parameters. If these parameters are null, they will default to the DB2-supplied plug-ins for group retrieval, user ID/password management, or Kerberos (if authentication is set to Kerberos -- on the server). DB2 does not provide a default GSS-API plug-in. Therefore, if system administrators specify an authentication type of GSSPLUGIN in *authentication* parameter, they must also

specify a GSS-API authentication plug-in in *srvcon_gssplugin_list*.

How DB2 loads security plug-ins

All of the supported plug-ins identified by the database manager configuration parameters are loaded when the database manager starts.

The DB2 client will load a plug-in appropriate for the security mechanism negotiated with the server during connect or attach operations. It is possible that a client application can cause multiple security plug-ins to be concurrently loaded and used. This situation can occur, for example, in a threaded program that has concurrent connections to different databases from different instances.

Actions other than connect or attach operations require authorization (such as updating the database manager configuration, starting and stopping the database manager, turning DB2 trace on and off) as well. For such actions, the DB2 client program will load a plug-in specified in another database manager configuration parameter. If *authentication* is set to GSSPLUGIN, DB2 database manager will use the plug-in specified by *local_gssplugin*. If *authentication* is set to KERBEROS, DB2 database manager will use the plug-in specified by *clnt_krb_plugin*. Otherwise, DB2 database manager will use the plug-in specified by *clnt_pw_plugin*.

Security plug-ins APIs can be called from either an IPv4 platform or an IPv6 platform. An IPv4 address is a 32-bit address which has a readable form a.b.c.d, where each of a-d represents a decimal number from 0-255. An IPv6 address is a 128 bit address of the form a:b:c:d:e:f:g:h, where each of a-h represents 4 hex digits.

Developing security plug-ins

If you are developing a security plug-in, you need to implement the standard authentication functions that DB2 database manager will use. If you are using your own customized security plug-in, you can use a user ID of up to 255 characters on a connect statement issued through the CLP or a dynamic SQL statement. For the available types of plug-ins, the functionality you will need to implement is as follows:

Group retrieval
Gets the list of groups to which a user belongs.

User ID/password authentication

- Identifies the default security context (client only).
- Validates and optionally changes a password.
- Determines if a given string represents a valid user (server only).
- Modifies the user ID or password provided on the client before it is sent to the server (client only).
- Returns the DB2 authorization ID associated with a given user.

GSS-API authentication

- Implements the required GSS-API functions.
- Identifies the default security context (client only).
- Generates initial credentials based on a user ID and password and optionally changes password (client only).
- Creates and accepts security tickets.
- Returns the DB2 authorization ID associated with a given GSS-API security context.

Security plug-in library locations

After you acquire your security plug-ins (either by developing them yourself, or purchasing them from a third party), copy them to specific locations on your database server.

DB2 clients looks for client-side user authentication plug-ins in the following directory:

- UNIX 32-bit: `$DB2PATH/security32/plugin/client`
- UNIX 64-bit: `$DB2PATH/security64/plugin/client`
- WINDOWS 32-bit and 64-bit: `$DB2PATH\security\plugin\`*`instance name`*`\client`

Note: On Windows-based platforms, the subdirectories *instance name* and *client* are not created automatically. The instance owner has to manually create them.

The DB2 database manager looks for server-side user authentication plug-ins in the following directory:

- UNIX 32-bit: `$DB2PATH/security32/plugin/server`
- UNIX 64-bit: `$DB2PATH/security64/plugin/server`
- WINDOWS 32-bit and 64-bit: `$DB2PATH\security\plugin\`*`instance name`*`\server`

Note: On Windows-based platforms, the subdirectories *instance name* and *server* are not created automatically. The instance owner has to manually create them.

The DB2 database manager looks for group plug-ins in the following directory:

- UNIX 32-bit: `$DB2PATH/security32/plugin/group`
- UNIX 64-bit: `$DB2PATH/security64/plugin/group`
- WINDOWS 32-bit and 64-bit: `$DB2PATH\security\plugin\`*`instance name`*`\group`

Note: On Windows-based platforms, the subdirectories *instance name* and *group* are not created automatically. The instance owner has to manually create them.

Security plug-in naming conventions

Security plug-in libraries must have a platform-specific file name extension. Security plug-in libraries written in C or C++ must have a platform-specific file name extension:

- Windows: `.dll`
- AIX: `.a or .so`, and if both extensions exist, `.a` extension is used.
- Linux, HP IPF and Solaris: `.so`
- HPUX on PA-RISC: `.sl or .so`, and if both extensions exist, `.sl` extension is used.

Note: Users can also develop security plug-ins with the DB2 Universal JDBC Driver.
For example, assume you have a security plug-in library called `MyPlugin`. For each supported operating system, the appropriate library file name follows:

- Windows 32-bit: MyPlugin.dll
- Windows 64-bit: MyPlugin64.dll
- AIX 32 or 64-bit: MyPlugin.a or MyPlugin.so
- SUN 32 or 64-bit, Linux 32 or 64 bit, HP 32 or 64 bit on IPF: MyPlugin.so

- HP-UX 32 or 64-bit on PA-RISC: MyPlugin.sl or MyPlugin.so

Note: The suffix ″64″ is only required on the library name for 64-bit Windows security plug-ins.

When you update the database manager configuration with the name of a security plug-in, use the full name of the library without the ″64″ suffix and omit both the file extension and any qualified path portion of the name. Regardless of the operating system, a security plug-in library called `MyPlugin` would be registered as follows:

```
UPDATE DBM CFG USING CLNT_PW_PLUGIN MyPlugin
```

The security plug-in name is case sensitive, and must exactly match the library name. DB2 database systems use the value from the relevant database manager configuration parameter to assemble the library path, and then uses the library path to load the security plug-in library.

To avoid security plug-in name conflicts, you should name the plug-in using the authentication method used, and an identifying symbol of the firm that wrote the plug-in. For instance, if the company `Foo, Inc.` wrote a plug-in implementing the authentication method `FOOsomemethod`, the plug-in could have a name like `FOOsomemethod.dll`.

The maximum length of a plug-in name (not including the file extension and the ″64″ suffix) is limited to 32 bytes. There is no maximum number of plug-ins supported by the database server, but the maximum length of the comma-separated list of plug-ins in the database manager configuration is 255 bytes. Two defines located in the include file `sqlenv.h` identifies these two limits:

```
#define SQL_PLUGIN_NAME_SZ     32     /* plug-in name */
#define SQL_SRVCON_GSSPLUGIN_LIST_SZ 255 /* GSS API plug-in list */
```

The security plug-in library files must have the following file permissions:

- Owned by the instance owner.
- Readable by all users on the system.
- Executable by all users on the system.

Security plug-in support for two-part user IDs

The DB2 database manager on Windows supports the use of two-part user IDs, and the mapping of two-part user IDs to two-part authorization IDs.

For example, consider a Windows operating system two-part user ID composed of a domain and user ID such as: `MEDWAY\pieter`. In this example, `MEDWAY` is a domain and `pieter` is the user name. In DB2 database systems, you can specify whether this two-part user ID should be mapped to either a one-part authorization ID or a two-part authorization ID.

The mapping of a two-part user ID to a two-part authorization ID is supported, but is not the default behavior. By default, both one-part user IDs and two-part user IDs map to one-part authorization IDs. The mapping of a two-part user ID to a two-part authorization ID is supported, but is not the default behavior.

The default mapping of a two-part user ID to a one-part user ID allows a user to connect to the database using:

```
db2 connect to db user MEDWAY\pieter using pw
```

In this situation, if the default behavior is used, the user ID MEDWAY\pieter is resolved to the authorization ID PIETER. If the support for mapping a two-part user ID to a two-part authorization ID is enabled, the authorization ID would be MEDWAY\PIETER.

To enable DB2 to map two-part user IDs to two-part authorization IDs, DB2 supplies two sets of authentication plug-ins:

- One set exclusively maps a one-part user ID to a one-part authorization ID and maps a two-part user-ID to a one-part authorization ID.
- Another set maps both one-part user ID or two-part user ID to a two-part authorization ID.

If a user name in your work environment can be mapped to multiple accounts defined in different locations (such as local account, domain account, and trusted domain accounts), you can specify the plug-ins that enable two-part authorization ID mapping.

It is important to note that a one-part authorization ID, such as, PIETER and a two-part authorization ID that combines a domain and a user ID like MEDWAY\pieter are functionally distinct authorization IDs. The set of privileges associated with one of these authorization IDs can be completely distinct from the set of privileges associated with the other authorization ID. Care should be taken when working with one-part and two-part authorization IDs.

The following table identifies the kinds of plug-ins supplied by DB2 database systems, and the plug-in names for the specific authentication implementations.

Table 30. DB2 security plug-ins

Authentication type	Name of one-part user ID plug-in	Name of two-part user ID plug-in
User ID/password (client)	IBMOSauthclient	IBMOSauthclientTwoPart
User ID/password (server)	IBMOSauthserver	IBMOSauthserverTwoPart
Kerberos	IBMkrb5	IBMkrb5TwoPart

Note: On Windows 64-bit platforms, the characters ″64″ are appended to the plug-in names listed here.

When you specify an authentication type that requires a user ID/password or Kerberos plug-in, the plug-ins that are listed in the ″Name of one-part user ID plug-in″ column in the previous table are used by default.

To map a two-part user ID to a two-part authorization ID, you must specify that the two-part plug-in, which is not the default plug-in, be used. Security plug-ins are specified at the instance level by setting the security related database manager configuration parameters as follows:

For server authentication that maps two-part user IDs to two-part authorization IDs, you must set:

- srvcon_pw_plugin to IBMOSauthserverTwoPart
- clnt_pw_plugin to IBMOSauthclientTwoPart

For client authentication that maps two-part user IDs to two-part authorization IDs, you must set:

- `srvcon_pw_plugin` to `IBMOSauthserverTwoPart`
- `clnt_pw_plugin` to `IBMOSauthclientTwoPart`

For Kerberos authentication that maps two-part user IDs to two-part authorization IDs, you must set:

- `srvcon_gssplugin_list` to `IBMOSkrb5TwoPart`
- `clnt_krb_plugin` to `IBMkrb5TwoPart`

The security plug-in libraries accept two-part user IDs specified in a Microsoft Windows Security Account Manager compatible format. For example, in the format: *domain\user ID*. Both the domain and user ID information will be used by the DB2 authentication and authorization processes at connection time.

You should consider implementing the two-part plug-ins when creating new databases to avoid conflicts with one-part authorization IDs in existing databases. New databases that use two-part authorization IDs must be created in a separate instance from databases that use single-part authorization IDs.

Security plug-in API versioning

The DB2 database system supports version numbering of the security plug-in APIs. These version numbers are integers starting with 1 for DB2 UDB, Version 8.2.

The version number that DB2 passes to the security plug-in APIs is the highest version number of the API that DB2 can support, and corresponds to the version number of the structure. If the plug-in can support a higher API version, it must return function pointers for the version that DB2 has requested. If the plug-in only supports a lower version of the API, the plug-in should fill in function pointers for the lower version. In either situation, the security plug-in APIs should return the version number for the API it is supporting in the version field of the functions structure.

For DB2, the version numbers of the security plug-ins will only change when necessary (for example, when there are changes to the parameters of the APIs). Version numbers will not automatically change with DB2 release numbers.

32-bit and 64-bit considerations for security plug-ins

In general, a 32-bit DB2 instance uses the 32-bit security plug-in and a 64-bit DB2 instance uses the 64-bit security plug-in. However, on a 64-bit instance, DB2 supports 32-bit applications, which require the 32-bit plug-in library.

A database instance where both the 32-bit and the 64-bit applications can run is known as a hybrid instance. If you have a hybrid instance and intend to run 32-bit applications, ensure that the required 32-bit security plug-ins are available in the 32-bit plug-in directory. For 64-bit DB2 instances on Linux and UNIX operating systems, excluding Linux on IPF, the directories `security32` and `security64` appear. For a 64-bit DB2 instance on Windows on X64 or IPF, both 32-bit and 64-bit security plug-ins are located in the same directory, but 64-bit plug-in names have a suffix, ″64″.

If you want to migrate from a 32-bit instance to a 64-bit instance, you should obtain versions of your security plug-ins that are recompiled for 64-bit.

If you acquired your security plug-ins from a vendor that does not supply 64-bit plug-in libraries, you can implement a 64-bit stub that executes a 32-bit application. In this situation, the security plug-in is an external program rather than a library.

Security plug-in problem determination

Problems with security plug-ins are reported in two ways: through SQL errors and through the administration notification log.

Following are the SQLCODE values related to security plug-ins:

- SQLCODE -1365 is returned when a plug-in error occurs during db2start or db2stop.
- SQLCODE -1366 is returned whenever there is a local authorization problem.
- SQLCODE -30082 is returned for all connection-related plug-in errors.

The administration notification log is a good resource for debugging and administrating security plug-ins. To see the administration notification log on UNIX, check `sqllib/db2dump/`*`instance name`*`.nfy`. To see the administration notification log on Windows operating systems, use the Event Viewer tool. The Event Viewer tool can be found by navigating from the Windows operating system "Start" button to `Settings -> Control Panel -> Administrative Tools -> Event Viewer`. Following are the administration notification log values related to security plug-ins:

- 13000 indicates that a call to a GSS-API security plug-in API failed with an error, and returned an optional error message.

  ```
  SQLT_ADMIN_GSS_API_ERROR (13000)
  Plug-in "plug-in name" received error code "error code" from
  GSS API  "gss api name" with the error message "error message"
  ```

- 13001 indicates that a call to a DB2 security plug-in API failed with an error, and returned an optional error message.

  ```
  SQLT_ADMIN_PLUGIN_API_ERROR(13001)
  Plug-in "plug-in name" received error code "error code" from DB2
  security plug-in API "gss api name" with the error message
  "error message"
  ```

- 13002 indicates that DB2 failed to unload a plug-in.

  ```
  SQLT_ADMIN_PLUGIN_UNLOAD_ERROR (13002)
  Unable to unload plug-in "plug-in name". No further action required.
  ```

- 13003 indicates a bad principal name.

  ```
  SQLT_ADMIN_INVALID_PRIN_NAME (13003)
  The principal name "principal name" used for "plug-in name"
  is invalid. Fix the principal name.
  ```

- 13004 indicates that the plug-in name is not valid. Path separators (On UNIX "/" and on Windows "\") are not allowed in the plug-in name.

  ```
  SQLT_ADMIN_INVALID_PLGN_NAME (13004)
  The plug-in name "plug-in name" is invalid. Fix the plug-in name.
  ```

- 13005 indicates that the security plug-in failed to load. Ensure the plug-in is in the correct directory and that the appropriate database manager configuration parameters are updated.

  ```
  SQLT_ADMIN_PLUGIN_LOAD_ERROR (13005)
  Unable to load plug-in "plug-in name". Verify the plug-in existence and
  directory where it is located is correct.
  ```

- 13006 indicates that an unexpected error was encountered by a security plug-in. Gather all the db2support information, if possible capture a db2trc, and then call IBM support for further assistance.

```
SQLT_ADMIN_PLUGIN_UNEXP_ERROR (13006)
Plug-in encountered unexpected error. Contact IBM Support for further assistance.
```

Note: If you are using security plug-ins on a Windows 64-bit database server and are seeing a load error for a security plug-in, see the topics about 32-bit and 64-bit considerations and security plug-in naming conventions. The 64-bit plug-in library requires the suffix ″64″ on the library name, but the entry in the security plug-in database manager configuration parameters should not indicate this suffix.

Enabling plug-ins

Deploying a group retrieval plug-in

To customize the DB2 security system's group retrieval behavior, you can develop your own group retrieval plug-in or buy one from a third party.

After you acquire a group retrieval plug-in that is suitable for your database management system, you can deploy it.

- To deploy a group retrieval plug-in on the database server, perform the following steps:
  1. Copy the group retrieval plug-in library into the server's group plug-in directory.
  2. Update the database manager configuration parameter *group_plugin* with the name of the plug-in.
- To deploy a group retrieval plug-in on database clients, perform the following steps:
  1. Copy the group retrieval plug-in library in the client's group plug-in directory.
  2. On the database client, update the database manager configuration parameter *group_plugin* with the name of the plug-in.

Deploying a user ID/password plug-in

To customize the DB2 security system's user ID/password authentication behavior, you can develop your own user ID/password authentication plug-ins or buy one from a third party.

Depending on their intended usage, all user ID-password based authentication plug-ins must be placed in either the client plug-in directory or the server plug-in directory. If a plug-in is placed in the client plug-in directory, it will be used both for local authorization checking and for validating the client when it attempts to connect with the server. If the plug-in is placed in the server plug-in directory, it will be used for handling incoming connections to the server and for checking whether an authorization ID exists and is valid whenever the GRANT statement is issued without specifying either the keyword USER or GROUP. In most situations, user ID/password authentication requires only a server-side plug-in. It is possible, though generally deemed less useful, to have only a client user ID/password plug-in. It is possible, though quite unusual to require matching user ID/password plug-ins on both the client and the server.

Note: You must stop the DB2 server or any applications using the plug-ins before you deploy a *new* version of an *existing* plug-in. Undefined behavior including traps will occur if a process is still using a plug-in when a new version (with the same name) is copied over it. This restriction is not in effect when you deploy a plugin for the first time or when the plug-in is not in use.

After you acquire user ID/password authentication plug-ins that are suitable for your database management system, you can deploy them.

- To deploy a user ID/password authentication plug-in on the database server, perform the following steps on the database server:
  1. Copy the user ID/password authentication plug-in library in the server plug-in directory.
  2. Update the database manager configuration parameter *srvcon_pw_plugin* with the name of the server plug-in. This plug-in is used by the server when it is handling CONNECT and ATTACH requests.
  3. Either:
     - Set the database manager configuration parameter *srvcon_auth* to the CLIENT, SERVER, SERVER_ENCRYPT, DATA_ENCRYPT, or DATA_ENCRYPT_CMP authentication type. Or:
     - Set the database manager configuration parameter *srvcon_auth* to NOT_SPECIFIED and set *authentication* to CLIENT, SERVER, SERVER_ENCRYPT, DATA_ENCRYPT, or DATA_ENCRYPT_CMP authentication type.
- To deploy a user ID/password authentication plug-in on database clients, perform the following steps on each client:
  1. Copy the user ID/password authentication plug-in library in the client plug-in directory.
  2. Update the database manager configuration parameter *clnt_pw_plugin* with the name of the client plug-in. This plug-in is loaded and called regardless of where the authentication is being done, not only when the database configuration parameter, *authentication* is set to CLIENT.
- For local authorization on a client, server, or gateway using a user ID/password authentication plug-in, perform the following steps on each client, server, or gateway:
  1. Copy the user ID/password authentication plug-in library in the client plug-in directory on the client, server, or gateway.
  2. Update the database manager configuration parameter *clnt_pw_plugin* with the name of the plug-in.
  3. Set the *authentication* database manager configuration parameter to CLIENT, SERVER, SERVER_ENCRYPT, DATA_ENCRYPT, or DATA_ENCRYPT_CMP.

Deploying a GSS-API plug-in

To customize the DB2 security system's authentication behavior, you can develop your own authentication plug-ins using the GSS-API, or buy one from a third party.

In the case of plug-in types other than Kerberos, you must have matching plug-in names on the client and the server along with the same plug-in type. The plug-ins on the client and server need not be from the same vendor, but they must generate and consume compatible GSS-API tokens. Any combination of Kerberos plug-ins deployed on the client and the server is acceptable since Kerberos plug-ins are standardized. However, different implementations of less standardized GSS-API mechanisms, such as *x.509* certificates, might only be partially compatible with DB2 database systems. Depending on their intended usage, all GSS-API authentication plug-ins must be placed in either the client plug-in directory or the server plug-in directory. If a plug-in is placed in the client plug-in directory, it will be used for local authorization checking and when a client attempts to connect with the server. If the plug-in is placed in the server plug-in directory, it will be

used for handling incoming connections to the server and for checking whether an authorization ID exists and is valid whenever the GRANT statement is issued without specifying either the keyword USER or GROUP.

Note: You must stop the DB2 server or any applications using the plug-ins before you deploy a *new* version of an *existing* plug-in. Undefined behavior including traps will occur if a process is still using a plug-in when a new version (with the same name) is copied over it. This restriction is not in effect when you deploy a plugin for the first time or when the plug-in is not in use.

After you acquire GSS-API authentication plug-ins that are suitable for your database management system, you can deploy them.

- To deploy a GSS-API authentication plug-in on the database server, perform the following steps on the server:
  1. Copy the GSS-API authentication plug-in library in the server plug-in directory. You can copy numerous GSS-API plug-ins into this directory.
  2. Update the database manager configuration parameter *srvcon_gssplugin_list* with an ordered, comma-delimited list of the names of the plug-ins installed in the GSS-API plug-in directory.
  3. Either:
     - Setting the database manager configuration parameter *srvcon_auth* to GSSPLUGIN or GSS_SERVER_ENCRYPT is a way to enable the server to use GSSAPI PLUGIN authentication method. Or:
     - Setting the database manager configuration parameter *srvcon_auth* to NOT_SPECIFIED and setting *authentication* to GSSPLUGIN or GSS_SERVER_ENCRYPT is a way to enable the server to use GSSAPI PLUGIN authentication method.
- To deploy a GSS-API authentication plug-in on database clients, perform the following steps on each client:
  1. Copy the GSS-API authentication plug-in library in the client plug-in directory. You can copy numerous GSS-API plug-ins into this directory. The client selects a GSS-API plug-in for authentication during a CONNECT or ATTACH operation by picking the first GSS-API plug-in contained in the server's plug-in list that is available on the client.
  2. Optional: Catalog the databases that the client will access, indicating that the client will only accept a GSS-API authentication plug-in as the authentication mechanism. For example:

     ```
     CATALOG DB testdb AT NODE testnode AUTHENTICATION GSSPLUGIN
     ```
- For local authorization on a client, server, or gateway using a GSS-API authentication plug-in, perform the following steps:
  1. Copy the GSS-API authentication plug-in library in the client plug-in directory on the client, server, or gateway.
  2. Update the database manager configuration parameter *local_gssplugin* with the name of the plug-in.
  3. Set the *authentication* database manager configuration parameter to GSSPLUGIN, or GSS_SERVER_ENCRYPT.

Deploying a Kerberos plug-in

To customize the DB2 security system's Kerberos authentication behavior, you can develop your own Kerberos authentication plug-ins or buy one from a third party. Note that the Kerberos security plug-in will not support IPv6.

Note: You must stop the DB2 server or any applications using the plug-ins before you deploy a *new* version of an *existing* plug-in. Undefined behavior including traps will occur if a process is still using a plug-in when a new version (with the same name) is copied over it. This restriction is not in effect when you deploy a plugin for the first time or when the plug-in is not in use.

After you acquire Kerberos authentication plug-ins that are suitable for your database management system, you can deploy them.

- To deploy a Kerberos authentication plug-in on the database server, perform the following steps on the server:
  1. Copy the Kerberos authentication plug-in library in the server plug-in directory.
  2. Update the database manager configuration parameter *srvcon_gssplugin_list*, which is presented as an ordered, comma delimited list, to include the Kerberos server plug-in name. Only one plug-in in this list can be a Kerberos plug-in. If this list is blank and *authentication* is set to KERBEROS or KRB_SVR_ENCRYPT, the default DB2 Kerberos plug-in: IBMkrb5 will be used.
  3. Either:
     - Set the database manager configuration parameter *srvcon_auth* to the KERBEROS or KRB_SERVER_ENCRYPT authentication type. (You can deploy a KERBEROS plugin and still use GSSPLUGIN or GSS_SERVER_ENCRYPT) Or:
     - Set the database manager configuration parameter *srvcon_auth* to NOT_SPECIFIED and set *authentication* to KERBEROS or KRB_SERVER_ENCRYPT authentication type.
- To deploy a Kerberos authentication plug-in on database clients, perform the following steps on each client:
  1. Copy the Kerberos authentication plug-in library in the client plug-in directory.
  2. Update the database manager configuration parameter *clnt_krb_plugin* with the name of the Kerberos plug-in. If *clnt_krb_plugin* is blank, DB2 assumes that the client cannot use Kerberos authentication. This setting is only appropriate when the server cannot support plug-ins. If both the server and the client support security plug-ins, the default server plug-in, *IBMkrb5* would be used over the client value of *clnt_krb_plugin*. For local authorization on a client, server, or gateway using a Kerberos authentication plug-in, perform the following steps:
     a. Copy the Kerberos authentication plug-in library in the client plug-in directory on the client, server, or gateway.
     b. Update the database manager configuration parameter *clnt_krb_plugin* with the name of the plug-in.
     c. Set the *authentication* database manager configuration parameter to KERBEROS, or KRB_SERVER_ENCRYPT.
  3. Optional: Catalog the databases that the client will access, indicating that the client will only use a Kerberos authentication plug-in. For example:

```
CATALOG DB testdb AT NODE testnode AUTHENTICATION KERBEROS
        TARGET PRINCIPAL service/host@REALM
```

Note: For platforms supporting Kerberos, the `IBMkrb5` library will be present in the client plug-in directory. DB2 will recognize this library as a valid GSS-API plug-in, because Kerberos plug-ins are implemented using GSS-API plug-in.

LDAP-based authentication and group lookup support

The DB2 database manager and DB2 Connect support LDAP-based authentication and group lookup functionality through the use of LDAP security plug-in modules.

LDAP security plug-in modules allow the DB2 database manager to authenticate users defined in an LDAP directory, removing the requirement that users and groups be defined to the operating system. Supported platforms are AIX, Linux on IA32, Linux on x64, Linux on zSeries, Solaris, and Windows. Compiled binary plug-in modules for these supported platforms are found under the appropriate directory (for example, aix64, win32, and so on).

Supported LDAP servers for use with security plug-in modules are:

- IBM Tivoli® Directory Server (ITDS) Version 5.2, 6.0, and later
- Microsoft Active Directory (MSAD) Version 2000, 2003, and later
- Sun Java System Directory Server Enterprise Edition, Version 5.2, and later
- Novell eDirectory, Version 8.7, and later
- IBM Lotus® Domino® LDAP Server, Version 7.0, and later
- z/OS Integrated Security Services LDAP Server Version V1R6, and later

Note: When you use the LDAP plugin modules, all users associated with the database must be defined on the LDAP server. This includes both the DB2 instance owner ID as well as the fenced user. (These users are typically defined in the operating system, but must also be defined in LDAP.) Similarly, if you use the LDAP group plug-in module, any groups required for authorization must be defined on the LDAP server. This includes the SYSADM, SYSMAINT, SYSCTRL and SYSMON groups defined in the database manager configuration.

DB2 security plug-in modules are available for server-side authentication, client-side authentication and group lookup, described later. Depending on your specific environment, you may need to use one, two or all three types of plug-in.

To use DB2 security plug-in modules, follow these steps:

1. Decide if you need server, client, or group plug-in modules, or a combination of these modules.
2. Configure the plug-in modules by setting values in the IBM LDAP security plug-in configuration file (default name is IBMLDAPSecurity.ini). You will need to consult with your LDAP administrator to determine appropriate values.
3. Enable the plug-in modules
4. Test connecting with various LDAP User IDs.

Server authentication plugin

The server authentication plug-in module performs server validation of user IDs and passwords supplied by clients on CONNECT and ATTACH statements. It also provides a way to map LDAP user IDs to DB2 authorization IDs, if required. The server plug-in module is generally required if you want users to authenticate to the DB2 database manager using their LDAP user ID and password.

Client authentication plug-in

The client authentication plug-in module is used where user ID and password validation occurs on the client system; that is, where the DB2 server is configured

with SRVCON_AUTH or AUTHENTICATION settings of CLIENT. The client validates any user IDs and passwords supplied on CONNECT or ATTACH statements, and sends the user ID to the DB2 server. Note that CLIENT authentication is difficult to secure, and not generally recommended.

The client authentication plug-in module may also be required if the local operating system user IDs on the database server are different from the DB2 authorization IDs associated with those users. You can use the client-side plugin to map local operating system user IDs to DB2 authorization IDs prior to performing authorization checks for local commands on the database server, such as for:`db2start`.

Group lookup plug-in

The group lookup plug-in module retrieves group membership information from the LDAP server for a particular user. It is required if you want to use LDAP to store your group definitions. The most common scenario is where:

- All users and groups are defined in the LDAP server
- Any users defined locally on the database server are also defined with the same user ID on the LDAP server (including the instance owner and the fenced user)
- Password validation occurs on the DB2 server (that is, an AUTHENTICATION or SRVCON_AUTH value of SERVER, SERVER_ENCRYPT or DATA_ENCRYPT is set in the server DBM config file).

It is generally sufficient to install only the server authentication plug-in module and the group lookup plug-in module on the server. DB2 clients typically do not need to have the LDAP plug-in module installed.

It is possible to use only the LDAP group lookup plug-in module in combination with some other form of authentication plug-in (such as Kerberos). In this case, the LDAP group lookup plug-in module will be provided the DB2 authorization IDs associated with a user. The plug-in module searches the LDAP directory for a user with a matching AUTHID_ATTRIBUTE, then retrieves the groups associated with that user object.

Configuring the LDAP plug-in modules

To configure the LDAP plug-in modules, you need to update your IBM LDAP security plug-in configuration file to suit your environment. In most cases, you will need to consult with your LDAP administrator to determine the appropriate configuration values.

The default name and location for the IBM LDAP security plug-in configuration file is:

- On UNIX: INSTHOME/sqllib/cfg/IBMLDAPSecurity.ini
- On Windows: %DB2PATH%\cfg\IBMLDAPSecurity.ini

Optionally, you can specify the location of this file using the DB2LDAPSecurityConfig environment variable. On Windows, you should set DB2LDAPSecurityConfig in the global system environment, to ensure it is picked up by the DB2 service.

The following tables provide information to help you determine appropriate configuration values.

Table 31. Server-related values

Parameter	Description
LDAP_HOST	The name of your LDAP server(s). This is a space separated list of LDAP server host names or IP addresses, with an optional port number for each one. For example: `host1[:port] [host2:[port2] ... ]` The default port number is 389, or 636 if SSL is enabled.
ENABLE_SSL	To enable SSL support, set ENABLE_SSL to TRUE (you must have the GSKit installed). This is an optional parameter; it defaults to FALSE (no SSL support).
SSL_KEYFILE	The path for the SSL keyring. A keyfile is only required if your LDAP server is using a certificate that is not automatically trusted by your GSKit installation. For example:`SSL_KEYFILE = /home/db2inst1/IBMLDAPSecurity.kdb`
SSL_PW	The SSL keyring password. For example: `SSL_PW = keyfile-password`

Table 32. User-related values

Parameter	Description
USER_ OBJECTCLASS	The LDAP object class used for users. Generally, set USER_OBJECTCLASS to `inetOrgPerson` (the user for Microsoft Active Directory) For example: `USER_OBJECTCLASS = inetOrgPerson`
USER_BASEDN	The LDAP base DN to use when searching for users. If not specified, user searches start at the root of the LDAP directory. Some LDAP servers require that you specify a value for this parameter. For example: `USER_BASEDN = o=ibm`

USERID_ ATTRIBUTE	The LDAP user attribute that represents the user ID.
	The USERID_ATTRIBUTE attribute is combined with the USER_OBJECTCLASS and USER_BASEDN (if specified) to construct an LDAP search filter when a user issues a DB2 CONNECT statement with an unqualified user ID.
	For example, if `USERID_ATTRIBUTE = uid`, then issuing this statement:
	`db2 connect to MYDB user bob using bobpass`
	results in the following search filter:
	`&(objectClass=inetOrgPerson)(uid=bob)`

AUTHID_ ATTRIBUTE	The LDAP user attribute that represents the DB2 authorization ID. Usually this is the same as the USERID_ATTRIBUTE. For example: `AUTHID_ATTRIBUTE = uid`

Table 33. Group-related values

Parameter	Description
GROUP_ OBJECTCLASS	The LDAP object class used for groups. Generally this is `groupOfNames` or `groupOfUniqueNames` (for Microsoft Active Directory, it is `group`) For example: `GROUP_OBJECTCLASS = groupOfNames`
GROUP_BASEDN	The LDAP base DN to use when searching for groups If not specified, group searches start at the root of the LDAP directory. Some LDAP servers require that you specify a value for this parameter. For example: `GROUP_BASEDN = o=ibm`

Table 33. Group-related values (continued)

Parameter	Description
GROUPNAME_ ATTRIBUTE	The LDAP group attribute that represents the name of the group. For example: `GROUPNAME_ATTRIBUTE = cn`
GROUP_LOOKUP_ METHOD	Determines the method used to find the group memberships for a user. Possible values are: • SEARCH_BY_DN Indicates to search for groups that list the user as a member. Membership is indicated by the group attribute defined as GROUP_LOOKUP_ATTRIBUTE (typically, `member` or `uniqueMember`). • USER_ATTRIBUTE In this case, a user's groups are listed as attributes of the user object itself. This setting indicates to search for the user attribute defined as GROUP_LOOKUP_ATTRIBUTE to get the user's groups (typically `memberOf` for Microsoft Active Directory or `ibm-allGroups` for IBM Tivoli Directory Server). For example:`GROUP_LOOKUP_METHOD = SEARCH_BY_DN` `GROUP_LOOKUP_METHOD = USER_ATTRIBUTE`

GROUP_LOOKUP_ ATTRIBUTE	Name of the attribute used to determine group membership, as described for GROUP_LOOKUP_METHOD.
	
	For example:
	`GROUP_LOOKUP_ATTRIBUTE = member`
	`GROUP_LOOKUP_ATTRIBUTE = ibm-allGroups`

NESTED_GROUPS	If NESTED_GROUPS is TRUE, the DB2 database manager recursively searches for group membership by attempting to look up the group memberships for every group that is found. Cycles (such as A belongs to B, and B belongs to A) are handled correctly. This parameter is optional, and defaults to FALSE.

Table 34. Miscellaneous values

Parameter	Description

SEARCH_DN, SEARCH_PW	If your LDAP server does not support anonymous access, or if anonymous access is not sufficient when searching for users or groups, then you can optionally define a DN and password that will be used to perform searches.
	
	For example:
	`SEARCH_DN = cn=root`
	`SEARCH_PW = rootpassword`

DEBUG	Set DEBUG to TRUE to write extra information to db2diag.log to aid in debugging LDAP related issues. Most of the additional information is logged at DIAGLEVEL 4 (INFO). DEBUG defaults to false.

Enabling the LDAP plug-in modules

The following tables show where the LDAP plug-in modules are located on your DB2 instance.

Table 35. For 64-bit UNIX and Linux systems

Plug-in module type	Location
server	/sqllib/security64/plugin/IBM/server
client	/sqllib/security64/plugin/IBM/client
group	/sqllib/security64/plugin/IBM/group

Table 36. For 32-bit UNIX and Linux systems

Plug-in module type	Location
server	/sqllib/security32/plugin/IBM/server
client	/sqllib/security32/plugin/IBM/client
group	/sqllib/security32/plugin/IBM/group

Table 37. For Windows systems (both 64-bit and 32-bit)

Plug-in module type	Location
server	%DB2PATH%\security\plugin\IBM*instance-name*\server
client	%DB2PATH%\security\plugin\IBM*instance-name*\client
group	%DB2PATH%\security\plugin\IBM*instance-name*\group

Note: 64-bit Windows plug-in modules include the digits 64 in the file name.

Use the DB2 command line processor to update the database manager configuration to enable the plug-in modules that you require:

- For the server plug-in module:
  ```
  UPDATE DBM CFG USING SRVCON_PW_PLUGIN IBMLDAPauthserver
  ```
- For the client plug-in module:
  ```
  UPDATE DBM CFG USING CLNT_PW_PLUGIN IBMLDAPauthclient
  ```
- For the group plug-in module:
  ```
  UPDATE DBM CFG USING GROUP_PLUGIN IBMLDAPgroups
  ```

Terminate all running DB2 command line processor backend processes, by using the db2 terminate command, and then stop and restart the instance by using the db2stop and db2start commands.

Connecting with an LDAP user ID

The location of an object within an LDAP directory is defined by its distinguished name (DN).

A DN is typically a multi-part name that reflects some sort of hierarchy, for example:

```
cn=John Smith, ou=Sales, o=WidgetCorp
```

After an LDAP plug-in module has been enabled and configured, a user can connect to a DB2 database using a variety of different strings:

- A full DN. For example:
  ```
  connect to MYDB user 'cn=John Smith, ou=Sales, o=WidgetCorp'
  ```

- A partial DN, provided that a search of the LDAP directory using the partial DN and the appropriate search base DN (if defined) results in exactly one match. For example:

  ```
  connect to MYDB user 'cn=John Smith' connect to MYDB user uid=jsmith
  ```

- A simple string (containing no equals signs). The string is qualified with the USERID_ATTRIBUTE and treated as a partial DN. For example:

  ```
  connect to MYDB user jsmith
  ```

Note: Any string supplied on a CONNECT or ATTACH statement must be delimited with single quotes if it contains spaces or special characters.

User IDs and DB2 authorization IDs

A user's *user ID* is defined by an attribute associated with the user object (typically the uid attribute). It may be a simple string (such as jsmith), or look like an email address (such as jsmith@sales.widgetcorp.com), that reflects part of the organizational hierarchy.

A user's DB2 *authorization ID* is the name associated with that user within the DB2 database.

In the past, users were typically defined in the server's host operating system, and the user ID and authorization ID were the same (though the authorization ID is usually in uppercase). The DB2 LDAP plug-in modules give you the ability to associate different attributes of the LDAP user object with the user ID and the authorization ID. In most cases, the user ID and authorization ID can be the same string, and you can use the same attribute name for both the USERID_ATTRIBUTE and the AUTHID_ATTRIBUTE. However, if in your environment the user ID attribute typically contains extra information that you do not want to carry over to the authorization ID, you can configure a different AUTHID_ATTRIBUTE in the plug-in initialization file. The value of the AUTHID_ATTRIBUTE attribute is retrieved from the server and used as the internal DB2 representation of the user.

For example, if your LDAP user IDs look like email addresses (such as jsmith@sales.widgetcorp.com), but you would rather use just the user portion (jsmith) as the DB2 authorization ID, then you can:

1. Associate a new attribute containing the shorter name with all user objects on your LDAP server
2. Configure the AUTHID_ATTRIBUTE with the name of this new attribute

Users are then able to connect to a DB2 database by specifying their full LDAP user ID and password, for example:

```
db2 connect to MYDB user 'jsmith@sales.widgetcorp.com' using 'pswd'
```

But internally, the DB2 database manager refers to the user using the short name retrieved using the AUTHID_ATTRIBUTE (jsmith in this case).

Considerations for group lookup

Group membership information is typically represented on an LDAP server either as an attribute of the user object, or as an attribute of the group object:

- As an attribute of the user object

  Each user object has an attribute called GROUP_LOOKUP_ATTRIBUTE that you can query to retrieve all of the group membership for that user.

- As an attribute of the group object

Each group object has an attribute, also called GROUP_LOOKUP_ATTRIBUTE, that you can use to list all the user objects that are members of the group. You can enumerate the groups for a particular user by searching for all groups that list the user object as a member.

Many LDAP servers can be configured in either of these ways, and some support both methods at the same time. Consult with your LDAP administrator to determine how your LDAP server is configured.

When configuring the LDAP plug-in modules, you can use the GROUP_LOOKUP_METHOD parameter to specify how group lookup should be performed:

- If you need to use the GROUP_LOOKUP_ATTRIBUTE attribute of the user object to find group membership, set GROUP_LOOKUP_METHOD = USER_ATTRIBUTE
- If you need to use the GROUP_LOOKUP_ATTRIBUTE attribute of the group object to find group membership, set GROUP_LOOKUP_METHOD = SEARCH_BY_DN

Many LDAP servers use the GROUP_LOOKUP_ATTRIBUTE attribute of the group object to determine membership. They can be configured as shown in this example:

```
GROUP_LOOKUP_METHOD = SEARCH_BY_DN
GROUP_LOOKUP_ATTRIBUTE = groupOfNames
```

Microsoft Active Directory typically stores group membership as a user attribute, and could be configured as shown in this example:

```
GROUP_LOOKUP_METHOD = USER_ATTRIBUTE
GROUP_LOOKUP_ATTRIBUTE = memberOf
```

The IBM Tivoli Directory Server supports both methods at the same time. To query the group membership for a user you can make use of the special user attribute ibm-allGroups, as shown in this example:

```
GROUP_LOOKUP_METHOD = USER_ATTRIBUTE
GROUP_LOOKUP_ATTRIBUTE = ibm-allGroups
```

Other LDAP servers may offer similar special attributes to aid in retrieving group membership. In general, retrieving membership through a user attribute is faster than searching for groups that list the user as a member.

Troubleshooting authenticating LDAP users or retrieving groups

If you encounter problems authenticating LDAP users or retrieving their groups, the DB2 diagnostic log, db2diag.log, and administration log are a good source of information to aid in troubleshooting.

The LDAP plug-in modules typically log LDAP return codes, search filters, and other useful data when a failure occurs. If you enable the DEBUG option in the LDAP plug-in configuration file, the plug-in modules will log even more information in db2diag.log. While this may be an aid in troubleshooting, it is not recommended for extended use on production systems due to the overhead associated with writing all of the extra data to a single file.

Ensure that the DIAGLEVEL configuration parameter in the database manager is set to 4 so that all messages from the LDAP plug-in modules will be captured.

Writing security plug-ins

How DB2 loads security plug-ins

Each plug-in library must contain an initialization function with a specific name determined by the plug-in type:

- Server side authentication plug-in: `db2secServerAuthPluginInit()`
- Client side authentication plug-in: `db2secClientAuthPluginInit()`
- Group plug-in: `db2secGroupPluginInit()`

This function is known as the plug-in initialization function. The plug-in initialization function initializes the specified plug-in and provides DB2 with information that it requires to call the plug-in's functions. The plug-in initialization function accepts the following parameters:

- The highest version number of the function pointer structure that the DB2 instance invoking the plugin can support
- A pointer to a structure containing pointers to all the APIs requiring implementation
- A pointer to a function that adds log messages to the db2diag.log file
- A pointer to an error message string
- The length of the error message

The following is a function signature for the initialization function of a group retrieval plug-in:

```
SQL_API_RC SQL_API_FN db2secGroupPluginInit(
  db2int32 version,
  void *group_fns,
  db2secLogMessage *logMessage_fn,
  char **errormsg,
  db2int32  *errormsglen);
```

Note: If the plug-in library is compiled as C++, all functions must be declared with: `extern "C"`. DB2 relies on the underlying operating system dynamic loader to handle the C++ constructors and destructors used inside of a C++ user-written plug-in library.

The initialization function is the only function in the plug-in library that uses a prescribed function name. The other plug-in functions are referenced through function pointers returned from the initialization function. Server plug-ins are loaded when the DB2 server starts. Client plug-ins are loaded when required on the client. Immediately after DB2 loads a plug-in library, it will resolve the location of this initialization function and call it. The specific task of this function is as follows:

- Cast the functions pointer to a pointer to an appropriate functions structure
- Fill in the pointers to the other functions in the library
- Fill in the version number of the function pointer structure being returned

DB2 can potentially call the plug-in initialization function more than once. This situation can occur when an application dynamically loads the DB2 client library, unloads it, and reloads it again, then performs authentication functions from a plug-in both before and after reloading. In this situation, the plug-in library might not be unloaded and then re-loaded; however, this behavior varies depending on the operating system.

Another example of DB2 issuing multiple calls to a plug-in initialization function occurs during the execution of stored procedures or federated system calls, where the database server can itself act as a client. If the client and server plug-ins on the database server are in the same file, DB2 could call the plug-in initialization function twice.

If the plug-in detects that `db2secGroupPluginInit` is called more than once, it should handle this event as if it was directed to terminate and reinitialize the plug-in library. As such, the plug-in initialization function should do the entire cleanup tasks that a call to `db2secPluginTerm` would do before returning the set of function pointers again.

On a DB2 server running on a UNIX or Linux-based operating system, DB2 can potentially load and initialize plug-in libraries more than once in different processes.

Restrictions for developing security plug-in libraries

Following are the restrictions for developing plug-in libraries.

C-linkage
Plug-in libraries must be linked with C-linkage. Header files providing the prototypes, data structures needed to implement the plug-ins, and error code definitions are provided for C/C++ only. Functions that DB2 will resolve at load time must be declared with extern ″C″ if the plug-in library is compiled as C++.

.NET common language runtime is not supported
The .NET common language runtime (CLR) is not supported for compiling and linking source code for plug-in libraries.

Signal handlers
Plug-in libraries must not install signal handlers or change the signal mask, because this will interfere with DB2's signal handlers. Interfering with the DB2 signal handlers could seriously interfere with DB2's ability to report and recover from errors, including traps in the plug-in code itself. Plug-in libraries should also never throw C++ exceptions, as this can also interfere with DB2's error handling.

Thread-safe
Plug-in libraries must be thread-safe and re-entrant. The plug-in initialization function is the only API that is not required to be re-entrant. The plug-in initialization function could potentially be called multiple times from different processes; in which case, the plug-in will cleanup all used resources and reinitialize itself.

Exit handlers and overriding standard C library and operating system calls
Plug-in libraries should not override standard C library or operating system calls. Plug-in libraries should also not install exit handlers or `pthread_atfork` handlers. The use of exit handlers is not recommended because they could be unloaded before the program exits.

Library dependencies
On Linux or UNIX, the processes that load the plug-in libraries can be `setuid` or `setgid`, which means that they will not be able to rely on the `$LD_LIBRARY_PATH`, `$SHLIB_PATH`, or `$LIBPATH` environment variables to find dependent libraries. Therefore, plug-in libraries should not depend on additional libraries, unless any dependant libraries are accessible through other methods, such as the following:

- By being in `/lib` or `/usr/lib`
- By having the directories they reside in being specified OS-wide (such as in the `ld.so.conf` file on Linux)
- By being specified in the `RPATH` in the plug-in library itself

This restriction is not applicable to Windows operating systems.

Symbol collisions

When possible, plug-in libraries should be compiled and linked with any available options that reduce the likelihood of symbol collisions, such as those that reduce unbound external symbolic references. For example, use of the ″-Bsymbolic″ linker option on HP, Solaris, and Linux can help prevent problems related to symbol collisions. However, for plug-ins written on AIX, do not use the `"-brtl"` linker option explicitly or implicitly.

32-bit and 64-bit applications

32-bit applications must use 32-bit plug-ins. 64-bit applications must use 64-bit plug-ins. Refer to the topic about 32-bit and 64-bit considerations for more details.

Text strings

Input text strings are not guaranteed to be null-terminated, and output strings are not required to be null-terminated. Instead, integer lengths are given for all input strings, and pointers to integers are given for lengths to be returned.

Passing authorization ID parameters

An authorization ID (authid) parameter that DB2 passes into a plug-in (an input authid parameter) will contain an upper-case authid, with padded blanks removed. An authid parameter that a plug-in returns to DB2 (an output authid parameter) does not require any special treatment, but DB2 will fold the authid to upper-case and pad it with blanks according to the internal DB2 standard.

Size limits for parameters

The plug-in APIs use the following as length limits for parameters:

```
#define DB2SEC_MAX_AUTHID_LENGTH 255
#define DB2SEC_MAX_USERID_LENGTH 255
#define DB2SEC_MAX_USERNAMESPACE_LENGTH 255
#define DB2SEC_MAX_PASSWORD_LENGTH 255
#define DB2SEC_MAX_DBNAME_LENGTH 128
```

A particular plug-in implementation may require or enforce smaller maximum lengths for the authorization IDs, user IDs, and passwords. In particular, the operating system authentication plug-ins supplied with DB2 database systems are restricted to the maximum user, group and namespace length limits enforced by the operating system for cases where the operating system limits are lower than those stated above.

Security plug-in library extensions in AIX

On AIX systems, security plug-in libraries can have a file name extension of *.a* or *.so*. The mechanism used to load the plug-in library depends on which extension is used:

- Plug-in libraries with a file name extension of *.a* are assumed to be archives containing shared object members. These members must be named *shr.o* (32-bit) or *shr64.o* (64-bit). A single archive can contain both the 32-bit and 64-bit members, allowing it to be deployed on both types of platforms.

For example, to build a 32-bit archive style plug-in library:

```
xlc_r -qmkshrobj -o shr.o MyPlugin.c -bE:MyPlugin.exp
ar rv MyPlugin.a shr.o
```

- Plug-in libraries with a file name extension of *.so* are assumed to be dynamically loadable shared objects. Such an object is either 32-bit or 64-bit, depending on the compiler and linker options used when it was built. For example, to build a 32-bit plug-in library:

  ```
  xlc_r -qmkshrobj -o MyPlugin.so MyPlugin.c -bE:MyPlugin.exp
  ```

On all platforms other than AIX, security plug-in libraries are always assumed to be dynamically loadable shared objects.

Restrictions on security plug-ins

The following are restrictions on the use of security plug-ins:

DB2 database family support restrictions

You cannot use a GSS-API plug-in to authenticate connections between DB2 clients on Linux, UNIX, and Windows and another DB2 family servers such as DB2 for z/OS. You also cannot authenticate connections from another DB2 database family product, acting as a client, to a DB2 server on Linux, UNIX, or Windows.

If you use a DB2 client on Linux, UNIX, or Windows to connect to other DB2 database family servers, you can use client-side user ID/password plug-ins (such as the IBM-shipped operating system authentication plug-in), or you can write your own user ID/password plug-in. You can also use the built-in Kerberos plug-ins, or implement your own.

With a DB2 client on Linux, UNIX, or Windows, you should not catalog a database using the GSSPLUGIN authentication type.

Restrictions on the AUTHID identifier. Version 9.5, and later, of the DB2 database system allows you to have an 128-byte authorization ID, but when the authorization ID is interpreted as an operating system user ID or group name, the operating system naming restrictions apply (for example, a limitation to 8 or 30 character user IDs and 30 character group names). Therefore, while you can grant an 128-byte authorization ID, it is not possible to connect as a user that has that authorization ID. If you write your own security plugin, you should be able to take full advantage of the extended sizes for the authorization ID. For example, you can give your security plugin a 30-byte user ID and it can return an 128-byte authorization ID during authentication that you are able to connect with.

WebSphere® Federation Server support restrictions

DB2 II does not support the use of delegated credentials from a GSS_API plug-in to establish outbound connections to data sources. Connections to data sources must continue to use the CREATE USER MAPPING command.

Database Administration Server support restrictions

The DB2 Administration Server (DAS) does not support security plug-ins. The DAS only supports the operating system authentication mechanism.

Security plug-in problem and restriction for DB2 clients (Windows)

When developing security plug-ins that will be deployed in DB2 clients on Windows operating systems, do not unload any auxiliary libraries in the plug-in termination function. This restriction applies to all types of client security plug-ins, including group, user ID and password, Kerberos, and GSS-API plug-ins. Since these termination APIs such as db2secPluginTerm, db2secClientAuthPluginTerm and db2secServerAuthPluginTerm are not called on any Windows platform, you need to do the appropriate resource cleanup.

This restriction is related to cleanup issues associated with the unloading of DLLs on Windows.

Loading plug-in libraries on AIX with extension of .a or .so

On AIX, security plug-in libraries can have a file name extension of `.a` or `.so`. The mechanism used to load the plug-in library depends on which extension is used:

- Plug-in libraries with a file name extension of `.a`

  Plug-in libraries with file name extensions of .a are assumed to be archives containing shared object members. These members must be named `shr.o` (32-bit) or `shr64.o` (64-bit). A single archive can contain both the 32-bit and 64-bit members, allowing it to be deployed on both types of platforms.

  For example, to build a 32-bit archive style plug-in library:

  ```
  xlc_r -qmkshrobj -o shr.o MyPlugin.c -bE:MyPlugin.exp
  ar rv MyPlugin.a shr.o
  ```

- Plug-in libraries with a file name extension of `.so`

  Plug-in libraries with file name extensions of .so are assumed to be dynamically loadable shared objects. Such an object is either 32-bit or 64-bit, depending on the compiler and linker options used when it was built. For example, to build a 32-bit plug-in library:

  ```
  xlc_r -qmkshrobj -o MyPlugin.so MyPlugin.c -bE:MyPlugin.exp
  ```

On all platforms other than AIX, security plug-in libraries are always assumed to be dynamically loadable shared objects.

GSS-API security plug-ins do not support message encryption and signing

Message encryption and signing is not available in GSS-API security plug-ins.

Return codes for security plug-ins

All security plug-in APIs must return an integer value to indicate the success or failure of the execution of the API. A return code value of 0 indicates that the API ran successfully. All negative return codes, with the exception of -3, -4, and -5, indicate that the API encountered an error.

All negative return codes returned from the security-plug-in APIs are mapped to SQLCODE -1365, SQLCODE -1366, or SQLCODE -30082, with the exception of return codes with the -3, -4, or -5. The values -3, -4, and -5 are used to indicate whether or not an authorization ID represents a valid user or group.

All the security plug-in API return codes are defined in db2secPlugin.h, which can be found in the DB2 include directory: SQLLIB/include.

Details regarding all of the security plug-in return codes are presented in the following table:

Table 38. Security plug-in return codes

Return code	Define value	Meaning	Applicable APIs
0	DB2SEC_PLUGIN_OK	The plug-in API executed successfully.	All
-1	DB2SEC_PLUGIN_UNKNOWNERROR	The plug-in API encountered an unexpected error.	All
-2	DB2SEC_PLUGIN_BADUSER	The user ID passed in as input is not defined.	db2secGenerateInitialCred db2secValidatePassword db2secRemapUserid db2secGetGroupsForUser
-3	DB2SEC_PLUGIN_INVALIDUSERORGROUP	No such user or group.	db2secDoesAuthIDExist db2secDoesGroupExist
-4	DB2SEC_PLUGIN_USERSTATUSNOTKNOWN	Unknown user status. This is not treated as an error by DB2; it is used by a GRANT statement to determine if an authid represents a user or an operating system group.	db2secDoesAuthIDExist
-5	DB2SEC_PLUGIN_GROUPSTATUSNOTKNOWN	Unknown group status. This is not treated as an error by DB2; it is used by a GRANT statement to determine if an authid represents a user or an operating system group.	db2secDoesGroupExist
-6	DB2SEC_PLUGIN_UID_EXPIRED	User ID expired.	db2secValidatePassword db2GetGroupsForUser db2secGenerateInitialCred
-7	DB2SEC_PLUGIN_PWD_EXPIRED	Password expired.	db2secValidatePassword db2GetGroupsForUser db2secGenerateInitialCred
-8	DB2SEC_PLUGIN_USER_REVOKED	User revoked.	db2secValidatePassword db2GetGroupsForUser
-9	DB2SEC_PLUGIN_USER_SUSPENDED	User suspended.	db2secValidatePassword db2GetGroupsForUser
-10	DB2SEC_PLUGIN_BADPWD	Bad password.	db2secValidatePassword db2secRemapUserid db2secGenerateInitialCred
-11	DB2SEC_PLUGIN_BAD_NEWPASSWORD	Bad new password.	db2secValidatePassword db2secRemapUserid

Table 38. Security plug-in return codes (continued)

Return code	Define value	Meaning	Applicable APIs
-12	DB2SEC_PLUGIN _CHANGEPASSWORD _NOTSUPPORTED	Change password not supported.	db2secValidatePassword db2secRemapUserid db2secGenerateInitialCred
-13	DB2SEC_PLUGIN_NOMEM	Plug-in attempt to allocate memory failed due to insufficient memory.	All
-14	DB2SEC_PLUGIN_DISKERROR	Plug-in encountered a disk error.	All
-15	DB2SEC_PLUGIN_NOPERM	Plug-in attempt to access a file failed because of wrong permissions on the file.	All
-16	DB2SEC_PLUGIN_NETWORKERROR	Plug-in encountered a network error.	All
-17	DB2SEC_PLUGIN _CANTLOADLIBRARY	Plug-in is unable to load a required library.	db2secGroupPluginInit db2secClientAuthPluginInit db2secServerAuthPluginInit
-18	DB2SEC_PLUGIN_CANT _OPEN_FILE	Plug-in is unable to open and read a file for a reason other than missing file or inadequate file permissions.	All
-19	DB2SEC_PLUGIN_FILENOTFOUND	Plug-in is unable to open and read a file, because the file is missing from the file system.	All
-20	DB2SEC_PLUGIN _CONNECTION_DISALLOWED	The plug-in is refusing the connection because of the restriction on which database is allowed to connect, or the TCP/IP address cannot connect to a specific database.	All server-side plug-in APIs.
-21	DB2SEC_PLUGIN_NO_CRED	GSS API plug-in only: initial client credential is missing.	db2secGetDefaultLoginContext db2secServerAuthPluginInit
-22	DB2SEC_PLUGIN_CRED_EXPIRED	GSS API plug-in only: client credential has expired.	db2secGetDefaultLoginContext db2secServerAuthPluginInit
-23	DB2SEC_PLUGIN _BAD_PRINCIPAL_NAME	GSS API plug-in only: the principal name is invalid.	db2secProcessServer PrincipalName
-24	DB2SEC_PLUGIN _NO_CON_DETAILS	This return code is returned by the db2secGetConDetails callback (for example, from DB2 to the plug-in) to indicate that DB2 is unable to determine the client's TCP/IP address.	db2secGetConDetails
-25	DB2SEC_PLUGIN _BAD_INPUT_PARAMETERS	Some parameters are not valid or are missing when plug-in API is called.	All

Table 38. Security plug-in return codes (continued)

Return code	Define value	Meaning	Applicable APIs
-26	DB2SEC_PLUGIN _INCOMPATIBLE_VER	The version of the APIs reported by the plug-in is not compatible with DB2.	db2secGroupPluginInit db2secClientAuthPluginInit db2secServerAuthPluginInit
-27	DB2SEC_PLUGIN_PROCESS_LIMIT	Insufficient resources are available for the plug-in to create a new process.	All
-28	DB2SEC_PLUGIN_NO_LICENSES	The plug-in encountered a user license problem. A possibility exists that the underlying mechanism license has reached the limit.	All

Error message handling for security plug-ins

When an error occurs in a security plug-in API, the API can return an ASCII text string in the `errormsg` field to provide a more specific description of the problem than the return code.

For example, the `errormsg` string can contain `"File /home/db2inst1/mypasswd.txt does not exist."` DB2 will write this entire string into the DB2 administration notification log, and will also include a truncated version as a token in some SQL messages. Because tokens in SQL messages can only be of limited length, these messages should be kept short, and important variable portions of these messages should appear at the front of the string. To aid in debugging, consider adding the name of the security plug-in to the error message.

For non-urgent errors, such as password expired errors, the `errormsg` string will only be dumped when the DIAGLEVEL database manager configuration parameter is set at 4.

The memory for these error messages must be allocated by the security plug-in. Therefore, the plug-ins must also provide an API to free this memory: `db2secFreeErrormsg`.

The `errormsg` field will only be checked by DB2 if an API returns a non-zero value. Therefore, the plug-in should not allocate memory for this returned error message if there is no error.

At initialization time a message logging function pointer, `logMessage_fn`, is passed to the group, client, and server plug-ins. The plug-ins can use the function to log any debugging information to db2diag.log. For example:

```
// Log an message indicate init successful
(*(logMessage_fn))(DB2SEC_LOG_CRITICAL,
                   "db2secGroupPluginInit successful",
                    strlen("db2secGroupPluginInit successful"));
```

For more details about each parameter for the `db2secLogMessage` function, refer to the initialization API for each of the plug-in types.

Calling sequences for the security plug-in APIs

These are the main scenarios in which the DB2 database manager will call security plug-in APIs:

- On a client for a database connection (implicit and explicit)
  - CLIENT
  - Server based (SERVER, SERVER_ENCRYPT, DATA_ENCRYPT)
  - GSSAPI and Kerberos
- On a client, server, or gateway for local authorization
- On a server for a database connection
- On a server for a grant statement
- On a server to get a list of groups to which an authorization ID belongs

Note: The DB2 database servers treat database actions requiring local authorizations, such as db2start, db2stop, and db2trc like client applications.

For each of these operations, the sequence with which the DB2 database manager calls the security plug-in APIs is different. Following are the sequences of APIs called by the DB2 database manager for each of these scenarios.

CLIENT - implicit

When the user-configured authentication type is CLIENT, the DB2 client application will call the following security plug-in APIs:

- `db2secGetDefaultLoginContext();`
- `db2secValidatePassword();`
- `db2secFreetoken();`

For an implicit authentication, that is, when you connect without specifying a particular user ID or password, the `db2secValidatePassword` API is called if you are using a user ID/password plug-in. This API permits plug-in developers to prohibit implicit authentication if necessary.

CLIENT - explicit

On an explicit authentication, that is, when you connect to a database in which both the user ID and password are specified, if the *authentication* database manager configuration parameter is set to CLIENT the DB2 client application will call the following security plug-in APIs multiple times if the implementation requires it:

- `db2secRemapUserid();`
- `db2secValidatePassword();`
- `db2secFreeToken();`

Server based (SERVER, SERVER_ENCRYPT, DATA_ENCRYPT) - implicit

On an implicit authentication, when the client and server have negotiated user ID/password authentication (for instance, when the *srvcon_auth* parameter at the server is set to SERVER; SERVER_ENCRYPT, DATA_ENCRYPT, or DATA_ENCRYPT_CMP), the client application will call the following security plug-in APIs:

- `db2secGetDefaultLoginContext();`
- `db2secFreeToken();`

Server based (SERVER, SERVER_ENCRYPT, DATA_ENCRYPT) - explicit

On an explicit authentication, when the client and server have negotiated userid/password authentication (for instance, when the srvcon_auth

parameter at the server is set to SERVER; SERVER_ENCRYPT, DATA_ENCRYPT, or DATA_ENCRYPT_CMP), the client application will call the following security plug-in APIs:

- `db2secRemapUserid();`

GSSAPI and Kerberos - implicit

On an implicit authentication, when the client and server have negotiated GSS-API or Kerberos authentication (for instance, when the *srvcon_auth* parameter at the server is set to KERBEROS; KRB_SERVER_ENCRYPT, GSSPLUGIN, or GSS_SERVER_ENCRYPT), the client application will call the following security plug-in APIs. (The call to gss_init_sec_context() will use GSS_C_NO_CREDENTIAL as the input credential.)

- `db2secGetDefaultLoginContext();`
- `db2secProcessServerPrincipalName();`
- `gss_init_sec_context();`
- `gss_release_buffer();`
- `gss_release_name();`
- `gss_delete_sec_context();`
- `db2secFreeToken();`

With multi-flow GSS-API support, `gss_init_sec_context()` can be called multiple times if the implementation requires it.

GSSAPI and Kerberos - explicit

If the negotiated authentication type is GSS-API or Kerberos, the client application will call the following security plug-in APIs for GSS-API plug-ins in the following sequence. These APIs are used for both implicit and explicit authentication unless otherwise stated.

- `db2secProcessServerPrincipalName();`
- `db2secGenerateInitialCred();` (For explicit authentication only)
- `gss_init_sec_context();`
- `gss_release_buffer ();`
- `gss_release_name();`
- `gss_release_cred();`
- `db2secFreeInitInfo();`
- `gss_delete_sec_context();`
- `db2secFreeToken();`

The API `gss_init_sec_context()` may be called multiple times if a mutual authentication token is returned from the server and the implementation requires it.

On a client, server, or gateway for local authorization

For a local authorization, the DB2 command being used will call the following security plug-in APIs:

- `db2secGetDefaultLoginContext();`
- `db2secGetGroupsForUser();`
- `db2secFreeToken();`
- `db2secFreeGroupList();`

These APIs will be called for both user ID/password and GSS-API authentication mechanisms.

On a server for a database connection

For a database connection on the database server, the DB2 agent process or thread will call the following security plug-in APIs for the user ID/password authentication mechanism:

- `db2secValidatePassword();` Only if the *authentication* database configuration parameter is not CLIENT
- `db2secGetAuthIDs();`
- `db2secGetGroupsForUser();`
- `db2secFreeToken();`
- `db2secFreeGroupList();`

For a CONNECT to a database, the DB2 agent process or thread will call the following security plug-in APIs for the GSS-API authentication mechanism:

- `gss_accept_sec_context();`
- `gss_release_buffer();`
- `db2secGetAuthIDs();`
- `db2secGetGroupsForUser();`
- `gss_delete_sec_context();`
- `db2secFreeGroupListMemory();`

On a server for a GRANT statement

For a GRANT statement that does not specify the USER or GROUP keyword, (for example, "`GRANT CONNECT ON DATABASE TO user1`"), the DB2 agent process or thread must be able to determine if `user1` is a user, a group, or both. Therefore, the DB2 agent process or thread will call the following security plug-in APIs:

- `db2secDoesGroupExist();`
- `db2secDoesAuthIDExist();`

On a server to get a list of groups to which an authid belongs

From your database server, when you need to get a list of groups to which an authorization ID belongs, the DB2 agent process or thread will call the following security plug-in API with only the authorization ID as input:

- `db2secGetGroupsForUser();`

There will be no token from other security plug-ins.

Chapter 8. Security plug-in APIs

To enable you to customize the DB2 database system authentication and group membership lookup behavior, the DB2 database system provides APIs that you can use to modify existing plug-in modules or build new security plug-in modules.

When you develop a security plug-in module, you need to implement the standard authentication or group membership lookup functions that the DB2 database manager will invoke. For the three available types of plug-in modules, the functionality you need to implement is as follows:

Group retrieval
Retrieves group membership information for a given user and determines if a given string represents a valid group name.

User ID/password authentication
Authentication that identifies the default security context (client only), validates and optionally changes a password, determines if a given string represents a valid user (server only), modifies the user ID or password provided on the client before it is sent to the server (client only), returns the DB2 authorization ID associated with a given user.

GSS-API authentication
Authentication that implements the required GSS-API functions, identifies the default security context (client side only), generates initial credentials based on user ID and password, and optionally changes password (client side only), creates and accepts security tickets, and returns the DB2 authorization ID associated with a given GSS-API security context.

The following are the definitions for terminology used in the descriptions of the plug-in APIs.

Plug-in
A dynamically loadable library that DB2 will load to access user-written authentication or group membership lookup functions.

Implicit authentication
A connection to a database without specifying a user ID or a password.

Explicit authentication
A connection to a database in which both the user ID and password are specified.

Authid
An internal ID representing an individual or group to which authorities and privileges within the database are granted. Internally, a DB2 authid is folded to upper-case and is a minimum of 8 characters (blank padded to 8 characters). Currently, DB2 requires authids, user IDs, passwords, group names, namespaces, and domain names that can be represented in 7-bit ASCII.

Local authorization
Authorization that is local to the server or client that implements it, that checks if a user is authorized to perform an action (other than connecting to the database), such as starting and stopping the database manager, turning DB2 trace on and off, or updating the database manager configuration.

Namespace
A collection or grouping of users within which individual user identifiers must be unique. Common examples include Windows domains and Kerberos Realms. For example, within the Windows domain ″usa.company.com″ all user names must be unique. For example, ″user1@usa.company.com″. The same user ID in another domain, as in the case of ″user1@canada.company.com″, however refers to a different person. A fully qualified user identifier includes a user ID and namespace pair; for example, ″user@domain.name″ or ″domain\user″.

Input Indicates that DB2 will fill in the value for the security plug-in API parameter.

Output
Indicates that the security plug-in API will fill in the value for the API parameter.

APIs for group retrieval plug-ins

For the group retrieval plug-in module, you need to implement the following APIs:

- `db2secGroupPluginInit`

  Note: The db2secGroupPluginInit API takes as input a pointer, `*logMessage_fn`, to an API with the following prototype:

  ```
  SQL_API_RC (SQL_API_FN db2secLogMessage)
  (
  db2int32 level,
  void    *data,
  db2int32 length
  );
  ```

  The db2secLogMessage API allows the plug-in to log messages to db2diag.log for debugging or informational purposes. This API is provided by the DB2 database system, so you need not implement it.
- `db2secPluginTerm`
- `db2secGetGroupsForUser`
- `db2secDoesGroupExist`
- `db2secFreeGroupListMemory`
- `db2secFreeErrormsg`
- The only API that must be resolvable externally is `db2secGroupPluginInit`. This API will take a `void *` parameter, which should be cast to the type:

  ```
  typedef struct db2secGroupFunctions_1
  {
  db2int32 version;
  db2int32 plugintype;
  SQL_API_RC (SQL_API_FN * db2secGetGroupsForUser)
  (
  const char *authid,
  db2int32    authidlen,
  const char *userid,
  db2int32    useridlen,
  const char *usernamespace,
  db2int32    usernamespacelen,
  db2int32    usernamespacetype,
  const char *dbname,
  db2int32    dbnamelen,
  const void *token,
  db2int32    tokentype,
  ```

```
db2int32    location,
const char *authpluginname,
db2int32    authpluginnamelen,
void      **grouplist,
db2int32   *numgroups,
char      **errormsg,
db2int32   *errormsglen
);

SQL_API_RC (SQL_API_FN * db2secDoesGroupExist)
(
const char *groupname,
db2int32    groupnamelen,
char      **errormsg,
db2int32   *errormsglen
);

SQL_API_RC (SQL_API_FN * db2secFreeGroupListMemory)
(
void     *ptr,
char    **errormsg,
db2int32 *errormsglen
);

SQL_API_RC (SQL_API_FN * db2secFreeErrormsg)
(
char *msgtobefree
);

SQL_API_RC (SQL_API_FN * db2secPluginTerm)
(
char    **errormsg,
db2int32 *errormsglen
);

} db2secGroupFunctions_1;
```

The `db2secGroupPluginInit` API assigns the addresses for the rest of the externally available functions.

Note: The _1 indicates that this is the structure corresponding to version 1 of the API. Subsequent interface versions will have the extension _2, _3, and so on.

db2secDoesGroupExist API - Check if group exists

Determines if an authid represents a group.

If the groupname exists, the API must be able to return the value DB2SEC_PLUGIN_OK, to indicate success. It must also be able to return the value DB2SEC_PLUGIN_INVALIDUSERORGROUP if the group name is not valid. It is permissible for the API to return the value DB2SEC_PLUGIN_GROUPSTATUSNOTKNOWN if it is impossible to determine if the input is a valid group. If an invalid group (DB2SEC_PLUGIN_INVALIDUSERORGROUP) or group not known (DB2SEC_PLUGIN_GROUPSTATUSNOTKNOWN) value is returned, DB2 might not be able to determine whether the authid is a group or user when issuing the GRANT statement without the keywords USER and GROUP, which would result in the error SQLCODE -569, SQLSTATE 56092 being returned to the user.

API and data structure syntax

```
SQL_API_RC ( SQL_API_FN *db2secDoesGroupExist)
                      ( const char *groupname,
                        db2int32 groupnamelen,
                        char      **errormsg,
                        db2int32 *errormsglen );
```

db2secDoesGroupExist API parameters

groupname
Input. An authid, upper-cased, with no trailing blanks.

groupnamelen
Input. Length in bytes of the groupname parameter value.

errormsg
Output. A pointer to the address of an ASCII error message string allocated by the plug-in that can be returned in this parameter if the db2secDoesGroupExist API execution is not successful.

errormsglen
Output. A pointer to an integer that indicates the length in bytes of the error message string in errormsg parameter.

db2secFreeErrormsg API - Free error message memory

Frees the memory used to hold an error message from a previous API call. This is the only API that does not return an error message. If this API returns an error, DB2 will log it and continue.

API and data structure syntax

```
SQL_API_RC ( SQL_API_FN *db2secFreeErrormsg)
                      ( char *errormsg );
```

db2secFreeErrormsg API parameters

msgtofree
Input. A pointer to the error message allocated from a previous API call.

db2secFreeGroupListMemory API - Free group list memory

Frees the memory used to hold the list of groups from a previous call to db2secGetGroupsForUser API.

API and data structure syntax

```
SQL_API_RC ( SQL_API_FN *db2secFreeGroupListMemory)
                      ( void *ptr,
                        char      **errormsg,
                        db2int32 *errormsglen );
```

db2secFreeGroupListMemory API parameters

ptr Input. Pointer to the memory to be freed.

errormsg
Output. A pointer to the address of an ASCII error message string allocated by the plug-in that can be returned in this parameter if the db2secFreeGroupListMemory API execution is not successful.

errormsglen
Output. A pointer to an integer that indicates the length in bytes of the error message string in the errormsg parameter.

db2secGetGroupsForUser API - Get list of groups for user

Returns the list of groups to which a user belongs.

API and data structure syntax

```
SQL_API_RC ( SQL_API_FN *db2secGetGroupsForUser)
                       ( const char *authid,
                         db2int32 authidlen,
                         const char *userid,
                         db2int32 useridlen,
                         const char *usernamespace,
                         db2int32 usernamespacelen,
                         db2int32 usernamespacetype,
                         const char *dbname,
                         db2int32 dbnamelen,
                         void *token,
                         db2int32 tokentype,
                         db2int32 location,
                         const char *authpluginname,
                         db2int32 authpluginnamelen,
                         void      **grouplist,
                         db2int32 *numgroups,
                         char      **errormsg,
                         db2int32 *errormsglen );
```

db2secGetGroupsForUser API parameters

authid Input. This parameter value is an SQL authid, which means that DB2 converts it to an uppercase character string with no trailing blanks. DB2 will always provide a non-null value for the authid parameter. The API must be able to return a list of groups to which the authid belongs without depending on the other input parameters. It is permissible to return a shortened or empty list if this cannot be determined.

If a user does not exist, the API must return the return code DB2SEC_PLUGIN_BADUSER. DB2 does not treat the case of a user not existing as an error, since it is permissible for an authid to not have any groups associated with it. For example, the db2secGetAuthids API can return an authid that does not exist on the operating system. The authid is not associated with any groups, however, it can still be assigned privileges directly.

If the API cannot return a complete list of groups using only the authid, then there will be some restrictions on certain SQL functions related to group support. For a list of possible problem scenarios, refer to the Usage notes section in this topic.

authidlen
Input. Length in bytes of the authid parameter value. The DB2 database manager always provides a non-zero value for the authidlen parameter.

userid Input. This is the user ID corresponding to the authid. When this API is called on the server in a non-connect scenario, this parameter will not be filled by DB2.

useridlen
Input. Length in bytes of the userid parameter value.

usernamespace
Input. The namespace from which the user ID was obtained. When the user ID is not available, this parameter will not be filled by the DB2 database manager.

usernamespacelen
Input. Length in bytes of the usernamespace parameter value.

usernamespacetype
Input. The type of namespace. Valid values for the usernamespacetype parameter (defined in db2secPlugin.h) are:

- DB2SEC_NAMESPACE_SAM_COMPATIBLE Corresponds to a username style like domain\myname
- DB2SEC_NAMESPACE_USER_PRINCIPAL Corresponds to a username style like myname@domain.ibm.com

Currently, the DB2 database system only supports the value DB2SEC_NAMESPACE_SAM_COMPATIBLE. When the user ID is not available, the usernamespacetype parameter is set to the value DB2SEC_USER_NAMESPACE_UNDEFINED (defined in db2secPlugin.h).

dbname
Input. Name of the database being connected to. This parameter can be NULL in a non-connect scenario.

dbnamelen
Input. Length in bytes of the dbname parameter value. This parameter is set to 0 if dbname parameter is NULL in a non-connect scenario.

token Input. A pointer to data provided by the authentication plug-in. It is not used by DB2. It provides the plug-in writer with the ability to coordinate user and group information. This parameter might not be provided in all cases (for example, in a non-connect scenario), in which case it will be NULL. If the authentication plug-in used is GSS-API based, the token will be set to the GSS-API context handle (gss_ctx_id_t).

tokentype
Input. Indicates the type of data provided by the authentication plug-in. If the authentication plug-in used is GSS-API based, the token will be set to the GSS-API context handle (gss_ctx_id_t). If the authentication plug-in used is user ID/password based, it will be a generic type. Valid values for the tokentype parameter (defined in db2secPlugin.h) are:

- `DB2SEC_GENERIC`: Indicates that the token is from a user ID/password based plug-in.
- `DB2SEC_GSSAPI_CTX_HANDLE`: Indicates that the token is from a GSS-API (including Kerberos) based plug-in.

location
Input. Indicates whether DB2 is calling this API on the client side or server side. Valid values for the location parameter (defined in db2secPlugin.h) are:

- `DB2SEC_SERVER_SIDE`: The API is to be called on the database server.
- `DB2SEC_CLIENT_SIDE`: The API is to be called on a client.

authpluginname
Input. Name of the authentication plug-in that provided the data in the token. The db2secGetGroupsForUser API might use this information in determining the correct group memberships. This parameter might not be filled by DB2 if the authid is not authenticated (for example, if the authid does not match the current connected user).

authpluginnamelen
Input. Length in bytes of the authpluginname parameter value.

grouplist
Output. List of groups to which the user belongs. The list of groups must be returned as a pointer to a section of memory allocated by the plug-in containing concatenated varchars (a varchar is a character array in which the first byte indicates the number of bytes following it). The length is an unsigned char (1 byte) and that limits the maximum length of a groupname to 255 characters. For example, ″\006GROUP1\ 007MYGROUP\008MYGROUP3″. Each group name should be a valid DB2 authid. The memory for this array must be allocated by the plug-in. The plug-in must therefore provide an API, such as the db2secFreeGroupListMemory API that DB2 will call to free the memory.

numgroups
Output. The number of groups contained in the grouplist parameter.

errormsg
Output. A pointer to the address of an ASCII error message string allocated by the plug-in that can be returned in this parameter if the db2secGetGroupsForUser API execution is not successful.

errormsglen
Output. A pointer to an integer that indicates the length in bytes of the error message string in errormsg parameter.

Usage notes

The following is a list of scenarios when problems can occur if an incomplete group list is returned by this API to DB2:

- Embedded SQL application with DYNAMICRULES BIND (or DEFINEDBIND or INVOKEDBIND if the packages are running as a standalone application). DB2 checks for SYSADM membership and the application will fail if it is dependent on the implicit DBADM authority granted by being a member of the SYSADM group.
- Alternate authorization is provided in CREATE SCHEMA statement. Group lookup will be performed against the AUTHORIZATION NAME parameter if there are nested CREATE statements in the CREATE SCHEMA statement.
- Embedded SQL applications with DYNAMICRULES DEFINERUN/ DEFINEBIND and the packages are running in a routine context. DB2 checks for SYSADM membership of the routine definer and the application will fail if it is dependent on the implicit DBADM authority granted by being a member of the SYSADM group.
- Processing a jar file in an MPP environment. In an MPP environment, the jar processing request is sent from the coordinator node with the session authid. The catalog node received the requests and process the jar files based on the privilege of the session authid (the user executing the jar processing requests).
  - Install jar file. The session authid needs to have one of the following rights: SYSADM, DBADM, or CREATEIN (implicit or explicit on the jar schema). The operation will fail if the above rights are granted to group containing the session authid, but not explicitly to the session authid or if only SYSADM is held, since SYSADM membership is determined by membership in the group defined by a database configuration parameter.
  - Remove jar file. The session authid needs to have one of the following rights rights: SYSADM, DBADM, or DROPIN (implicit or explicit on the jar schema), or is the definer of the jar file. The operation will fail if the above rights are granted to group containing the session authid, but not explicitly to the session authid, and if the session authid is not the definer of the jar file or

if only SYSADM is held since SYSADM membership is determined by membership in the group defined by a database configuration parameter.
  - Replace jar file. This is same as removing the jar file, followed by installing the jar file. Both of the above apply.
- Regenerate views. This is triggered by the ALTER TABLE, ALTER COLUMN, SET DATA TYPE VARCHAR/VARGRAPHIC statements, or during migration. The DB2 database manager checks for SYSADM membership of the view definer. The application will fail if it is dependent on the implicit DBADM authority granted by being a member of the SYSADM group.
- When SET SESSION_USER statement is issued. Subsequent DB2 operations are run under the context of the authid specified by this statement. These operations will fail if the privileges required are owned by one of the SESSION_USER's group is not explicitly granted to the SESSION_USER authid.

db2secGroupPluginInit API - Initialize group plug-in

Initialization API, for the group-retrieval plug-in, that the DB2 database manager calls immediately after loading the plug-in.

API and data structure syntax

```
SQL_API_RC SQL_API_FN db2secGroupPluginInit
                            ( db2int32 version,
                              void *group_fns,
                              db2secLogMessage *logMessage_fn,
                              char     **errormsg,
                              db2int32 *errormsglen );
```

db2secGroupPluginInit API parameters

version
Input. The highest version of the API supported by the instance loading that plugin. The value DB2SEC_API_VERSION (in db2secPlugin.h) contains the latest version number of the API that the DB2 database manager currently supports.

group_fns
Output. A pointer to the db2secGroupFunctions_<version_number> (also known as group_functions_<version_number>) structure. The db2secGroupFunctions_<version_number> structure contains pointers to the APIs implemented for the group-retrieval plug-in. In future, there might be different versions of the APIs (for example, db2secGroupFunctions_<version_number>), so the group_fns parameter is cast as a pointer to the db2secGroupFunctions_<version_number> structure corresponding to the version the plug-in has implemented. The first parameter of the group_functions_<version_number> structure tells DB2 the version of the APIs that the plug-in has implemented. Note: The casting is done only if the DB2 version is higher or equal to the version of the APIs that the plug-in has implemented. The version number represents the version of the APIs implemented by the plugin, and the pluginType should be set to DB2SEC_PLUGIN_TYPE_GROUP.

logMessage_fn
Input. A pointer to the db2secLogMessage API, which is implemented by the DB2 database system. The db2secGroupPluginInit API can call the db2secLogMessage API to log messages to db2diag.log for debugging or informational purposes. The first parameter (level) of db2secLogMessage API specifies the type of diagnostic errors that will be recorded in the db2diag.log file and the last two parameters respectively are the message

string and its length. The valid values for the first parameter of dbesecLogMessage API (defined in db2secPlugin.h) are:

- DB2SEC_LOG_NONE: (0) No logging
- DB2SEC_LOG_CRITICAL: (1) Severe Error encountered
- DB2SEC_LOG_ERROR: (2) Error encountered
- DB2SEC_LOG_WARNING: (3) Warning
- DB2SEC_LOG_INFO: (4) Informational

The message text will show up in the diag.log only if the value of the 'level' parameter of the db2secLogMessage API is less than or equal to the diaglevel database manager configuration parameter. So for example, if you use the DB2SEC_LOG_INFO value, the message text will only show up in the db2diag.log if the diaglevel database manager configuration parameter is set to 4.

errormsg
Output. A pointer to the address of an ASCII error message string allocated by the plug-in that can be returned in this parameter if the db2secGroupPluginInit API execution is not successful.

errormsglen
Output. A pointer to an integer that indicates the length in bytes of the error message string in errormsg parameter.

db2secPluginTerm - Clean up group plug-in resources

Frees resources used by the group-retrieval plug-in.

This API is called by the DB2 database manager just before it unloads the group-retrieval plug-in. It should be implemented in a manner that it does a proper cleanup of any resources the plug-in library holds, for instance, free any memory allocated by the plug-in, close files that are still open, and close network connections. The plug-in is responsible for keeping track of these resources in order to free them. This API is not called on any Windows platform.

API and data structure syntax

```
SQL_API_RC ( SQL_API_FN *db2secPluginTerm)
                        ( char      **errormsg,
                          db2int32 *errormsglen );
```

db2secPluginTerm API parameters

errormsg
Output. A pointer to the address of an ASCII error message string allocated by the plug-in that can be returned in this parameter if the db2secPluginTerm API execution is not successful.

errormsglen
Output. A pointer to an integer that indicates the length in bytes of the error message string in errormsg parameter.

APIs for user ID/password authentication plug-ins

For the user ID/password plug-in module, you need to implement the following client-side APIs:

- db2secClientAuthPluginInit

Note: The db2secClientAuthPluginInit API takes as input a pointer, *logMessage_fn, to an API with the following prototype:

```
SQL_API_RC (SQL_API_FN db2secLogMessage)
(
db2int32 level,
void    *data,
db2int32 length
);
```

The db2secLogMessage API allows the plug-in to log messages to db2diag.log for debugging or informational purposes. This API is provided by the DB2 database system, so you need not implement it.

- db2secClientAuthPluginTerm
- db2secGenerateInitialCred (Only used for gssapi)
- db2secRemapUserid (Optional)
- db2secGetDefaultLoginContext
- db2secValidatePassword
- db2secProcessServerPrincipalName (This is only for GSS-API)
- db2secFreeToken (Functions to free memory held by the DLL)
- db2secFreeErrormsg
- db2secFreeInitInfo
- The only API that must be resolvable externally is db2secClientAuthPluginInit. This API will take a void * parameter, which should be cast to either:

```
typedef struct db2secUseridPasswordClientAuthFunctions_1
{
db2int32 version;
db2int32 plugintype;

SQL_API_RC (SQL_API_FN * db2secGetDefaultLoginContext)
(
char        authid[DB2SEC_MAX_AUTHID_LENGTH],
db2int32   *authidlen,
char        userid[DB2SEC_MAX_USERID_LENGTH],
db2int32   *useridlen,
db2int32    useridtype,
char        usernamespace[DB2SEC_MAX_USERNAMESPACE_LENGTH],
db2int32   *usernamespacelen,
db2int32   *usernamespacetype,
const char *dbname,
db2int32    dbnamelen,
void      **token,
char      **errormsg,
db2int32   *errormsglen
);
/* Optional */
SQL_API_RC (SQL_API_FN * db2secRemapUserid)
(
char        userid[DB2SEC_MAX_USERID_LENGTH],
db2int32   *useridlen,
char        usernamespace[DB2SEC_MAX_USERNAMESPACE_LENGTH],
db2int32   *usernamespacelen,
db2int32   *usernamespacetype,
char        password[DB2SEC_MAX_PASSWORD_LENGTH],
db2int32   *passwordlen,
char        newpassword[DB2SEC_MAX_PASSWORD_LENGTH],
db2int32   *newpasswordlen,
const char *dbname,
db2int32    dbnamelen,
char      **errormsg,
db2int32   *errormsglen
```

```
);

SQL_API_RC (SQL_API_FN * db2secValidatePassword)
(
const char *userid,
db2int32    useridlen,
const char *usernamespace,
db2int32    usernamespacelen,
db2int32    usernamespacetype,
const char *password,
db2int32    passwordlen,
const char *newpassword,
db2int32    newpasswordlen,
const char *dbname,
db2int32    dbnamelen,
db2Uint32   connection_details,
void      **token,
char      **errormsg,
db2int32   *errormsglen
);

SQL_API_RC (SQL_API_FN * db2secFreeToken)
(
void     **token,
char     **errormsg,
db2int32  *errormsglen
);

SQL_API_RC (SQL_API_FN * db2secFreeErrormsg)
(
char *errormsg
);

SQL_API_RC (SQL_API_FN * db2secClientAuthPluginTerm)
(
char     **errormsg,
db2int32  *errormsglen
);
}
```

or

```
typedef struct db2secGssapiClientAuthFunctions_1
{
db2int32 version;
db2int32 plugintype;

SQL_API_RC (SQL_API_FN * db2secGetDefaultLoginContext)
(
char        authid[DB2SEC_MAX_AUTHID_LENGTH],
db2int32   *authidlen,
char        userid[DB2SEC_MAX_USERID_LENGTH],
db2int32   *useridlen,
db2int32    useridtype,
char        usernamespace[DB2SEC_MAX_USERNAMESPACE_LENGTH],
db2int32   *usernamespacelen,
db2int32   *usernamespacetype,
const char *dbname,
db2int32    dbnamelen,
void      **token,
char      **errormsg,
db2int32   *errormsglen
);

SQL_API_RC (SQL_API_FN * db2secProcessServerPrincipalName)
(
const void *data,
```

```
gss_name_t *gssName,
char      **errormsg,
db2int32   *errormsglen
);

SQL_API_RC (SQL_API_FN * db2secGenerateInitialCred)
(
const char    *userid,
db2int32       useridlen,
const char    *usernamespace,
db2int32       usernamespacelen,
db2int32       usernamespacetype,
const char    *password,
db2int32       passwordlen,
const char    *newpassword,
db2int32       newpasswordlen,
const char    *dbname,
db2int32       dbnamelen,
gss_cred_id_t *pGSSCredHandle,
void         **initInfo,
char         **errormsg,
db2int32      *errormsglen
);

SQL_API_RC (SQL_API_FN * db2secFreeToken)
(
void      *token,
char     **errormsg,
db2int32  *errormsglen
);

SQL_API_RC (SQL_API_FN * db2secFreeErrormsg)
(
char *errormsg
);


SQL_API_RC (SQL_API_FN * db2secFreeInitInfo)
(
void      *initInfo,
char     **errormsg,
db2int32  *errormsglen
);

SQL_API_RC (SQL_API_FN * db2secClientAuthPluginTerm)
(
char     **errormsg,
db2int32  *errormsglen
);

/* GSS-API specific functions -- refer to db2secPlugin.h
   for parameter list*/

  OM_uint32 (SQL_API_FN * gss_init_sec_context )(<parameter list>);
  OM_uint32 (SQL_API_FN * gss_delete_sec_context )(<parameter list>);
  OM_uint32 (SQL_API_FN * gss_display_status )(<parameter list>);
  OM_uint32 (SQL_API_FN * gss_release_buffer )(<parameter list>);
  OM_uint32 (SQL_API_FN * gss_release_cred )(<parameter list>);
  OM_uint32 (SQL_API_FN * gss_release_name )(<parameter list>);
  }
```

You should use the db2secUseridPasswordClientAuthFunctions_1 structure if you are writing an user ID/password plug-in. If you are writing a GSS-API (including Kerberos) plug-in, you should use the db2secGssapiClientAuthFunctions_1 structure.

For the user ID/password plug-in library, you will need to implement the following server-side APIs:

- `db2secServerAuthPluginInit`

  The db2secServerAuthPluginInit API takes as input a pointer, `*logMessage_fn`, to the db2secLogMessage API, and a pointer, `*getConDetails_fn`, to the db2secGetConDetails API with the following prototypes:

```
SQL_API_RC (SQL_API_FN db2secLogMessage)
(
db2int32 level,
void    *data,
db2int32 length
);

SQL_API_RC (SQL_API_FN db2secGetConDetails)
(
db2int32    conDetailsVersion,
const void *pConDetails
);
```

  The db2secLogMessage API allows the plug-in to log messages to db2diag.log for debugging or informational purposes. The db2secGetConDetails API allows the plug-in to obtain details about the client that is trying to attempt to have a database connection. Both the db2secLogMessage API and db2secGetConDetails API are provided by the DB2 database system, so you do not need to implement them. The db2secGetConDetails API in turn, takes as its second parameter,`pConDetails`, a pointer to one of the following structures:

  db2sec_con_details_1:

```
typedef struct db2sec_con_details_1
{
  db2int32  clientProtocol;
  db2Uint32 clientIPAddress;
  db2Uint32 connect_info_bitmap;
  db2int32  dbnameLen;
  char      dbname[DB2SEC_MAX_DBNAME_LENGTH + 1];
} db2sec_con_details_1;
```

  db2sec_con_details_2:

```
typedef struct db2sec_con_details_2
{
  db2int32  clientProtocol;     /* See SQL_PROTOCOL_ in sqlenv.h */
  db2Uint32 clientIPAddress;    /* Set if protocol is TCPIP4     */
  db2Uint32 connect_info_bitmap;
  db2int32  dbnameLen;
  char dbname[DB2SEC_MAX_DBNAME_LENGTH + 1];
  db2Uint32 clientIP6Address[4];/* Set if protocol is TCPIP6     */
} db2sec_con_details_2;
```

  db2sec_con_details_3:

```
typedef struct db2sec_con_details_3
{
  db2int32 clientProtocol; /* See SQL_PROTOCOL_ in sqlenv.h */
  db2Uint32 clientIPAddress; /* Set if protocol is TCPIP4 */
  db2Uint32 connect_info_bitmap;
  db2int32 dbnameLen;
  char dbname[DB2SEC_MAX_DBNAME_LENGTH + 1];
  db2Uint32 clientIP6Address[4];/* Set if protocol is TCPIP6 */
  db2Uint32 clientPlatform; /* SQLM_PLATFORM_* from sqlmon.h */
  db2Uint32 _reserved[16];
} db2sec_con_details_3;
```

The possible values for conDetailsVersion are DB2SEC_CON_DETAILS_VERSION_1, DB2SEC_CON_DETAILS_VERSION_2, and DB2SEC_CON_DETAILS_VERSION_3 representing the version of the API.

Note: While using db2sec_con_details_1, db2sec_con_details_2, or db2sec_con_details_3, consider the following:

- Existing plugins that are using the db2sec_con_details_1 structure and the DB2SEC_CON_DETAILS_VERSION_1 value will continue to work as they did with Version 8.2 when calling the db2GetConDetails API. If this API is called on an IPv4 platform, the client IP address is returned in the clientIPAddress field of the structure. If this API is called on an IPv6 platform,a value of 0 is returned in the clientIPAddress field. To retrieve the client IP address on an IPv6 platform, the security plug-in code should be changed to use either the db2sec_con_details_2 structure and the DB2SEC_CON_DETAILS_VERSION_2 value, or the db2sec_con_details_3 structure and the DB2SEC_CON_DETAILS_VERSION_3 value .
- New plugins should use the db2sec_con_details_3 structure and the DB2SEC_CON_DETAILS_VERSION_3 value. If the db2secGetConDetails API is called on an IPv4 platform, the client IP address is returned in the clientIPAddress field of the db2sec_con_details_3 structure and if the API is called on an IPv6 platform the client IP address is returned in the clientIP6Address field of the db2sec_con_details_3 structure. The *clientProtocol* field of the connection details structure will be set to one of SQL_PROTOCOL_TCPIP (IPv4, with v1 of the structure), SQL_PROTOCOL_TCPIP4 (IPv4, with v2 of the structure) or SQL_PROTOCOL_TCPIP6 (IPv6, with v2 or v3 of the structure).
- The structure db2sec_con_details_3 is identical to the structure db2sec_con_details_2 except that it contains an additional field (*clientPlatform*) that identifies the client platform type (as reported by the communication layer) using platform type constants defined in sqlmon.h, such as SQLM_PLATFORM_AIX.

- db2secServerAuthPluginTerm
- db2secValidatePassword
- db2secGetAuthIDs
- db2secDoesAuthIDExist
- db2secFreeToken
- db2secFreeErrormsg
- The only API that must be resolvable externally is db2secServerAuthPluginInit. This API will take a void * parameter, which should be cast to either:

```
typedef struct db2secUseridPasswordServerAuthFunctions_1
{
db2int32 version;
db2int32 plugintype;

  /* parameter lists left blank for readability
    see above for parameters */
SQL_API_RC (SQL_API_FN * db2secValidatePassword)(<parameter list>);
SQL_API_RC (SQL_API_FN * db2secGetAuthIDs)(<parameter list);
SQL_API_RC (SQL_API_FN * db2secDoesAuthIDExist)(<parameter list>);
SQL_API_RC (SQL_API_FN * db2secFreeToken)(<parameter list>);
SQL_API_RC (SQL_API_FN * db2secFreeErrormsg)(<parameter list>);
SQL_API_RC (SQL_API_FN * db2secServerAuthPluginTerm)();
} userid_password_server_auth_functions;
```

or

```
typedef struct db2secGssapiServerAuthFunctions_1
{
db2int32 version;
db2int32 plugintype;
gss_buffer_desc serverPrincipalName;
gss_cred_id_t ServerCredHandle;
SQL_API_RC (SQL_API_FN * db2secGetAuthIDs)(<parameter list);
SQL_API_RC (SQL_API_FN * db2secDoesAuthIDExist)(<parameter list>);
SQL_API_RC (SQL_API_FN * db2secFreeErrormsg)(<parameter list>);
SQL_API_RC (SQL_API_FN * db2secServerAuthPluginTerm)();

/* GSS-API specific functions
refer to db2secPlugin.h for parameter list*/
OM_uint32 (SQL_API_FN * gss_accept_sec_context )(<parameter list>);
OM_uint32 (SQL_API_FN * gss_display_name )(<parameter list>);
OM_uint32 (SQL_API_FN * gss_delete_sec_context )(<parameter list>);
OM_uint32 (SQL_API_FN * gss_display_status )(<parameter list>);
OM_uint32 (SQL_API_FN * gss_release_buffer )(<parameter list>);
OM_uint32 (SQL_API_FN * gss_release_cred )(<parameter list>);
OM_uint32 (SQL_API_FN * gss_release_name )(<parameter list>);

} gssapi_server_auth_functions;
```

You should use the db2secUseridPasswordServerAuthFunctions_1 structure if you are writing an user ID/password plug-in. If you are writing a GSS-API (including Kerberos) plug-in, you should use the db2secGssapiServerAuthFunctions_1 structure.

db2secClientAuthPluginInit API - Initialize client authentication plug-in

Initialization API, for the client authentication plug-in, that the DB2 database manager calls immediately after loading the plug-in.

API and data structure syntax

```
SQL_API_RC SQL_API_FN db2secClientAuthPluginInit
                              ( db2int32 version,
                                void *client_fns,
                                db2secLogMessage *logMessage_fn,
                                char    **errormsg,
                                db2int32 *errormsglen );
```

db2secClientAuthPluginInit API parameters

version

Input. The highest version number of the API that the DB2 database manager currently supports. The DB2SEC_API_VERSION value (in db2secPlugin.h) contains the latest version number of the API that DB2 currently supports.

client_fns

Output. A pointer to memory provided by the DB2 database manager for a db2secGssapiClientAuthFunctions_<version_number> structure (also known as gssapi_client_auth_functions_<version_number>), if GSS-API authentication is used, or a db2secUseridPasswordClientAuthFunctions_<version_number> structure (also known as userid_password_client_auth_functions_<version_number>), if userid/password authentication is used. The db2secGssapiClientAuthFunctions_<version_number> structure and db2secUseridPasswordClientAuthFunctions_<version_number> structure respectively contain pointers to the APIs implemented for the GSS-API

authentication plug-in and userid/password authentication plug-in. In future versions of DB2, there might be different versions of the APIs, so the client_fns parameter is cast as a pointer to the gssapi_client_auth_functions_<version_number> structure corresponding to the version the plug-in has implemented.

The first parameter of the gssapi_client_auth_functions_<version_number> structure or the userid_password_client_auth_functions_<version_number> structure tells the DB2 database manager the version of the APIs that the plug-in has implemented.

Note: The casting is done only if the DB2 version is higher or equal to the version of the APIs that the plug-in has implemented.
Inside the gssapi_server_auth_functions_<version_number> or userid_password_server_auth_functions_<version_number> structure, the plugintype parameter should be set to one of DB2SEC_PLUGIN_TYPE_USERID_PASSWORD, DB2SEC_PLUGIN_TYPE_GSSAPI, or DB2SEC_PLUGIN_TYPE_KERBEROS. Other values can be defined in future versions of the API.

logMessage_fn
Input. A pointer to the db2secLogMessage API, which is implemented by the DB2 database manager. The db2secClientAuthPluginInit API can call the db2secLogMessage API to log messages to db2diag.log for debugging or informational purposes. The first parameter (level) of db2secLogMessage API specifies the type of diagnostic errors that will be recorded in the db2diag.log file and the last two parameters respectively are the message string and its length. The valid values for the first parameter of dbesecLogMessage API (defined in db2secPlugin.h) are:

- DB2SEC_LOG_NONE (0) No logging
- DB2SEC_LOG_CRITICAL (1) Severe Error encountered
- DB2SEC_LOG_ERROR (2) Error encountered
- DB2SEC_LOG_WARNING (3) Warning
- DB2SEC_LOG_INFO (4) Informational

The message text will show up in db2diag.log only if the value of the 'level' parameter of the db2secLogMessage API is less than or equal to the diaglevel database manager configuration parameter. For example, if you use the DB2SEC_LOG_INFO value, the message text will only appear in db2diag.log if the diaglevel database manager configuration parameter is set to 4.

errormsg
Output. A pointer to the address of an ASCII error message string allocated by the plug-in that can be returned in this parameter if the db2secClientAuthPluginInit API execution is not successful.

errormsglen
Output. A pointer to an integer that indicates the length in bytes of the error message string in errormsg parameter.

db2secClientAuthPluginTerm API - Clean up client authentication plug-in resources

Frees resources used by the client authentication plug-in.

This API is called by the DB2 database manager just before it unloads the client authentication plug-in. It should be implemented in a manner that it does a proper cleanup of any resources the plug-in library holds, for instance, free any memory allocated by the plug-in, close files that are still open, and close network connections. The plug-in is responsible for keeping track of these resources in order to free them. This API is not called on any Windows platform.

API and data structure syntax

```
SQL_API_RC ( SQL_API_FN *db2secClientAuthPluginTerm)
                      ( char     **errormsg,
                        db2int32 *errormsglen);
```

db2secClientAuthPluginTerm API parameters

errormsg
Output. A pointer to the address of an ASCII error message string allocated by the plug-in that can be returned in this parameter if the db2secClientAuthPluginTerm API execution is not successful.

errormsglen
Output. A pointer to an integer that indicates the length in bytes of the error message string in errormsg parameter.

db2secDoesAuthIDExist - Check if authentication ID exists

Determines if the authid represents an individual user (for example, whether the API can map the authid to an external user ID).

The API should return the value DB2SEC_PLUGIN_OK if it is successful - the authid is valid, DB2SEC_PLUGIN_INVALID_USERORGROUP if it is not valid, or DB2SEC_PLUGIN_USERSTATUSNOTKNOWN if the authid existence cannot be determined.

API and data structure syntax

```
SQL_API_RC ( SQL_API_FN *db2secDoesAuthIDExist)
                      ( const char *authid,
                        db2int32 authidlen,
                        char       **errormsg,
                        db2int32 *errormsglen );
```

db2secDoesAuthIDExist API parameters

authid Input. The authid to validate. This is upper-cased, with no trailing blanks.

authidlen
Input. Length in bytes of the authid parameter value.

errormsg
Output. A pointer to the address of an ASCII error message string allocated by the plug-in that can be returned in this parameter if the db2secDoesAuthIDExist API execution is not successful.

errormsglen
Output. A pointer to an integer that indicates the length of the error message string in errormsg parameter.

db2secFreeInitInfo API - Clean up resources held by the db2secGenerateInitialCred

Frees any resources allocated by the db2secGenerateInitialCred API. This can include, for example, handles to underlying mechanism contexts or a credential cache created for the GSS-API credential cache.

API and data structure syntax

```
SQL_API_RC ( SQL_API_FN *db2secFreeInitInfo)
                     ( void *initinfo,
                       char     **errormsg,
                       db2int32 *errormsglen);
```

db2secFreeInitInfo API parameters

initinfo
Input. A pointer to data that is not known to the DB2 database manager. The plug-in can use this memory to maintain a list of resources that are allocated in the process of generating the credential handle. These resources are freed by calling this API.

errormsg
Output. A pointer to the address of an ASCII error message string allocated by the plug-in that can be returned in this parameter if the db2secFreeInitInfo API execution is not successful.

errormsglen
Output. A pointer to an integer that indicates the length in bytes of the error message string in errormsg parameter.

db2secFreeToken API - Free memory held by token

Frees the memory held by a token. This API is called by the DB2 database manager when it no longer needs the memory held by the token parameter.

API and data structure syntax

```
SQL_API_RC ( SQL_API_FN *db2secFreeToken)
                     ( void *token,
                       char       **errormsg,
                       db2int32 *errormsglen );
```

db2secFreeToken API parameters

token Input. Pointer to the memory to be freed.

errormsg
Output. A pointer to the address of an ASCII error message string allocated by the plug-in that can be returned in this parameter if the db2secFreeToken API execution is not successful.

errormsglen
Output. A pointer to an integer that indicates the length in bytes of the error message string in errormsg parameter.

db2secGenerateInitialCred API - Generate initial credentials

Obtains the initial GSS-API credentials based on the user ID and password that are passed in. For Kerberos, this is the ticket-granting ticket (TGT). The credential handle that is returned in pGSSCredHandle parameter is the handle that is used with the gss_init_sec_context API and must be either an INITIATE or BOTH credential. The db2secGenerateInitialCred API is only called when a user ID, and

possibly a password are supplied. Otherwise, the DB2 database manager specifies the value GSS_C_NO_CREDENTIAL when calling the gss_init_sec_context API to signify that the default credential obtained from the current login context is to be used.

API and data structure syntax

```
SQL_API_RC ( SQL_API_FN *db2secGenerateInitialCred)
                       ( const char *userid,
                         db2int32 useridlen,
                         const char *usernamespace,
                         db2int32 usernamespacelen,
                         db2int32 usernamespacetype,
                         const char *password,
                         db2int32 passwordlen,
                         const char *newpassword,
                         db2int32 newpasswordlen,
                         const char *dbname,
                         db2int32 dbnamelen,
                         gss_cred_id_t *pGSSCredHandle,
                         void          **InitInfo,
                         char          **errormsg,
                         db2int32 *errormsglen );
```

db2secGenerateInitialCred API parameters

userid Input. The user ID whose password is to be verified on the database server.

useridlen
Input. Length in bytes of the userid parameter value.

usernamespace
Input. The namespace from which the user ID was obtained.

usernamespacelen
Input. Length in bytes of the usernamespace parameter value.

usernamespacetype
Input. The type of namespace.

password
Input. The password to be verified.

passwordlen
Input. Length in bytes of the password parameter value.

newpassword
Input. A new password if the password is to be changed. If no change is requested, the newpassword parameter is set to NULL. If it is not NULL, the API should validate the old password before setting the password to its new value. The API does not have to honour a request to change the password, but if it does not, it should immediately return with the return value DB2SEC_PLUGIN_CHANGEPASSWORD_NOTSUPPORTED without validating the old password.

newpasswordlen
Input. Length in bytes of the newpassword parameter value.

dbname
Input. The name of the database being connected to. The API is free to ignore this parameter, or the API can return the value

DB2SEC_PLUGIN_CONNECTION_DISALLOWED if it has a policy of restricting access to certain databases to users who otherwise have valid passwords.

dbnamelen
Input. Length in bytes of the dbname parameter value.

pGSSCredHandle
Output. Pointer to the GSS-API credential handle.

InitInfo
Output. A pointer to data that is not known to DB2. The plug-in can use this memory to maintain a list of resources that are allocated in the process of generating the credential handle. The DB2 database manager calls the db2secFreeInitInfo API at the end of the authentication process, at which point these resources are freed. If the db2secGenerateInitialCred API does not need to maintain such a list, then it should return NULL.

errormsg
Output. A pointer to the address of an ASCII error message string allocated by the plug-in that can be returned in this parameter if the db2secGenerateInitialCred API execution is not successful.

Note: For this API, error messages should not be created if the return value indicates a bad user ID or password. An error message should only be returned if there is an internal error in the API that prevented it from completing properly.

errormsglen
Output. A pointer to an integer that indicates the length in bytes of the error message string in errormsg parameter.

db2secGetAuthIDs API - Get authentication IDs

Returns an SQL authid for an authenticated user. This API is called during database connections for both user ID/password and GSS-API authentication methods.

API and data structure syntax

```
SQL_API_RC ( SQL_API_FN *db2secGetAuthIDs)
                      ( const char *userid,
                        db2int32 useridlen,
                        const char *usernamespace,
                        db2int32 usernamespacelen,
                        db2int32 usernamespacetype,
                        const char *dbname,
                        db2int32 dbnamelen,
                        void      **token,
                        char SystemAuthID[DB2SEC_MAX_AUTHID_LENGTH],
                        db2int32 *SystemAuthIDlen,
                        char InitialSessionAuthID[DB2SEC_MAX_AUTHID_LENGTH],
                        db2int32 *InitialSessionAuthIDlen,
                        char username[DB2SEC_MAX_USERID_LENGTH],
                        db2int32 *usernamelen,
                        db2int32 *initsessionidtype,
                        char      **errormsg,
                        db2int32 *errormsglen );
```

db2secGetAuthIDs API parameters

userid Input. The authenticated user. This is usually not used for GSS-API authentication unless a trusted context is defined to permit switch user

operations without authentication. In those situations, the user name provided for the switch user request is passed in this parameter.

useridlen
Input. Length in bytes of the userid parameter value.

usernamespace
Input. The namespace from which the user ID was obtained.

usernamespacelen
Input. Length in bytes of the usernamespace parameter value.

usernamespacetype
Input. Namespacetype value. currently, the only supported namespace type value is DB2SEC_NAMESPACE_SAM_COMPATIBLE (corresponds to a username style like domain\myname).

dbname
Input. The name of the database being connected to. The API can ignore this, or it can return differing authids when the same user connects to different databases. This parameter can be NULL.

dbnamelen
Input. Length in bytes of the dbname parameter value. This parameter is set to 0 if dbname parameter is NULL.

token Input or output. Data that the plug-in might pass to the db2secGetGroupsForUser API. For GSS-API, this is a context handle (gss_ctx_id_t). Ordinarily, token is an input-only parameter and its value is taken from the db2secValidatePassword API. It can also be an output parameter when authentication is done on the client and therefore db2secValidatePassword API is not called. In environments where a trusted context is defined that allows switch user operations without authentication, the db2secGetAuthIDs API must be able to accommodate receiving a NULL value for this token parameter and be able to derive a system authorization ID based on the userid and useridlen input parameters above.

SystemAuthID
Output. The system authorization ID that corresponds to the ID of the authenticated user. The size is 255 bytes, but the DB2 database manager currently uses only up to (and including) 30 bytes.

SystemAuthIDlen
Output. Length in bytes of the SystemAuthID parameter value.

InitialSessionAuthID
Output. Authid used for this connection session. This is usually the same as the SystemAuthID parameter but can be different in some situations, for instance, when issuing a SET SESSION AUTHORIZATION statement. The size is 255 bytes, but the DB2 database manager currently uses only up to (and including) 30 bytes.

InitialSessionAuthIDlen
Output. Length in bytes of the InitialSessionAuthID parameter value.

username
Output. A username corresponding to the authenticated user and authid. This will only be used for auditing and will be logged in the ″User ID″ field in the audit record for CONNECT statement. If the API does not fill in the username parameter, the DB2 database manager copies it from the userid.

usernamelen
Output. Length in bytes of the username parameter value.

initsessionidtype
Output. Session authid type indicating whether or not the InitialSessionAuthid parameter is a role or an authid. The API should return one of the following values (defined in db2secPlugin.h):

- DB2SEC_ID_TYPE_AUTHID (0)
- DB2SEC_ID_TYPE_ROLE (1)

errormsg
Output. A pointer to the address of an ASCII error message string allocated by the plug-in that can be returned in this parameter if the db2secGetAuthIDs API execution is not successful.

errormsglen
Output. A pointer to an integer that indicates the length in bytes of the error message string in errormsg parameter.

db2secGetDefaultLoginContext API - Get default login context

Determines the user associated with the default login context, in other words, determines the DB2 authid of the user invoking a DB2 command without explicitly specifying a user ID (either an implicit authentication to a database, or a local authorization). This API must return both an authid and a user ID.

API and data structure syntax

```
SQL_API_RC ( SQL_API_FN *db2secGetDefaultLoginContext)
                      ( char authid[DB2SEC_MAX_AUTHID_LENGTH],
                        db2int32 *authidlen,
                        char userid[DB2SEC_MAX_USERID_LENGTH],
                        db2int32 *useridlen,
                        db2int32 useridtype,
                        char usernamespace[DB2SEC_MAX_USERNAMESPACE_LENGTH],
                        db2int32 *usernamespacelen,
                        db2int32 *usernamespacetype,
                        const char *dbname,
                        db2int32 dbnamelen,
                        void      **token,
                        char      **errormsg,
                        db2int32 *errormsglen );
```

db2secGetDefaultLoginContext API parameters

authid Output. The parameter in which the authid should be returned. The returned value must conform to DB2 authid naming rules, or the user will not be authorized to perform the requested action.

authidlen
Output. Length in bytes of the authid parameter value.

userid Output. The parameter in which the user ID associated with the default login context should be returned.

useridlen
Output. Length in bytes of the userid parameter value.

useridtype
Input. Indicates if the real or effective user ID of the process is being specified. On Windows, only the real user ID exists. On UNIX and Linux, the real user ID and effective user ID can be different if the `uid` user ID for

the application is different than the ID of the user executing the process. Valid values for the userid parameter (defined in db2secPlugin.h) are:

DB2SEC_PLUGIN_REAL_USER_NAME
Indicates that the real user ID is being specified.

DB2SEC_PLUGIN_EFFECTIVE_USER_NAME
Indicates that the effective user ID is being specified.

Note: Some plug-in implementations might not distinguish between the real and effective userid. In particular, a plug-in that does not use the UNIX or Linux identity of the user to establish the DB2 authorization ID can safely ignore this distinction.

usernamespace
Output. The namespace of the user ID.

usernamespacelen
Output. Length in bytes of the usernamespace parameter value. Under the limitation that the usernamespacetype parameter must be set to the value DB2SEC_NAMESPACE_SAM_COMPATIBLE (defined in db2secPlugin.h), the maximum length currently supported is 15 bytes.

usernamespacetype
Output. Namespacetype value. Currently, the only supported namespace type is DB2SEC_NAMESPACE_SAM_COMPATIBLE (corresponds to a username style like domain\myname).

dbname
Input. Contains the name of the database being connected to, if this call is being used in the context of a database connection. For local authorization actions or instance attachments, this parameter is set to NULL.

dbnamelen
Input. Length in bytes of the dbname parameter value.

token Output. This is a pointer to data allocated by the plug-in that it might pass to subsequent authentication calls in the plug-in, or possibly to the group retrieval plug-in. The structure of this data is determined by the plug-in writer.

errormsg
Output. A pointer to the address of an ASCII error message string allocated by the plug-in that can be returned in this parameter if the db2secGetDefaultLoginContext API execution is not successful.

errormsglen
Output. A pointer to an integer that indicates the length in bytes of the error message string in errormsg parameter.

db2secProcessServerPrincipalName API - Process service principal name returned from server

Processes the service principal name returned from the server and returns the principal name in the gss_name_t internal format to be used with the gss_init_sec_context API. The db2secProcessServerPrincipalName API also processes the service principal name cataloged with the database directory when Kerberos authentication is used. Ordinarily, this conversion uses the gss_import_name API. After the context is established, the gss_name_t object is freed through the call to gss_release_name API. The db2secProcessServerPrincipalName API returns the value DB2SEC_PLUGIN_OK if

gssName parameter points to a valid GSS name; a DB2SEC_PLUGIN_BAD_PRINCIPAL_NAME error code is returned if the principal name is invalid.

API and data structure syntax

```
SQL_API_RC ( SQL_API_FN *db2secProcessServerPrincipalName)
                      ( const char *name,
                        db2int32 namelen,
                        gss_name_t *gssName,
                        char       **errormsg,
                        db2int32 *errormsglen );
```

db2secProcessServerPrincipalName API parameters

name Input. Text name of the service principal in GSS_C_NT_USER_NAME format; for example, service/host@REALM.

namelen
Input. Length in bytes of the name parameter value.

gssName
Output. Pointer to the output service principal name in the GSS-API internal format.

errormsg
Output. A pointer to the address of an ASCII error message string allocated by the plug-in that can be returned in this parameter if the db2secProcessServerPrincipalName API execution is not successful.

errormsglen
Output. A pointer to an integer that indicates the length in bytes of the error message string in errormsg parameter.

db2secRemapUserid API - Remap user ID and password

This API is called by the DB2 database manager on the client side to remap a given user ID and password (and possibly new password and usernamespace) to values different from those given at connect time. The DB2 database manager only calls this API if a user ID and a password are supplied at connect time. This prevents a plug-in from remapping a user ID by itself to a user ID/password pair. This API is optional and is not called if it is not provided or implemented by the security plug-in.

API and data structure syntax

```
SQL_API_RC ( SQL_API_FN *db2secRemapUserid)
                      ( char userid[DB2SEC_MAX_USERID_LENGTH],
                        db2int32 *useridlen,
                        char usernamespace[DB2SEC_MAX_USERNAMESPACE_LENGTH],
                        db2int32 *usernamespacelen,
                        db2int32 *usernamespacetype,
                        char password[DB2SEC_MAX_PASSWORD_LENGTH],
                        db2int32 *passwordlen,
                        char newpasswd[DB2SEC_MAX_PASSWORD_LENGTH],
                        db2int32 *newpasswdlen,
                        const char *dbname,
                        db2int32 dbnamelen,
                        char      **errormsg,
                        db2int32 *errormsglen);
```

db2secRemapUserid API parameters

userid Input or output. The user ID to be remapped. If there is an input user ID

value, then the API must provide an output user ID value that can be the same or different from the input user ID value. If there is no input user ID value, then the API should not return an output user ID value.

useridlen
Input or output. Length in bytes of the userid parameter value.

usernamespace
Input or output. The namespace of the user ID. This value can optionally be remapped. If no input parameter value is specified, but an output value is returned, then the usernamespace will only be used by the DB2 database manager for CLIENT type authentication and is disregarded for other authentication types.

usernamespacelen
Input or output. Length in bytes of the usernamespace parameter value. Under the limitation that the usernamespacetype parameter must be set to the value DB2SEC_NAMESPACE_SAM_COMPATIBLE (defined in db2secPlugin.h), the maximum length currently supported is 15 bytes.

usernamespacetype
Input or output. Old and new namespacetype value. Currently, the only supported namespace type value is DB2SEC_NAMESPACE_SAM_COMPATIBLE (corresponds to a username style like domain\myname).

password
Input or output. As an input, it is the password that is to be remapped. As an output it is the remapped password. If an input value is specified for this parameter, the API must be able to return an output value that differs from the input value. If no input value is specified, the API must not return an output password value.

passwordlen
Input or output. Length in bytes of the password parameter value.

newpasswd
Input or output. As an input, it is the new password that is to be set. As an output it is the confirmed new password.

Note: This is the new password that is passed by the DB2 database manager into the newpassword parameter of the db2secValidatePassword API on the client or the server (depending on the value of the authentication database manager configuration parameter). If a new password was passed as input, then the API must be able to return an output value and it can be a different new password. If there is no new password passed in as input, then the API should not return an output new password.

newpasswdlen
Input or output. Length in bytes of the newpasswd parameter value.

dbname
Input. Name of the database to which the client is connecting.

dbnamelen
Input. Length in bytes of the dbname parameter value.

errormsg
Output. A pointer to the address of an ASCII error message string allocated

by the plug-in that can be returned in this parameter if the db2secRemapUserid API execution is not successful.

errormsglen
Output. A pointer to an integer that indicates the length in bytes of the error message string in errormsg parameter.

db2secServerAuthPluginInit - Initialize server authentication plug-in

Initialization API, for the server authentication plug-in, that the DB2 database manager calls immediately after loading the plug-in. In the case of GSS-API, the plug-in is responsible for filling in the server's principal name in the serverPrincipalName parameter inside the gssapi_server_auth_functions structure at initialization time and providing the server's credential handle in the serverCredHandle parameter inside the gssapi_server_auth_functions structure. The freeing of the memory allocated to hold the principal name and the credential handle must be done by the db2secServerAuthPluginTerm API by calling the gss_release_name and gss_release_cred APIs.

API and data structure syntax

```
SQL_API_RC SQL_API_FN db2secServerAuthPluginInit
                            ( db2int32 version,
                              void *server_fns,
                              db2secGetConDetails *getConDetails_fn,
                              db2secLogMessage *logMessage_fn,
                              char               **errormsg,
                              db2int32 *errormsglen );
```

db2secServerAuthPluginInit API parameters

version
Input. The highest version number of the API that the DB2 database manager currently supports. The DB2SEC_API_VERSION value (in db2secPlugin.h) contains the latest version number of the API that the DB2 database manager currently supports.

server_fns
Output. A pointer to memory provided by the DB2 database manager for a db2secGssapiServerAuthFunctions_<version_number> structure (also known as gssapi_server_auth_functions_<version_number>), if GSS-API authentication is used, or a db2secUseridPasswordServerAuthFunctions_<version_number> structure (also known as userid_password_server_auth_functions_<version_number>), if userid/password authentication is used. The db2secGssapiServerAuthFunctions_<version_number> structure and db2secUseridPasswordServerAuthFunctions_<version_number> structure respectively contain pointers to the APIs implemented for the GSS-API authentication plug-in and userid/password authentication plug-in.

The server_fns parameter is cast as a pointer to the gssapi_server_auth_functions_<version_number> structure corresponding to the version the plug-in has implemented. The first parameter of the gssapi_server_auth_functions_<version_number> structure or the userid_password_server_auth_functions_<version_number> structure tells theDB2 database manager the version of the APIs that the plug-in has implemented.

Note: The casting is done only if the DB2 version is higher or equal to the version of the APIs that the plug-in has implemented.
Inside the gssapi_server_auth_functions_<version_number> or userid_password_server_auth_functions_<version_number> structure, the plugintype parameter should be set to one of DB2SEC_PLUGIN_TYPE_USERID_PASSWORD, DB2SEC_PLUGIN_TYPE_GSSAPI, or DB2SEC_PLUGIN_TYPE_KERBEROS. Other values can be defined in future versions of the API.

getConDetails_fn

Input. Pointer to the db2secGetConDetails API, which is implemented by DB2. The db2secServerAuthPluginInit API can call the db2secGetConDetails API in any one of the other authentication APIs to obtain details related to the database connection. These details include information about the communication mechanism associated with the connection (such as the IP address, in the case of TCP/IP), which the plug-in writer might need to reference when making authentication decisions. For example, the plug-in could disallow a connection for a particular user, unless that user is connecting from a particular IP address. The use of the db2secGetConDetails API is optional.

If the db2secGetConDetails API is called in a situation not involving a database connection, it returns the value DB2SEC_PLUGIN_NO_CON_DETAILS, otherwise, it returns 0 on success.

The db2secGetConDetails API takes two input parameters; pConDetails, which is a pointer to the db2sec_con_details_<version_number> structure, and conDetailsVersion, which is a version number indicating which db2sec_con_details structure to use. Possible values are DB2SEC_CON_DETAILS_VERSION_1 when db2sec_con_details1 is used or DB2SEC_CON_DETAILS_VERSION_2 when db2sec_con_details2. The recommended version number to use is DB2SEC_CON_DETAILS_VERSION_2.

Upon a successful return, the db2sec_con_details structure (either db2sec_con_details1 or db2sec_con_details2) will contain the following information:

- The protocol used for the connection to the server. The listing of protocol definitions can be found in the file sqlenv.h (located in the include directory) (SQL_PROTOCOL_*). This information is filled out in the clientProtocol parameter.
- The TCP/IP address of the inbound connect to the server if the clientProtocol is SQL_PROTOCOL_TCPIP or SQL_PROTOCOL_TCPIP4. This information is filled out in the clientIPAddress parameter.
- The database name the client is attempting to connect to. This will not be set for instance attachments. This information is filled out in the dbname and dbnameLen parameters.
- A connection information bit-map that contains the same details as documented in the connection_details parameter of the db2secValidatePassword API. This information is filled out in the connect_info_bitmap parameter.
- The TCP/IP address of the inbound connect to the server if the clientProtocol is SQL_PROTOCOL_TCPIP6. This information is filled out in the clientIP6Address parameter and it is only available if DB2SEC_CON_DETAILS_VERSION_2 is used for db2secGetConDetails API call.

logMessage_fn

Input. A pointer to the db2secLogMessage API, which is implemented by the DB2 database manager. The db2secClientAuthPluginInit API can call the db2secLogMessage API to log messages to db2diag.log for debugging or informational purposes. The first parameter (level) of db2secLogMessage API specifies the type of diagnostic errors that will be recorded in the db2diag.log file and the last two parameters respectively are the message string and its length. The valid values for the first parameter of dbesecLogMessage API (defined in db2secPlugin.h) are:

DB2SEC_LOG_NONE (0)
No logging

DB2SEC_LOG_CRITICAL (1)
Severe Error encountered

DB2SEC_LOG_ERROR (2)
Error encountered

DB2SEC_LOG_WARNING (3)
Warning

DB2SEC_LOG_INFO (4)
Informational

The message text will show up in db2diag.log only if the value of the 'level' parameter of the db2secLogMessage API is less than or equal to the diaglevel database manager configuration parameter.

So for example, if you use the DB2SEC_LOG_INFO value, the message text will only show up in the db2diag.log if the diaglevel database manager configuration parameter is set to 4.

errormsg

Output. A pointer to the address of an ASCII error message string allocated by the plug-in that can be returned in this parameter if the db2secServerAuthPluginInit API execution is not successful.

errormsglen

Output. A pointer to an integer that indicates the length in bytes of the error message string in errormsg parameter.

db2secServerAuthPluginTerm API - Clean up server authentication plug-in resources

Frees resources used by the server authentication plug-in. This API is called by the DB2 database manager just before it unloads the server authentication plug-in. It should be implemented in a manner that it does a proper cleanup of any resources the plug-in library holds, for instance, free any memory allocated by the plug-in, close files that are still open, and close network connections. The plug-in is responsible for keeping track of these resources in order to free them. This API is not called on any Windows platform.

API and data structure syntax

```
SQL_API_RC ( SQL_API_FN *db2secServerAuthPluginTerm)
                      ( char     **errormsg,
                        db2int32 *errormsglen );
```

db2secServerAuthPluginTerm API parameters

errormsg
Output. A pointer to the address of an ASCII error message string allocated by the plug-in that can be returned in this parameter if the db2secServerAuthPluginTerm API execution is not successful.

errormsglen
Output. A pointer to an integer that indicates the length in bytes of the error message string in errormsg parameter.

db2secValidatePassword API - Validate password

Provides a method for performing user ID and password style authentication during a database connect operation.

Note: When the API is run on the client side, the API code is run with the privileges of the user executing the CONNECT statement. This API will only be called on the client side if the authentication configuration parameter is set to CLIENT.

When the API is run on the server side, the API code is run with the privileges of the instance owner.

The plug-in writer should take the above into consideration if authentication requires special privileges (such as root level system access on UNIX).

This API must return the value DB2SEC_PLUGIN_OK (success) if the password is valid, or an error code such as DB2SEC_PLUGIN_BADPWD if the password is invalid.

API and data structure syntax

```
SQL_API_RC ( SQL_API_FN *db2secValidatePassword)
                       ( const char *userid,
                         db2int32 useridlen,
                         const char *usernamespace,
                         db2int32 usernamespacelen,
                         db2int32 usernamespacetype,
                         const char *password,
                         db2int32 passwordlen,
                         const char *newpasswd,
                         db2int32 newpasswdlen,
                         const char *dbname,
                         db2int32 dbnamelen,
                         db2Uint32 connection_details,
                         void      **token,
                         char      **errormsg,
                         db2int32 *errormsglen );
```

db2secValidatePassword API parameters

userid Input. The user ID whose password is to be verified.

useridlen
Input. Length in bytes of the userid parameter value.

usernamespace
Input. The namespace from which the user ID was obtained.

usernamespacelen
Input. Length in bytes of the usernamespace parameter value.

usernamespacetype
Input. The type of namespace. Valid values for the usernamespacetype parameter (defined in db2secPlugin.h) are:

- `DB2SEC_NAMESPACE_SAM_COMPATIBLE` Corresponds to a username style like domain\myname
- `DB2SEC_NAMESPACE_USER_PRINCIPAL` Corresponds to a username style like myname@domain.ibm.com

Currently, the DB2 database system only supports the value DB2SEC_NAMESPACE_SAM_COMPATIBLE. When the user ID is not available, the usernamespacetype parameter is set to the value DB2SEC_USER_NAMESPACE_UNDEFINED (defined in db2secPlugin.h).

password
Input. The password to be verified.

passwordlen
Input. Length in bytes of the password parameter value.

newpasswd
Input. A new password, if the password is to be changed. If no change is requested, this parameter is set to NULL. If this parameter is not NULL, the API should validate the old password before changing it to the new password. The API does not have to fulfill a request to change the password, but if it does not, it should immediately return with the return value DB2SEC_PLUGIN_CHANGEPASSWORD_NOTSUPPORTED without validating the old password.

newpasswdlen
Input. Length in bytes of the newpasswd parameter value.

dbname
Input. The name of the database being connected to. The API is free to ignore the dbname parameter, or it can return the value DB2SEC_PLUGIN_CONNECTIONREFUSED if it has a policy of restricting access to certain databases to users who otherwise have valid passwords. This parameter can be NULL.

dbnamelen
Input. Length in bytes of the dbname parameter value. This parameter is set to 0 if dbname parameter is NULL.

connection_details
Input. A 32-bit parameter of which 3 bits are currently used to store the following information:

- The rightmost bit indicates whether the source of the user ID is the default from the db2secGetDefaultLoginContext API, or was explicitly provided during the connect.
- The second-from-right bit indicates whether the connection is local (using Inter Process Communication (IPC) or a connect from one of the nodes in the db2nodes.cfg in the partitioned database environment), or remote (through a network or loopback). This gives the API the ability to decide whether clients on the same machine can connect to the DB2 server without a password. Due to the default operating-system-based user ID/password plugin, local connections are permitted without a password from clients on the same machine (assuming the user has connect privileges).
- The third-from-right bit indicates whether the DB2 database manager is calling the API from the server side or client side.

The bit values are defined in db2secPlugin.h:

- `DB2SEC_USERID_FROM_OS` (0x00000001) Indicates that the user ID is obtained from OS and not explicitly given on the connect statement.
- `DB2SEC_CONNECTION_ISLOCAL` (0x00000002) Indicates a local connection.
- `DB2SEC_VALIDATING_ON_SERVER_SIDE` (0x0000004) Indicates whether the DB2 database manager is calling from the server side or client side to validate password. If this bit value is set, then the DB2 database manager is calling from server side; otherwise, it is calling from the client side.

The DB2 database system default behavior for an implicit authentication is to allow the connection without any password validation. However, plug-in developers have the option to disallow implicit authentication by returning a DB2SEC_PLUGIN_BADPASSWORD error.

token Input. A pointer to data which can be passed as a parameter to subsequent API calls during the current connection. Possible APIs that might be called include db2secGetAuthIDs API and db2secGetGroupsForUser API.

errormsg
Output. A pointer to the address of an ASCII error message string allocated by the plug-in that can be returned in this parameter if the db2secValidatePassword API execution is not successful.

errormsglen
Output. A pointer to an integer that indicates the length in bytes of the error message string in errormsg parameter.

Required APIs and definitions for GSS-API authentication plug-ins

Following is a complete list of GSS-APIs required for the DB2 security plug-in interface.

The supported APIs follow these specifications: *Generic Security Service Application Program Interface, Version 2* (IETF RFC2743) and *Generic Security Service API Version 2: C-Bindings* (IETF RFC2744). Before implementing a GSS-API based plug-in, you should have a complete understanding of these specifications.

Table 39. Required APIs and Definitions for GSS-API authentication plug-ins

Name		Description
Client-side APIs	`gss_init_sec_context`	Initiate a security context with a peer application.
Server-side APIs	`gss_accept_sec_context`	Accept a security context initiated by a peer application.
Server-side APIs	`gss_display_name`	Convert an internal format name to text.
Common APIs	`gss_delete_sec_context`	Delete an established security context.
Common APIs	`gss_display_status`	Obtain the text error message associated with a GSS-API status code.
Common APIs	`gss_release_buffer`	Delete a buffer.
Common APIs	`gss_release_cred`	Release local data structures associated with a GSS-API credential.
Common APIs	`gss_release_name`	Delete internal format name.
Required definitions	`GSS_C_DELEG_FLAG`	Requests delegation.

Table 39. Required APIs and Definitions for GSS-API authentication plug-ins (continued)

Name		Description
Required definitions	GSS_C_EMPTY_BUFFER	Signifies that the gss_buffer_desc does not contain any data.
Required definitions	GSS_C_GSS_CODE	Indicates a GSS major status code.
Required definitions	GSS_C_INDEFINITE	Indicates that the mechanism does not support context expiration.
Required definitions	GSS_C_MECH_CODE	Indicates a GSS minor status code.
Required definitions	GSS_C_MUTUAL_FLAG	Mutual authentication requested.
Required definitions	GSS_C_NO_BUFFER	Signifies that the gss_buffer_t variable does not point to a valid gss_buffer_desc structure.
Required definitions	GSS_C_NO_CHANNEL_BINDINGS	No communication channel bindings.
Required definitions	GSS_C_NO_CONTEXT	Signifies that the gss_ctx_id_t variable does not point to a valid context.
Required definitions	GSS_C_NO_CREDENTIAL	Signifies that gss_cred_id_t variable does not point to a valid credential handle.
Required definitions	GSS_C_NO_NAME	Signifies that the gss_name_t variable does not point to a valid internal name.
Required definitions	GSS_C_NO_OID	Use default authentication mechanism.
Required definitions	GSS_C_NULL_OID_SET	Use default mechanism.
Required definitions	GSS_S_COMPLETE	API completed successfully.
Required definitions	GSS_S_CONTINUE_NEEDED	Processing is not complete and the API must be called again with the reply token received from the peer.

Restrictions for GSS-API authentication plug-ins

The following is a list of restrictions for GSS-API authentication plug-ins.

- The default security mechanism is always assumed; therefore, there is no OID consideration.
- The only GSS services requested in gss_init_sec_context() are mutual authentication and delegation. The DB2 database manager always requests a ticket for delegation, but does not use that ticket to generate a new ticket.
- Only the default context time is requested.
- Context tokens from gss_delete_sec_context() are not sent from the client to the server and vice-versa.
- Anonymity is not supported.
- Channel binding is not supported
- If the initial credentials expire, the DB2 database manager does not automatically renew them.
- The GSS-API specification stipulates that even if gss_init_sec_context() or gss_accept_sec_context() fail, either function must return a token to send to

the peer. However, because of DRDA limitations, the DB2 database manager only sends a token if `gss_init_sec_context()` fails and generates a token on the first call.

Chapter 9. Audit facility record layouts

When an audit record is extracted from the audit log, each record has one of the formats shown in the following tables. Each table is preceded by a sample record.

The description of each item in the record is shown one row at a time in the associated table. Each item is shown in the table in the same order as it is output in the delimited file after the extract operation.

Note:

1. Not all fields in the sample records will have values.
2. Some fields such as "Access Attempted" are stored in the delimited ASCII format as bit maps. In this flat report file, however, these fields appear as a set of strings representing the bit map values.

Audit record object types

The following table shows for each audit record object type whether it can generate CHECKING, OBJMAINT, and SECMAINT events.

Table 40. Audit Record Object Types Based on Audit Events

Object type	CHECKING events	OBJMAINT events	SECMAINT events
ACCESS_RULE			X
ALIAS	X	X	
ALL	X		
AUDIT_POLICY	X	X	
BUFFERPOOL	X	X	
CHECK_CONSTRAINT		X	
DATABASE	X		X
DATA TYPE		X	
EVENT_MONITOR	X	X	
FOREIGN_KEY		X	
FUNCTION	X	X	X
FUNCTION MAPPING	X	X	
GLOBAL_VARIABLE	X	X	X
HISTOGRAM TEMPLATE	X	X	
INDEX	X	X	X
INDEX EXTENSION		X	
INSTANCE	X		
JAR_FILE		X	
METHOD_BODY	X	X	X
NICKNAME	X	X	X
NODEGROUP	X	X	
NONE	X	X	X
OPTIMIZATION PROFILE	X		

Table 40. Audit Record Object Types Based on Audit Events (continued)

Object type	CHECKING events	OBJMAINT events	SECMAINT events
PACKAGE	X	X	X
PACKAGE CACHE	X		
PRIMARY_KEY		X	
REOPT_VALUES	X		
ROLE	X	X	X
SCHEMA	X	X	X
SECURITY LABEL		X	X
SECURITY LABEL COMPONENT		X	
SECURITY POLICY		X	X
SEQUENCE	X	X	
SERVER	X	X	X
SERVER OPTION	X	X	
SERVICE CLASS	X	X	
STORED_PROCEDURE	X	X	X
SUMMARY TABLES	X	X	X
TABLE	X	X	X
TABLESPACE	X	X	X
THRESHOLD	X	X	
TRIGGER		X	
TRUSTED CONTEXT	X	X	X
TYPE MAPPING	X	X	
TYPE&TRANSFORM	X	X	
UNIQUE_CONSTRAINT		X	
USER MAPPING	X	X	
VIEW	X	X	X
WORK ACTION SET	X	X	
WORK CLASS SET	X	X	
WORKLOAD	X	X	X
WRAPPER	X	X	
XSR object	X	X	X

Audit record layout for AUDIT events

The following table shows the layout of the audit record for AUDIT events.

Sample audit record:

```
timestamp=2007-04-10-08.29.52.000001;
category=AUDIT;
audit event=START;
event correlator=0;
event status=0;
```

```
userid=newton;
authid=NEWTON;
application id=*LOCAL_APPLICATION;
application name=db2audit.exe;
```

Table 41. Audit Record Layout for AUDIT Events

NAME	FORMAT	DESCRIPTION
Timestamp	CHAR(26)	Date and time of the audit event.
Category	CHAR(8)	Category of audit event. Possible values are: AUDIT
Audit Event	VARCHAR(32)	Specific Audit Event. Possible values include: • CONFIGURE • DB2AUD • EXTRACT • FLUSH • PRUNE (not generated in Version 9.5, and later). • START • STOP • UPDATE_ADMIN_CFG • ARCHIVE • LIST_LOGS • CREATE_AUDIT_POLICY • ALTER_AUDIT_POLICY • DROP_AUDIT_POLICY • AUDIT_USING • AUDIT_REPLACE • AUDIT_REMOVE
Event Correlator	INTEGER	Correlation identifier for the operation being audited. Can be used to identify what audit records are associated with a single event.
Event Status	INTEGER	Status of audit event, represented by an SQLCODE where Successful event > = 0 Failed event < 0
User ID	VARCHAR(1024)	User ID at time of audit event.
Authorization ID	VARCHAR(128)	Authorization ID at time of audit event.
Database Name	CHAR(8)	Name of the database for which the event was generated. Blank if this was an instance level audit event.
Origin Node Number	SMALLINT	Node number at which the audit event occurred.
Coordinator Node Number	SMALLINT	Node number of the coordinator node.
Application ID	VARCHAR(255)	Application ID in use at the time the audit event occurred.
Application Name	VARCHAR(1024)	Application name in use at the time the audit event occurred.
Package Schema	VARCHAR(128)	Schema of the package in use at the time of the audit event.
Package Name	VARCHAR(128)	Name of package in use at the time the audit event occurred.
Package Section	SMALLINT	Section number in package being used at the time the audit event occurred

Table 41. Audit Record Layout for AUDIT Events (continued)

NAME	FORMAT	DESCRIPTION
Package Version	VARCHAR(64)	Version of the package in use at the time the audit event occurred.
Local Transaction ID	VARCHAR(10) FOR BIT DATA	The local transaction ID in use at the time the audit event occurred. This is the SQLU_TID structure that is part of the transaction logs.
Global Transaction ID	VARCHAR(30) FOR BIT DATA	The global transaction ID in use at the time the audit event occurred. This is the data field in the SQLP_GXID structure that is part of the transaction logs.
Client User ID	VARCHAR(255)	The value of the CURRENT CLIENT USERID special register at the time the audit event occurred.
Client Workstation Name	VARCHAR(255)	The value of the CURRENT CLIENT_WRKSTNNAME special register at the time the audit event occurred.
Client Application Name	VARCHAR(255)	The value of the CURRENT CLIENT_APPLNAME special register at the time the audit event occurred.
Client Accounting String	VARCHAR(255)	The value of the CURRENT CLIENT_ACCTNG special register at the time the audit event occurred.
Trusted Context Name	VARCHAR(128)	The name of the trusted context associated with the trusted connection.
Connection Trust Type	INTEGER	Possible values are: IMPLICIT_TRUSTED_CONNECTION EXPLICIT_TRUSTED_CONNECTION
Role Inherited	VARCHAR(128)	The role inherited through a trusted connection.
Policy Name	VARCHAR(128)	The audit policy name.
Policy Association Object Type	CHAR(1)	The type of the object that the audit policy is associated with. Possible values include: • N = Nickname • S = MQT • T = Table (untyped) • i = Authorization ID • g= Authority • x = Trusted context • blank = Database
Policy Association Subobject Type	CHAR(1)	The type of sub-object that the audit policy is associated with. If the Object Type is ? (authorization id), then possible values are: • U = User • G = Group • R = Role
Policy Association Object Name	VARCHAR(128)	The name of the object that the audit policy is associated with.
Policy Association Object Schema	VARCHAR(128)	The schema name of the object that the audit policy is associated with. This is NULL if the Policy Association Object Type identifies an object to which a schema does not apply.
Audit Status	CHAR(1)	The status of the AUDIT category in an audit policy. Possible values are: • B-Both • F-Failure • N-None • S-Success

Table 41. Audit Record Layout for AUDIT Events (continued)

NAME	FORMAT	DESCRIPTION
Checking Status	CHAR(1)	The status of the CHECKING category in an audit policy. Possible values are: • B-Both • F-Failure • N-None • S-Success
Context Status	CHAR(1)	The status of the CONTEXT category in an audit policy. Possible values are: • B-Both • F-Failure • N-None • S-Success
Execute Status	CHAR(1)	The status of the EXECUTE category in an audit policy. Possible values are: • B-Both • F-Failure • N-None • S-Success
Execute With Data	CHAR(1)	The WITH DATA option of the EXECUTE category in the audit policy. Possible values are: • Y-WITH DATA • N-WITHOUT DATA
Objmaint Status	CHAR(1)	The status of the OBJMAINT category in an audit policy. Possible values are: • B-Both • F-Failure • N-None • S-Success
Secmaint Status	CHAR(1)	The status of the SECMAINT category in an audit policy. See Audit Status field for possible values.
Sysadmin Status	CHAR(1)	The status of the SYSADMIN category in an audit policy. Possible values are: • B-Both • F-Failure • N-None • S-Success
Validate Status	CHAR(1)	The status of the VALIDATE category in an audit policy. Possible values are: • B-Both • F-Failure • N-None • S-Success
Error Type	CHAR(8)	The error type in an audit policy. Possible values are: AUDIT and NORMAL.
Data Path	VARCHAR(1024)	The path to the active audit logs specified on the db2audit configure command.

Table 41. Audit Record Layout for AUDIT Events (continued)

NAME	FORMAT	DESCRIPTION
Archive Path	VARCHAR(1024)	The path to the archived audit logs specified on the db2audit configure command

Audit record layout for CHECKING events

The format of the audit record for CHECKING events is shown in the following table.

Sample audit record:

```
timestamp=1998-06-24-08.42.11.622984;
category=CHECKING;
audit event=CHECKING_OBJECT;
event correlator=2;
event status=0;
database=FOO;
userid=boss;
authid=BOSS;
application id=*LOCAL.newton.980624124210;
application name=testapp;
package schema=NULLID;
package name=SYSSH200;
package section=0;
object schema=GSTAGER;
object name=NONE;
object type=REOPT_VALUES;
access approval reason=DBADM;
access attempted=STORE;
```

Table 42. Audit record layout for CHECKING events

NAME	FORMAT	DESCRIPTION
Timestamp	CHAR(26)	Date and time of the audit event.
Category	CHAR(8)	Category of audit event. Possible values are: CHECKING
Audit Event	VARCHAR(32)	Specific Audit Event. Possible values include: CHECKING_OBJECT, CHECKING_FUNCTION, CHECKING_TRANSFER, and CHECKING_MEMBERSHIP_IN_ROLES
Event Correlator	INTEGER	Correlation identifier for the operation being audited. Can be used to identify what audit records are associated with a single event.
Event Status	INTEGER	Status of audit event, represented by an SQLCODE where Successful event > = 0 Failed event < 0
Database Name	CHAR(8)	Name of the database for which the event was generated. Blank if this was an instance level audit event.
User ID	VARCHAR(1024)	User ID at time of audit event.
Authorization ID	VARCHAR(128)	Authorization ID at time of audit event.
Origin Node Number	SMALLINT	Node number at which the audit event occurred.
Coordinator Node Number	SMALLINT	Node number of the coordinator node.

Table 42. Audit record layout for CHECKING events (continued)

NAME	FORMAT	DESCRIPTION
Application ID	VARCHAR(255)	Application ID in use at the time the audit event occurred.
Application Name	VARCHAR(1024)	Application name in use at the time the audit event occurred.
Package Schema	VARCHAR(128)	Schema of the package in use at the time of the audit event.
Package Name	VARCHAR(128)	Name of package in use at the time the audit event occurred.
Package Section Number	SMALLINT	Section number in package being used at the time the audit event occurred.
Object Schema	VARCHAR(128)	Schema of the object for which the audit event was generated.
Object Name	VARCHAR(128)	Name of object for which the audit event was generated.
Object Type	VARCHAR(32)	Type of object for which the audit event was generated. Possible values include: those shown in the topic titled "Audit record object types".
Access Approval Reason	CHAR(18)	Indicates the reason why access was approved for this audit event. Possible values include: those shown in the topic titled "List of possible CHECKING access approval reasons".
Access Attempted	CHAR(18)	Indicates the type of access that was attempted. Possible values include: those shown in the topic titled "List of possible CHECKING access attempted types".
Package Version	VARCHAR (64)	Version of the package in use at the time that the audit event occurred.
Checked Authorization ID	VARCHAR(128)	Authorization ID is checked when it is different than the authorization ID at the time of the audit event. For example, this can be the target owner in a TRANSFER OWNERSHIP statement. When the audit event is SWITCH_USER, this field represents the authorization ID that is switched to.
Local Transaction ID	VARCHAR(10) FOR BIT DATA	The local transaction ID in use at the time the audit event occurred. This is the SQLU_TID structure that is part of the transaction logs.
Global Transaction ID	VARCHAR(30) FOR BIT DATA	The global transaction ID in use at the time the audit event occurred. This is the data field in the SQLP_GXID structure that is part of the transaction logs.
Client User ID	VARCHAR(255)	The value of the CURRENT CLIENT USERID special register at the time the audit event occurred.
Client Workstation Name	VARCHAR(255)	The value of the CURRENT CLIENT_WRKSTNNAME special register at the time the audit event occurred.
Client Application Name	VARCHAR(255)	The value of the CURRENT CLIENT_APPLNAME special register at the time the audit event occurred.
Client Accounting String	VARCHAR(255)	The value of the CURRENT CLIENT_ACCTNG special register at the time the audit event occurred.
Trusted Context Name	VARCHAR(128)	The name of the trusted context associated with the trusted connection.
Connection Trust Type	INTEGER	Possible values are: IMPLICIT_TRUSTED_CONNECTION EXPLICIT_TRUSTED_CONNECTION
Role Inherited	VARCHAR(128)	The role inherited through a trusted connection.

CHECKING access approval reasons

The following list shows the possible CHECKING access approval reasons.

0x0000000000000001 ACCESS DENIED
Access is not approved; rather, it was denied.

0x0000000000000002 SYSADM
Access is approved; the application or user has SYSADM authority.

0x0000000000000004 SYSCTRL
Access is approved; the application or user has SYSCTRL authority.

0x0000000000000008 SYSMAINT
Access is approved; the application or user has SYSMAINT authority.

0x0000000000000010 DBADM
Access is approved; the application or user has DBADM authority.

0x0000000000000020 DATABASE PRIVILEGE
Access is approved; the application or user has an explicit privilege on the database.

0x0000000000000040 OBJECT PRIVILEGE
Access is approved; the application or user has a privilege on the object or function.

0x0000000000000080 DEFINER
Access is approved; the application or user is the definer of the object or function.

0x0000000000000100 OWNER
Access is approved; the application or user is the owner of the object or function.

0x0000000000000200 CONTROL
Access is approved; the application or user has CONTROL privilege on the object or function.

0x0000000000000400 BIND
Access is approved; the application or user has bind privilege on the package.

0x0000000000000800 SYSQUIESCE
Access is approved; if the instance or database is in quiesce mode, the application or user may connect or attach.

0x0000000000001000 SYSMON
Access is approved; the application or user has SYSMON authority.

0x0000000000002000 SECADM
Access is approved; the application or user has SECADM authority.

0x0000000000004000 SETSESSIONUSER
Access is approved; the application or user has SETSESSIONUSER authority.

0x0000000000008000 TRUSTED_CONTEXT_MATCH
Connection attributes matched the attributes of a unique trusted context defined at the DB2 server.

0x0000000000010000 TRUSTED_CONTEXT_USE
Access is approved to use a trusted context.

CHECKING access attempted types

The following list shows the possible CHECKING access attempted types.

If Audit Event is CHECKING_TRANSFER, then the audit entry reflects that a privilege is held or not.

0x0000000000000001 CONTROL
Attempt to verify if CONTROL privilege is held.

0x0000000000000002 ALTER
Attempt to alter an object or to verify if ALTER privilege is held if Audit Event is CHECKING_TRANSFER.

0x0000000000000004 DELETE
Attempt to delete an object or to verify if DELETE privilege is held if Audit Event is CHECKING_TRANSFER.

0x0000000000000008 INDEX
Attempt to use an index or to verify if INDEX privilege is held if Audit Event is CHECKING_TRANSFER.

0x0000000000000010 INSERT
Attempt to insert into an object or to verify if INSERT privilege is held if Audit Event is CHECKING_TRANSFER.

0x0000000000000020 SELECT
Attempt to query a table or view or to verify if SELECT privilege is held if Audit Event is CHECKING_TRANSFER.

0x0000000000000040 UPDATE
Attempt to update data in an object or to verify if UPDATE privilege is held if Audit Event is CHECKING_TRANSFER.

0x0000000000000080 REFERENCE
Attempt to establish referential constraints between objects or to verify if REFERENCE privilege is held if Audit Event is CHECKING_TRANSFER.

0x0000000000000100 CREATE
Attempt to create an object.

0x0000000000000200 DROP
Attempt to drop an object.

0x0000000000000400 CREATEIN
Attempt to create an object within another schema.

0x0000000000000800 DROPIN
Attempt to drop an object found within another schema.

0x0000000000001000 ALTERIN
Attempt to alter or modify an object found within another schema.

0x0000000000002000 EXECUTE
Attempt to execute or run an application or to invoke a routine, create a function sourced from the routine (applies to functions only), or reference a routine in any DDL statement or to verify if EXECUTE privilege is held if Audit Event is CHECKING_TRANSFER.

0x0000000000004000 BIND
Attempt to bind or prepare an application.

0x0000000000008000 SET EVENT MONITOR
Attempt to set event monitor switches.

0x0000000000010000 SET CONSTRAINTS
Attempt to set constraints on an object.

0x0000000000020000 COMMENT ON
Attempt to create comments on an object.

0x0000000000040000 GRANT
Attempt to grant privileges or roles on an object to another authorization ID.

0x0000000000080000 REVOKE
Attempt to revoke privileges or roles from an object from an authorization ID.

0x0000000000100000 LOCK
Attempt to lock an object.

0x0000000000200000 RENAME
Attempt to rename an object.

0x0000000000400000 CONNECT
Attempt to connect to an object.

0x0000000000800000 Member of SYS Group
Attempt to access or use a member of the SYS group.

0x0000000001000000 Access All
Attempt to execute a statement with all required privileges on objects held (only used for DBADM/SYSADM).

0x0000000002000000 Drop All
Attempt to drop multiple objects.

0x0000000004000000 LOAD
Attempt to load a table in a table space.

0x0000000008000000 USE
Attempt to create a table in a table space or to verify if USE privilege is held if Audit Event is CHECKING_TRANSFER.

0x0000000010000000 SET SESSION_USER
Attempt to execute the SET SESSION_USER statement.

0x0000000020000000 FLUSH
Attempt to execute the FLUSH statement.

0x0000000040000000 STORE
Attempt to view the values of a re-optimized statement in the EXPLAIN_PREDICATE table.

0x0000000400000000 TRANSFER
Attempt to transfer an object.

0x0000000800000000 ALTER_WITH_GRANT
Attempt to verify if ALTER with GRANT privilege is held.

0x0000001000000000 DELETE_WITH_GRANT
Attempt to verify if DELETE with GRANT privilege is held.

0x0000002000000000 INDEX_WITH_GRANT
Attempt to verify if INDEX with GRANT privilege is held

0x0000004000000000 INSERT_WITH_GRANT
Attempt to verify if INSERT with GRANT privilege is held.

0x0000008000000000 SELECT_WITH_GRANT
Attempt to verify if SELECT with GRANT privilege is held.

0x0000010000000000 UPDATE_WITH_GRANT
Attempt to verify if UPDATE with GRANT privilege is held.

0x0000020000000000 REFERENCE_WITH_GRANT
Attempt to verify if REFERENCE with GRANT privilege is held.

0x0000040000000000 USAGE
Attempt to use a sequence or an XSR object or to verify if USAGE privilege is held if Audit Event is CHECKING_TRANSFER.

0x0000080000000000 SET ROLE
Attempt to set a role.

0x0000100000000000 EXPLICIT_TRUSTED_CONNECTION
Attempt to establish an explicit trusted connection.

0x0000200000000000 IMPLICIT_TRUSTED_CONNECTION
Attempt to establish an implicit trusted connection.

0x0000400000000000 READ
Attempt to read a global variable.

0x0000800000000000 WRITE
Attempt to write a global variable.

0x0001000000000000 SWITCH_USER
Attempt to switch a user ID on an explicit trusted connection.

0x0002000000000000 AUDIT_USING
Attempt to associate an audit policy with an object.

0x0004000000000000 AUDIT_REPLACE
Attempt to replace an audit policy association with an object.

0x0008000000000000 AUDIT_REMOVE
Attempt to remove an audit policy association with an object.

0x0010000000000000 AUDIT_ARCHIVE
Attempt to archive the audit log.

0x0020000000000000 AUDIT_EXTRACT
Attempt to extract the audit log.

0x0040000000000000 AUDIT_LIST_LOGS
Attempt to list the audit logs.

Audit record layout for OBJMAINT events

The format of the audit record for OBJMAINT events is shown in the following table.

Sample audit record:

```
timestamp=1998-06-24-08.42.41.957524;
category=OBJMAINT;
audit event=CREATE_OBJECT;
event correlator=3;
event status=0;
database=FOO;
userid=boss;
authid=BOSS;
application id=*LOCAL.newton.980624124210;
application name=testapp;
package schema=NULLID;
package name=SQLC28A1;
```

```
package section=0;
object schema=BOSS;
object name=AUDIT;
object type=TABLE;
```

Table 43. Audit Record Layout for OBJMAINT Events

NAME	FORMAT	DESCRIPTION
Timestamp	CHAR(26)	Date and time of the audit event.
Category	CHAR(8)	Category of audit event. Possible values are: OBJMAINT
Audit Event	VARCHAR(32)	Specific Audit Event. Possible values include: CREATE_OBJECT, RENAME_OBJECT, DROP_OBJECT, and ALTER_OBJECT. ALTER_OBJECT events are generated only when altering protected tables.
Event Correlator	INTEGER	Correlation identifier for the operation being audited. Can be used to identify what audit records are associated with a single event.
Event Status	INTEGER	Status of audit event, represented by an SQLCODE where Successful event > = 0 Failed event < 0
Database Name	CHAR(8)	Name of the database for which the event was generated. Blank if this was an instance level audit event.
User ID	VARCHAR(1024)	User ID at time of audit event.
Authorization ID	VARCHAR(128)	Authorization ID at time of audit event.
Origin Node Number	SMALLINT	Node number at which the audit event occurred.
Coordinator Node Number	SMALLINT	Node number of the coordinator node.
Application ID	VARCHAR(255)	Application ID in use at the time the audit event occurred.
Application Name	VARCHAR(1024)	Application name in use at the time the audit event occurred.
Package Schema	VARCHAR(128)	Schema of the package in use at the time of the audit event.
Package Name	VARCHAR(256)	Name of package in use at the time the audit event occurred.
Package Section Number	SMALLINT	Section number in package being used at the time the audit event occurred.
Object Schema	VARCHAR(128)	Schema of the object for which the audit event was generated.
Object Name	VARCHAR(128)	Name of object for which the audit event was generated.
Object Type	VARCHAR(32)	Type of object for which the audit event was generated. Possible values include: those shown in the topic titled “Audit record object types”.
Package Version	VARCHAR(64)	Version of the package in use at the time the audit event occurred.
Security Policy Name	VARCHAR(128)	The name of the security policy if the object type is TABLE and that table is associated with a security policy.

Table 43. Audit Record Layout for OBJMAINT Events (continued)

NAME	FORMAT	DESCRIPTION
Alter Action	VARCHAR(32)	Specific Alter operation Possible values include: • ADD_PROTECTED_COLUMN • ADD_COLUMN_PROTECTION • DROP_COLUMN_PROTECTION • ADD_ROW_PROTECTION • ADD_SECURITY_POLICY • ADD_ELEMENT • ADD COMPONENT • USE GROUP AUTHORIZATIONS • IGNORE GROUP AUTHORIZATIONS • USE ROLE AUTHORIZATIONS • IGNORE ROLE AUTHORIZATIONS • OVERRIDE NOT AUTHORIZED WRITE SECURITY LABEL • RESTRICT NOT AUTHORIZED WRITE SECURITY LABEL
Protected Column Name	VARCHAR(128)	If the Alter Action is ADD_COLUMN_PROTECTION or DROP_COLUMN_PROTECTION this is the name of the affected column.
Column Security Label	VARCHAR(128)	The security label protecting the column specified in the field Column Name.
Security Label Column Name	VARCHAR(128)	Name of the column containing the security label protecting the row.
Local Transaction ID	VARCHAR(10) FOR BIT DATA	The local transaction ID in use at the time the audit event occurred. This is the SQLU_TID structure that is part of the transaction logs.
Global Transaction ID	VARCHAR(30) FOR BIT DATA	The global transaction ID in use at the time the audit event occurred. This is the data field in the SQLP_GXID structure that is part of the transaction logs.
Client User ID	VARCHAR(255)	The value of the CURRENT CLIENT USERID special register at the time the audit event occurred.
Client Workstation Name	VARCHAR(255)	The value of the CURRENT CLIENT_WRKSTNNAME special register at the time the audit event occurred.
Client Application Name	VARCHAR(255)	The value of the CURRENT CLIENT_APPLNAME special register at the time the audit event occurred.
Client Accounting String	VARCHAR(255)	The value of the CURRENT CLIENT_ACCTNG special register at the time the audit event occurred.
Trusted Context Name	VARCHAR(128)	The name of the trusted context associated with the trusted connection.
Connection Trust Type	INTEGER	Possible values are: IMPLICIT_TRUSTED_CONNECTION EXPLICIT_TRUSTED_CONNECTION
Role Inherited	VARCHAR(128)	The role inherited through a trusted connection.

Audit record layout for SECMAINT events

The format of the audit record for SECMAINT events is shown in the following table.

Sample audit record:

```
timestamp=1998-06-24-11.57.45.188101;
category=SECMAINT;
audit event=GRANT;
event correlator=4;
event status=0;
database=FOO;
userid=boss;
authid=BOSS;
application id=*LOCAL.boss.980624155728;
application name=db2bp;
package schema=NULLID;
package name=SQLC28A1;
package section=0;
object schema=BOSS;
object name=T1;
object type=TABLE;
grantor=BOSS;
grantee=WORKER;
grantee type=USER;
privilege=SELECT;
```

Table 44. Audit Record Layout for SECMAINT Events

NAME	FORMAT	DESCRIPTION
Timestamp	CHAR(26)	Date and time of the audit event.
Category	CHAR(8)	Category of audit event. Possible values are: SECMAINT
Audit Event	VARCHAR(32)	Specific Audit Event. Possible values include: GRANT, REVOKE, IMPLICIT_GRANT, IMPLICIT_REVOKE, SET_SESSION_USER, UPDATE_DBM_CFG, TRANSFER_OWNERSHIP, ADD_DEFAULT_ROLE, DROP_DEFAULT_ROLE, ALTER_DEFAULT_ROLE, ADD_USER, DROP_USER, ALTER_USER_ADD_ROLE, ALTER_USER_DROP_ROLE, ALTER_USER_AUTHENTICATION, and ALTER SECURITY POLICY.
Event Correlator	INTEGER	Correlation identifier for the operation being audited. Can be used to identify what audit records are associated with a single event.
Event Status	INTEGER	Status of audit event, represented by an SQLCODE where Successful event > = 0 Failed event < 0
Database Name	CHAR(8)	Name of the database for which the event was generated. Blank if this was an instance level audit event.
User ID	VARCHAR(1024)	User ID at time of audit event.
Authorization ID	VARCHAR(128)	Authorization ID at time of audit event.
Origin Node Number	SMALLINT	Node number at which the audit event occurred.
Coordinator Node Number	SMALLINT	Node number of the coordinator node.
Application ID	VARCHAR(255)	Application ID in use at the time the audit event occurred.

Table 44. Audit Record Layout for SECMAINT Events (continued)

NAME	FORMAT	DESCRIPTION
Application Name	VARCHAR(1024)	Application name in use at the time the audit event occurred.
Package Schema	VARCHAR(128)	Schema of the package in use at the time of the audit event.
Package Name	VARCHAR(128)	Name of package in use at the time the audit event occurred.
Package Section Number	SMALLINT	Section number in package being used at the time the audit event occurred.
Object Schema	VARCHAR(128)	Schema of the object for which the audit event was generated. If the object type field is ACCESS_RULE then this field contains the security policy name associated with the rule. The name of the rule is stored in the field Object Name. If the object type field is SECURITY_LABEL, then this field contains the name of the security policy that the security label is part of. The name of the security label is stored in the field Object Name.
Object Name	VARCHAR(128)	Name of object for which the audit event was generated. Represents a role name when the audit event is any of ADD_DEFAULT_ROLE, DROP_DEFAULT_ROLE, ALTER_DEFAULT_ROLE, ADD_USER, DROP_USER, ALTER_USER_ADD_ROLE, ALTER_USER_DROP_ROLE, or ALTER_USER_AUTHENTICATION If the object type field is ACCESS_RULE then this field contains the name of the rule. The security policy name associated with the rule is stored in the field Object Schema. If the object type field is SECURITY_LABEL, then this field contains the name of the security label. The name of the security policy that it is part of is stored in the field Object Schema.
Object Type	VARCHAR(32)	Type of object for which the audit event was generated. Possible values include: those shown in the topic titled “Audit record object types”. The value is ROLE when the audit event is any of ADD_DEFAULT_ROLE, DROP_DEFAULT_ROLE, ALTER_DEFAULT_ROLE, ADD_USER, DROP_USER, ALTER_USER_ADD_ROLE, ALTER_USER_DROP_ROLE, or ALTER_USER_AUTHENTICATION
Grantor	VARCHAR(128)	The ID of the grantor or the revoker of the privilege or authority.
Grantee	VARCHAR(128)	Grantee ID for which a privilege or authority was granted or revoked. Represents a trusted context object when the audit event is any of ADD_DEFAULT_ROLE, DROP_DEFAULT_ROLE, ALTER_DEFAULT_ROLE, ADD_USER, DROP_USER, ALTER_USER_ADD_ROLE, ALTER_USER_DROP_ROLE, or ALTER_USER_AUTHENTICATION
Grantee Type	VARCHAR(32)	Type of the grantee that was granted to or revoked from. Possible values include: USER, GROUP, ROLE, AMBIGUOUS, or is TRUSTED_CONTEXT when the audit event is any of: ADD_DEFAULT_ROLE, DROP_DEFAULT_ROLE, ALTER_DEFAULT_ROLE, ADD_USER, DROP_USER, ALTER_USER_ADD_ROLE, ALTER_USER_DROP_ROLE, or ALTER_USER_AUTHENTICATION

Table 44. Audit Record Layout for SECMAINT Events (continued)

NAME	FORMAT	DESCRIPTION
Privilege or Authority	CHAR(18)	Indicates the type of privilege or authority granted or revoked. Possible values include: those shown in the topic titled "List of possible SECMAINT privileges or authorities". The value is ROLE MEMBERSHIP when the audit event is any of ADD_DEFAULT_ROLE, DROP_DEFAULT_ROLE, ALTER_DEFAULT_ROLE, ADD_USER, DROP_USER, ALTER_USER_ADD_ROLE, ALTER_USER_DROP_ROLE, or ALTER_USER_AUTHENTICATION
Package Version	VARCHAR(64)	Version of the package in use at the time the audit event occurred.
Access Type	VARCHAR(32)	The access type for which a security label is granted. Possible values: • READ • WRITE • ALL The access type for which a security policy is altered. Possible values: • USE GROUP AUTHORIZATIONS • IGNORE GROUP AUTHORIZATIONS • USE ROLE AUTHORIZATIONS • IGNORE ROLE AUTHORIZATIONS • OVERRIDE NOT AUTHORIZED WRITE SECURITY LABEL • RESTRICT NOT AUTHORIZED WRITE SECURITY LABEL
Assumable Authid	VARCHAR(128)	When the privilege granted is a SETSESSIONUSER privilege this is the authorization ID that the grantee is allowed to set as the session user.
Local Transaction ID	VARCHAR(10) FOR BIT DATA	The local transaction ID in use at the time the audit event occurred. This is the SQLU_TID structure that is part of the transaction logs.
Global Transaction ID	VARCHAR(30) FOR BIT DATA	The global transaction ID in use at the time the audit event occurred. This is the data field in the SQLP_GXID structure that is part of the transaction logs.
Grantor Type	VARCHAR(32)	Type of the grantor. Possible values include: USER.
Client User ID	VARCHAR(255)	The value of the CURRENT CLIENT USERID special register at the time the audit event occurred.
Client Workstation Name	VARCHAR(255)	The value of the CURRENT CLIENT_WRKSTNNAME special register at the time the audit event occurred.
Client Application Name	VARCHAR(255)	The value of the CURRENT CLIENT_APPLNAME special register at the time the audit event occurred.
Client Accounting String	VARCHAR(255)	The value of the CURRENT CLIENT_ACCTNG special register at the time the audit event occurred.
Trusted Context User	VARCHAR(128)	Identifies a trusted context user when the audit event is ADD_USER or DROP_USER.
Trusted Context User Authentication	INTEGER	Specifies the authentication setting for a trusted context user when the audit event is ADD_USER, DROP_USER or ALTER_USER_AUTHENTICATION 1 : Authentication is required 0 : Authentication is not required

Table 44. Audit Record Layout for SECMAINT Events (continued)

NAME	FORMAT	DESCRIPTION
Trusted Context Name	VARCHAR(128)	The name of the trusted context associated with the trusted connection.
Connection Trust Type	INTEGER	Possible values are: IMPLICIT_TRUSTED_CONNECTION EXPLICIT_TRUSTED_CONNECTION
Role Inherited	VARCHAR(128)	The role inherited through a trusted connection.

SECMAINT privileges or authorities

The following list shows the possible SECMAINT privileges or authorities.

0x0000000000000001 Control Table
Control privilege granted or revoked on or from a table or view.

0x0000000000000002 ALTER
Privilege granted or revoked to alter a table or sequence.

0x0000000000000004 ALTER with GRANT
Privilege granted or revoked to alter a table or sequence with granting of privileges allowed.

0x0000000000000008 DELETE TABLE
Privilege granted or revoked to drop a table or view.

0x0000000000000010 DELETE TABLE with GRANT
Privilege granted or revoked to drop a table with granting of privileges allowed.

0x0000000000000020 Table Index
Privilege granted or revoked on or from an index.

0x0000000000000040 Table Index with GRANT
Privilege granted or revoked on or from an index with granting of privileges allowed.

0x0000000000000080 Table INSERT
Privilege granted or revoked on or from an insert on a table or view.

0x0000000000000100 Table INSERT with GRANT
Privilege granted or revoked on or from an insert on a table with granting of privileges allowed.

0x0000000000000200 Table SELECT
Privilege granted or revoked on or from a select on a table.

0x0000000000000400 Table SELECT with GRANT
Privilege granted or revoked on or from a select on a table with granting of privileges allowed.

0x0000000000000800 Table UPDATE
Privilege granted or revoked on or from an update on a table or view.

0x0000000000001000 Table UPDATE with GRANT
Privilege granted or revoked on or from an update on a table or view with granting of privileges allowed.

0x0000000000002000 Table REFERENCE
Privilege granted or revoked on or from a reference on a table.

0x0000000000004000 Table REFERENCE with GRANT
Privilege granted or revoked on or from a reference on a table with granting of privileges allowed.

0x0000000000020000 CREATEIN Schema
CREATEIN privilege granted or revoked on or from a schema.

0x0000000000040000 CREATEIN Schema with GRANT
CREATEIN privilege granted or revoked on or from a schema with granting of privileges allowed.

0x0000000000080000 DROPIN Schema
DROPIN privilege granted or revoked on or from a schema.

0x0000000000100000 DROPIN Schema with GRANT
DROPIN privilege granted or revoked on or from a schema with granting of privileges allowed.

0x0000000000200000 ALTERIN Schema
ALTERIN privilege granted or revoked on or from a schema.

0x0000000000400000 ALTERIN Schema with GRANT
ALTERIN privilege granted or revoked on or from a schema with granting of privileges allowed.

0x0000000000800000 DBADM Authority
DBADM authority granted or revoked.

0x0000000001000000 CREATETAB Authority
Createtab authority granted or revoked.

0x0000000002000000 BINDADD Authority
Bindadd authority granted or revoked.

0x0000000004000000 CONNECT Authority
CONNECT authority granted or revoked.

0x0000000008000000 Create not fenced Authority
Create not fenced authority granted or revoked.

0x0000000010000000 Implicit Schema Authority
Implicit schema authority granted or revoked.

0x0000000020000000 Server PASSTHRU
Privilege granted or revoked to use the pass-through facility with this server (federated database data source).

0x0000000040000000 ESTABLISH TRUSTED CONNECTION
Trusted connection was created

0x0000000100000000 Table Space USE
Privilege granted or revoked to create a table in a table space.

0x0000000200000000 Table Space USE with GRANT
Privilege granted or revoked to create a table in a table space with granting of privileges allowed.

0x0000000400000000 Column UPDATE
Privilege granted or revoked on or from an update on one or more specific columns of a table.

0x0000000800000000 Column UPDATE with GRANT
Privilege granted or revoked on or from an update on one or more specific columns of a table with granting of privileges allowed.

0x0000001000000000 Column REFERENCE
Privilege granted or revoked on or from a reference on one or more specific columns of a table.

0x0000002000000000 Column REFERENCE with GRANT
Privilege granted or revoked on or from a reference on one or more specific columns of a table with granting of privileges allowed.

0x0000004000000000 LOAD Authority
LOAD authority granted or revoked.

0x0000008000000000 Package BIND
BIND privilege granted or revoked on or from a package.

0x0000010000000000 Package BIND with GRANT
BIND privilege granted or revoked on or from a package with granting of privileges allowed.

0x0000020000000000 EXECUTE
EXECUTE privilege granted or revoked on or from a package or a routine.

0x0000040000000000 EXECUTE with GRANT
EXECUTE privilege granted or revoked on or from a package or a routine with granting of privileges allowed.

0x0000080000000000 EXECUTE IN SCHEMA
EXECUTE privilege granted or revoked for all routines in a schema.

0x0000100000000000 EXECUTE IN SCHEMA with GRANT
EXECUTE privilege granted or revoked for all routines in a schema with granting of privileges allowed.

0x0000200000000000 EXECUTE IN TYPE
EXECUTE privilege granted or revoked for all routines in a type.

0x0000400000000000 EXECUTE IN TYPE with GRANT
EXECUTE privilege granted or revoked for all routines in a type with granting of privileges allowed.

0x0000800000000000 CREATE EXTERNAL ROUTINE
CREATE EXTERNAL ROUTINE privilege granted or revoked.

0x0001000000000000 QUIESCE_CONNECT
QUIESCE_CONNECT privilege granted or revoked.

0x0004000000000000 SECADM Authority
SECADM authority granted or revoked

0x0008000000000000 USAGE Authority
USAGE privilege granted or revoked on or from a sequence

0x0010000000000000 USAGE with GRANT Authority
USAGE privilege granted or revoked on or from a sequence with granting of privileges allowed.

0x0020000000000000 WITH ADMIN Option
WITH ADMIN Option is granted or revoked to or from a role.

0x0040000000000000 SETSESSIONUSER Privilege
SETSESSIONUSER granted or revoked

0x0080000000000000 Exemption
Exemption granted or revoked

0x0100000000000000 Security label
Security label granted or revoked

0x0200000000000000 WRITE with GRANT
Privilege granted or revoked to write a global variable with granting of privileges allowed.

0x0400000000000000 Role Membership
Role membership that is granted or revoked

0x0800000000000000 Role Membership with ADMIN Option
Role membership with ADMIN Option that is granted or revoked

0x1000000000000000 READ
Privilege granted or revoked to read a global variable.

0x2000000000000000 READ with GRANT
Privilege granted or revoked to read a global variable with granting of privileges allowed.

0x4000000000000000 WRITE
Privilege granted or revoked to write a global variable.

Audit record layout for SYSADMIN events

The following table shows the audit record layout for SYSADMIN events.

Sample audit record:

```
timestamp=1998-06-24-11.54.04.129923;
category=SYSADMIN;
audit event=DB2AUDIT;
event correlator=1;
event status=0;
userid=boss;authid=BOSS;
application id=*LOCAL.boss.980624155404;
application name=db2audit;
```

Table 45. Audit Record Layout for SYSADMIN Events

NAME	FORMAT	DESCRIPTION
Timestamp	CHAR(26)	Date and time of the audit event.
Category	CHAR(8)	Category of audit event. Possible values are: SYSADMIN
Audit Event	VARCHAR(32)	Specific Audit Event. Possible values include: Those shown in the list following this table.
Event Correlator	INTEGER	Correlation identifier for the operation being audited. Can be used to identify what audit records are associated with a single event.
Event Status	INTEGER	Status of audit event, represented by an SQLCODE where Successful event > = 0 Failed event < 0
Database Name	CHAR(8)	Name of the database for which the event was generated. Blank if this was an instance level audit event.
User ID	VARCHAR(1024)	User ID at time of audit event.
Authorization ID	VARCHAR(128)	Authorization ID at time of audit event.
Origin Node Number	SMALLINT	Node number at which the audit event occurred.

Table 45. Audit Record Layout for SYSADMIN Events (continued)

NAME	FORMAT	DESCRIPTION
Coordinator Node Number	SMALLINT	Node number of the coordinator node.
Application ID	VARCHAR(255)	Application ID in use at the time the audit event occurred.
Application Name	VARCHAR(1024)	Application name in use at the time the audit event occurred.
Package Schema	VARCHAR(128)	Schema of the package in use at the time of the audit event.
Package Name	VARCHAR(128)	Name of package in use at the time the audit event occurred.
Package Section Number	SMALLINT	Section number in package being used at the time the audit event occurred.
Package Version	VARCHAR(64)	Version of the package in use at the time the audit event occurred.
Local Transaction ID	VARCHAR(10) FOR BIT DATA	The local transaction ID in use at the time the audit event occurred. This is the SQLU_TID structure that is part of the transaction logs.
Global Transaction ID	VARCHAR(30) FOR BIT DATA	The global transaction ID in use at the time the audit event occurred. This is the data field in the SQLP_GXID structure that is part of the transaction logs.
Client User ID	VARCHAR(255)	The value of the CURRENT CLIENT USERID special register at the time the audit event occurred.
Client Workstation Name	VARCHAR(255)	The value of the CURRENT CLIENT_WRKSTNNAME special register at the time the audit event occurred.
Client Application Name	VARCHAR(255)	The value of the CURRENT CLIENT_APPLNAME special register at the time the audit event occurred.
Client Accounting String	VARCHAR(255)	The value of the CURRENT CLIENT_ACCTNG special register at the time the audit event occurred.
Trusted Context Name	VARCHAR(128)	The name of the trusted context associated with the trusted connection.
Connection Trust Type	INTEGER	Possible values are: IMPLICIT_TRUSTED_CONNECTION EXPLICIT_TRUSTED_CONNECTION
Role Inherited	VARCHAR(128)	The role inherited through a trusted connection.

SYSADMIN audit events

The following table lists the possible SYSADMIN audit events.

Table 46. SYSADMIN Audit Events

START_DB2	ROLLFORWARD_DB
STOP_DB2	SET_RUNTIME_DEGREE
CREATE_DATABASE	SET_TABLESPACE_CONTAINERS
ALTER_DATABASE	UNCATALOG_DB
DROP_DATABASE	UNCATALOG_DCS_DB
UPDATE_DBM_CFG	UNCATALOG_NODE
UPDATE_DB_CFG	UPDATE_ADMIN_CFG
CREATE_TABLESPACE	UPDATE_MON_SWITCHES
DROP_TABLESPACE	LOAD_TABLE
ALTER_TABLESPACE	DB2AUDIT
RENAME_TABLESPACE	SET_APPL_PRIORITY
CREATE_NODEGROUP	CREATE_DB_AT_NODE
DROP_NODEGROUP	KILLDBM
ALTER_NODEGROUP	MIGRATE_SYSTEM_DIRECTORY
CREATE_BUFFERPOOL	DB2REMOT
DROP_BUFFERPOOL	DB2AUD
ALTER_BUFFERPOOL	MERGE_DBM_CONFIG_FILE
CREATE_EVENT_MONITOR	UPDATE_CLI_CONFIGURATION
DROP_EVENT_MONITOR	OPEN_TABLESPACE_QUERY
ENABLE_MULTIPAGE	SINGLE_TABLESPACE_QUERY
MIGRATE_DB_DIR	CLOSE_TABLESPACE_QUERY
DB2TRC	FETCH_TABLESPACE
DB2SET	OPEN_CONTAINER_QUERY
ACTIVATE_DB	FETCH_CONTAINER_QUERY
ADD_NODE	CLOSE_CONTAINER_QUERY
BACKUP_DB	GET_TABLESPACE_STATISTICS
CATALOG_NODE	DESCRIBE_DATABASE
CATALOG_DB	ESTIMATE_SNAPSHOT_SIZE
CATALOG_DCS_DB	READ_ASYNC_LOG_RECORD
CHANGE_DB_COMMENT	PRUNE_RECOVERY_HISTORY
DEACTIVATE_DB	UPDATE_RECOVERY_HISTORY
DROP_NODE_VERIFY	QUIESCE_TABLESPACE
FORCE_APPLICATION	UNLOAD_TABLE
GET_SNAPSHOT	UPDATE_DATABASE_VERSION
LIST_DRDA_INDOUBT_TRANSACTIONS	CREATE_INSTANCE
MIGRATE_DB	DELETE_INSTANCE
RESET_ADMIN_CFG	SET_EVENT_MONITOR
RESET_DB_CFG	GRANT_DBADM
RESET_DBM_CFG	REVOKE_DBADM
RESET_MONITOR	GRANT_DB_AUTHORITIES
RESTORE_DB	REVOKE_DB_AUTHORITIES
	REDISTRIBUTE_NODEGROUP

Audit record layout for VALIDATE events

The format of the audit record for VALIDATE events is shown in the following table.

Sample audit record:

```
timestamp=2007-05-07-10.30.51.585626;
category=VALIDATE;
audit event=AUTHENTICATION;
event correlator=1;
event status=0;
userid=newton;
authid=NEWTON;
execution id=gstager;
```

```
application id=*LOCAL.gstager.070507143051;
application name=db2bp;
auth type=SERVER;
plugin name=IBMOSauthserver;
```

Table 47. Audit Record Layout for VALIDATE Events

NAME	FORMAT	DESCRIPTION
Timestamp	CHAR(26)	Date and time of the audit event.
Category	CHAR(8)	Category of audit event. Possible values are: VALIDATE
Audit Event	VARCHAR(32)	Specific Audit Event. Possible values include: AUTHENTICATE and GET_USERMAPPING_FROM_PLUGIN. **Note:** GET_GROUPS, GET_USERID, and CHECK_GROUP_MEMBERSHIP are not generated in Version 9.5, and later.
Event Correlator	INTEGER	Correlation identifier for the operation being audited. Can be used to identify what audit records are associated with a single event.
Event Status	INTEGER	Status of audit event, represented by an SQLCODE where Successful event > = 0 Failed event < 0
Database Name	CHAR(8)	Name of the database for which the event was generated. Blank if this was an instance level audit event.
User ID	VARCHAR(1024)	User ID at time of audit event.
Authorization ID	VARCHAR(128)	Authorization ID at time of audit event.
Execution ID	VARCHAR(1024)	Execution ID in use at the time of the audit event.
Origin Node Number	SMALLINT	Node number at which the audit event occurred.
Coordinator Node Number	SMALLINT	Node number of the coordinator node.
Application ID	VARCHAR(255)	Application ID in use at the time the audit event occurred.
Application Name	VARCHAR(1024)	Application name in use at the time the audit event occurred.
Authentication Type	VARCHAR(32)	Authentication type at the time of the audit event.
Package Schema	VARCHAR(128)	Schema of the package in use at the time of the audit event.
Package Name	VARCHAR(128)	Name of package in use at the time the audit event occurred.
Package Section Number	SMALLINT	Section number in package being used at the time the audit event occurred.
Package Version	VARCHAR(64)	Version of the package in use at the time the audit event occurred.
Plug-in Name	VARCHAR(32)	The name of the plug-in in use at the time the audit event occurred.
Local Transaction ID	VARCHAR(10) FOR BIT DATA	The local transaction ID in use at the time the audit event occurred. This is the SQLU_TID structure that is part of the transaction logs.
Global Transaction ID	VARCHAR(30) FOR BIT DATA	The global transaction ID in use at the time the audit event occurred. This is the data field in the SQLP_GXID structure that is part of the transaction logs.
Client User ID	VARCHAR(255)	The value of the CURRENT CLIENT USERID special register at the time the audit event occurred.
Client Workstation Name	VARCHAR(255)	The value of the CURRENT CLIENT_WRKSTNNAME special register at the time the audit event occurred.

Table 47. Audit Record Layout for VALIDATE Events (continued)

NAME	FORMAT	DESCRIPTION
Client Application Name	VARCHAR(255)	The value of the CURRENT CLIENT_APPLNAME special register at the time the audit event occurred.
Client Accounting String	VARCHAR(255)	The value of the CURRENT CLIENT_ACCTNG special register at the time the audit event occurred.
Trusted Context Name	VARCHAR(128)	The name of the trusted context associated with the trusted connection.
Connection Trust Type	INTEGER	Possible values are: IMPLICIT_TRUSTED_CONNECTION EXPLICIT_TRUSTED_CONNECTION
Role Inherited	VARCHAR(128)	The name of the role inherited through the trusted context.

Audit record layout for CONTEXT events

The following table shows the audit record layout for CONTEXT events.

Sample audit record:

```
timestamp=1998-06-24-08.42.41.476840;
category=CONTEXT;
audit event=EXECUTE_IMMEDIATE;
event correlator=3;
database=FOO;
userid=boss;
authid=BOSS;
application id=*LOCAL.newton.980624124210;
application name=testapp;
package schema=NULLID;
package name=SQLC28A1;
package section=203;
text=create table audit(c1 char(10), c2 integer);
```

Table 48. Audit Record Layout for CONTEXT Events

NAME	FORMAT	DESCRIPTION
Timestamp	CHAR(26)	Date and time of the audit event.
Category	CHAR(8)	Category of audit event. Possible values are: CONTEXT
Audit Event	VARCHAR(32)	Specific Audit Event. Possible values include: Those shown in the list following this table.
Event Correlator	INTEGER	Correlation identifier for the operation being audited. Can be used to identify what audit records are associated with a single event.
Database Name	CHAR(8)	Name of the database for which the event was generated. Blank if this was an instance level audit event.
User ID	VARCHAR(1024)	User ID at time of audit event. When the audit event is SWITCH_USER, this field represents the user ID that is switched to.
Authorization ID	VARCHAR(128)	Authorization ID at time of audit event. When the audit event is SWITCH_USER, this field represents the authorization ID that is switched to.

Table 48. Audit Record Layout for CONTEXT Events (continued)

NAME	FORMAT	DESCRIPTION
Origin Node Number	SMALLINT	Node number at which the audit event occurred.
Coordinator Node Number	SMALLINT	Node number of the coordinator node.
Application ID	VARCHAR(255)	Application ID in use at the time the audit event occurred.
Application Name	VARCHAR(1024)	Application name in use at the time the audit event occurred.
Package Schema	VARCHAR(128)	Schema of the package in use at the time of the audit event.
Package Name	VARCHAR(128)	Name of package in use at the time the audit event occurred.
Package Section Number	SMALLINT	Section number in package being used at the time the audit event occurred.
Statement Text	CLOB(8M)	Text of the SQL or XQuery statement, if applicable. Null if no SQL or XQuery statement text is available.
Package Version	VARCHAR(64)	Version of the package in use at the time the audit event occurred.
Local Transaction ID	VARCHAR(10) FOR BIT DATA	The local transaction ID in use at the time the audit event occurred. This is the SQLU_TID structure that is part of the transaction logs.
Global Transaction ID	VARCHAR(30) FOR BIT DATA	The global transaction ID in use at the time the audit event occurred. This is the data field in the SQLP_GXID structure that is part of the transaction logs.
Client User ID	VARCHAR(255)	The value of the CURRENT CLIENT USERID special register at the time the audit event occurred.
Client Workstation Name	VARCHAR(255)	The value of the CURRENT CLIENT_WRKSTNNAME special register at the time the audit event occurred.
Client Application Name	VARCHAR(255)	The value of the CURRENT CLIENT_APPLNAME special register at the time the audit event occurred.
Client Accounting String	VARCHAR(255)	The value of the CURRENT CLIENT_ACCTNG special register at the time the audit event occurred.
Trusted Context Name	VARCHAR(128)	The name of the trusted context associated with the trusted connection.
Connection Trust Type	INTEGER	Possible values are: IMPLICIT_TRUSTED_CONNECTION EXPLICIT_TRUSTED_CONNECTION
Role Inherited	VARCHAR(128)	The role inherited through a trusted connection.

CONTEXT audit events

The following table lists the possible CONTEXT audit events.

Table 49. CONTEXT Audit Events

CONNECT	SET_APPL_PRIORITY
CONNECT_RESET	RESET_DB_CFG
ATTACH	GET_DB_CFG
DETACH	GET_DFLT_CFG
DARI_START	UPDATE_DBM_CFG
DARI_STOP	SET_MONITOR
BACKUP_DB	GET_SNAPSHOT
RESTORE_DB	ESTIMATE_SNAPSHOT_SIZE
ROLLFORWARD_DB	RESET_MONITOR
OPEN_TABLESPACE_QUERY	OPEN_HISTORY_FILE
FETCH_TABLESPACE	CLOSE_HISTORY_FILE
CLOSE_TABLESPACE_QUERY	FETCH_HISTORY_FILE
OPEN_CONTAINER_QUERY	SET_RUNTIME_DEGREE
CLOSE_CONTAINER_QUERY	UPDATE_AUDIT
FETCH_CONTAINER_QUERY	DBM_CFG_OPERATION
SET_TABLESPACE_CONTAINERS	DISCOVER
GET_TABLESPACE_STATISTIC	OPEN_CURSOR
READ_ASYNC_LOG_RECORD	CLOSE_CURSOR
QUIESCE_TABLESPACE	FETCH_CURSOR
LOAD_TABLE	EXECUTE
UNLOAD_TABLE	EXECUTE_IMMEDIATE
UPDATE_RECOVERY_HISTORY	PREPARE
PRUNE_RECOVERY_HISTORY	DESCRIBE
SINGLE_TABLESPACE_QUERY	BIND
LOAD_MSG_FILE	REBIND
UNQUIESCE_TABLESPACE	RUNSTATS
ENABLE_MULTIPAGE	REORG
DESCRIBE_DATABASE	REDISTRIBUTE
DROP_DATABASE	COMMIT
CREATE_DATABASE	ROLLBACK
ADD_NODE	REQUEST_ROLLBACK
FORCE_APPLICATION	IMPLICIT_REBIND
	EXTERNAL_CANCEL
	SWITCH_USER

Audit record layout for EXECUTE events

The following table describes all of the fields that are audited as part of the EXECUTE category.

Sample audit record:

Note: Unlike other audit categories, the EXECUTE category, when the audit log is viewed in a table format, can show multiple rows describing one event. The first record describes the main event, and its event column contains the key word STATEMENT. The remaining rows describe the parameter markers or host variables, one row per parameter, and their event column contains the key word DATA. When the audit log is viewed in report format, there is one record, but it has multiple entries for the Statement Value. The DATA key word is only be present in table format.

```
timestamp=2006-04-10-13.20.51.029203;
  category=EXECUTE;
  audit event=STATEMENT;
  event correlator=1;
  event status=0;
  database=SAMPLE;
```

```
userid=smith;
authid=SMITH;
session authid=SMITH;
application id=*LOCAL.prodrig.060410172044;
application name=myapp;
package schema=NULLID;
package name=SQLC2F0A;
package section=201;
uow id=2;
activity id=3;
statement invocation id=0;
statement nesting level=0;
statement text=SELECT * FROM DEPARTMENT WHERE DEPTNO = ? AND DEPTNAME = ?;
statement isolation level=CS;
compilation environment=
  isolation level=CS
  query optimization=5
  min_dec_div_3=NO
  degree=1
  sqlrules=DB2
  refresh age=+00000000000000.000000
  schema=SMITH
  maintained table type=SYSTEM
  resolution timestamp=2006-06-29-20.32.13.000000
  federated asynchrony=0;
value index=0;
value type=CHAR;
value data=C01;
value index=1;
value type=VARCHAR;
value index=INFORMATION CENTER;
```

Table 50. Audit Record Layout for EXECUTE Events

NAME	FORMAT	DESCRIPTION
Timestamp	CHAR(26)	Date and time of the audit event
Category	CHAR(8)	Category of audit event. Possible values are: EXECUTE

Table 50. Audit Record Layout for EXECUTE Events (continued)

NAME	FORMAT	DESCRIPTION
Audit Event	VARCHAR(32)	Specific Audit Event. Possible values include: • STATEMENT Execution of an SQL statement • DATA A host variable or parameter marker data values for the statement. This event is repeated for each host variable or parameter marker that is part of the statement. It is only present in a delimited extract of an audit log. • COMMIT Execution of a COMMIT statement • ROLLBACK Execution of a ROLLBACK statement • SAVEPOINT Execution of a SAVEPOINT statement • RELEASE SAVEPOINT Execution of a RELEASE SAVEPOINT statement • GLOBAL COMMIT Execution of a COMMIT within a global transaction • GLOBAL ROLLBACK Execution of a ROLLBACK within a global transaction • CONNECT Establishment of a database connection • CONNECT RESET Termination of a database connection • SWITCH USER Switching of a user within a trusted connection.
Event Correlator	INTEGER	Correlation identifier for the operation being audited. Can be used to identify what audit records are associated with a single event.
Event Status	INTEGER	Status of audit event, represented by an SQLCODE where Successful event > = 0 Failed event < 0
Database Name	CHAR(8)	Name of the database for which the event was generated. Blank if this was an instance level audit event
User ID	VARCHAR(1024)	User ID at time of audit event.

Table 50. Audit Record Layout for EXECUTE Events (continued)

NAME	FORMAT	DESCRIPTION
Authorization ID	VARCHAR(128)	The Statement Authorization ID at time of audit event.
Session Authorization ID	VARCHAR(128)	The Session Authorization ID at the time of the audit event.
Origin Node Number	SMALLINT	Node number at which the audit event occurred
Coordinator Node Number	SMALLINT	Node number of the coordinator node
Application ID	VARCHAR(255)	Application ID in use at the time the audit event occurred.
Application Name	VARCHAR(1024)	Application name in use at the time the audit event occurred.
Client User ID	VARCHAR(255)	The value of the CURRENT CLIENT USERID special register at the time the audit event occurred
Client Accounting String	VARCHAR(255)	The value of the CURRENT CLIENT_ACCTNG special register at the time the audit event occurred
Client Workstation Name	VARCHAR(255)	The value of the CURRENT CLIENT_WRKSTNNAME special register at the time the audit event occurred
Client Application Name	VARCHAR(255)	The value of the CURRENT CLIENT_APPLNAME special register at the time the audit event occurred
Trusted Context Name	VARCHAR(128)	The name of the trusted context associated with the trusted connection.
Connection Trust type	INTEGER	Possible values are IMPLICIT_TRUSTED_ CONNECTION and EXPLICIT_TRUSTED_ CONNECTION.
Role Inherited	VARCHAR(128)	The role inherited through a trusted connection.
Package Schema	VARCHAR(128)	Schema of the package in use at the time of the audit event.
Package Name	VARCHAR(128)	Name of package in use at the time the audit event occurred.
Package Section	SMALLINT	Section number in package being used at the time the audit event occurred.

Table 50. Audit Record Layout for EXECUTE Events (continued)

NAME	FORMAT	DESCRIPTION
Package Version	VARCHAR(164)	Version of the package in use at the time the audit event occurred.
Local Transaction ID	VARCHAR(10) FOR BIT DATA	The local transaction ID in use at the time the audit event occurred. This is the SQLU_TID structure that is part of the transaction logs.
Global Transaction ID	VARCHAR(30) FOR BIT DATA	The global transaction ID in use at the time the audit event occurred. This is the data field in the SQLP_GXID structure that is part of the transaction logs
UOW ID	BIGINT	The unit of work identifier in which an activity originates. This value is unique within an application ID for each unit of work.
Activity ID	BIGINT	The unique activity ID within the unit of work.
Statement Invocation ID	BIGINT	The identifier (ID) of the routine invocation in which the SQL statement was run. The value indicates the number of routine invocations at the current nesting level that occurred while that level was active in the application. You can use this element, along with Statement Nesting Level, to uniquely identify an invocation of a particular SQL statement.
Statement Nesting Level	BIGINT	The level of nesting or recursion in effect when the statement was being run; each level of nesting corresponds to nested or recursive invocation of a stored procedure or user-defined function (UDF).
Activity Type	VARCHAR(32)	The type of activity. Possible values are: • READ_DML • WRITE_DML • DDL • CALL • NONE
Statement Text	CLOB(8M)	Text of the SQL or XQuery statement, if applicable.

Table 50. Audit Record Layout for EXECUTE Events (continued)

NAME	FORMAT	DESCRIPTION
Statement Isolation Level	CHAR(8)	The isolation value in effect for the statement while it was being run. Possible values are: • NONE (no isolation specified) • UR (uncommitted read) • CS (cursor stability) • RS (read stability) • RR (repeatable read)
Compilation Environment Description	BLOB(8K)	The compilation environment used when compiling the SQL statement. You can provide this element as input to the COMPILATION_ENV table function, or to the SET COMPILATION ENVIRONMENT SQL statement
Rows Modified	INTEGER	Contains the total number of rows deleted, inserted, or updated as a result of both: • The enforcement of constraints after a successful delete operation • The processing of triggered SQL statements from activated triggers If compound SQL is invoked, contains an accumulation of the number of such rows for all sub-statements. In some cases, when an error is encountered, this field contains a negative value that is an internal error pointer. This value is equivalent to the sqlerrd(5) field of the SQLCA.
Rows Returned	BIGINT	Contains the total number of rows returned by the statement.
Savepoint ID	BIGINT	The Savepoint ID in effect for the statement while it is being run. If the Audit Event is SAVEPOINT, RELEASE_SAVEPOINT or ROLLBACK_SAVEPOINT, then the Savepoint ID is the save point that is being set, released, or rolled back to, respectively.

Table 50. Audit Record Layout for EXECUTE Events (continued)

NAME	FORMAT	DESCRIPTION
Statement Value Index	INTEGER	The position of the input parameter marker or host variable used in the SQL statement.
Statement Value Type	CHAR(16)	A string representation of the type of a data value associated with the SQL statement. INTEGER or CHAR are examples of possible values.
Statement Value Data	CLOB(128K)	A string representation of a data value to the SQL statement. LOB, LONG, XML, and structured type parameters are not present. Date, time, and timestamp fields are recorded in ISO format.

Chapter 10. Working with operating system security

Operating systems provide security features that you can use to support security for your database installation.

DB2 and Windows security

A Windows domain is an arrangement of client and server computers referenced by a specific and unique name; and, that share a single user accounts database called the Security Access Manager (SAM). One of the computers in the domain is the domain controller. The domain controller manages all aspects of user-domain interactions.

The domain controller uses the information in the domain user accounts database to authenticate users logging onto domain accounts. For each domain, one domain controller is the primary domain controller (PDC). Within the domain, there may also be backup domain controllers (BDC) which authenticate user accounts when there is no primary domain controller or the primary domain controller is not available. Backup domain controllers hold a copy of the Windows Security Account Manager (SAM) database which is regularly synchronized against the master copy on the PDC.

User accounts, user IDs, and passwords only need to be defined at the primary domain controller to be able to access domain resources.

Note: Two-part user IDs are supported by the CONNECT statement and the ATTACH command. The qualifier of the SAM-compatible user ID is a name of the style 'Domain\User' which has a maximum length of 15 characters.

During the setup procedure when a Windows server is installed, you may select to create:

- A primary domain controller in a new domain
- A backup domain controller in a known domain
- A stand-alone server in a known domain.

Selecting "controller" in a new domain makes that server the primary domain controller.

The user may log on to the local machine, or when the machine is installed in a Windows Domain, the user may log on to the Domain. To authenticate the user, DB2 checks the local machine first, then the Domain Controller for the current Domain, and finally any Trusted Domains known to the Domain Controller.

To illustrate how this works, suppose that the DB2 instance requires Server authentication. The configuration is as follows:

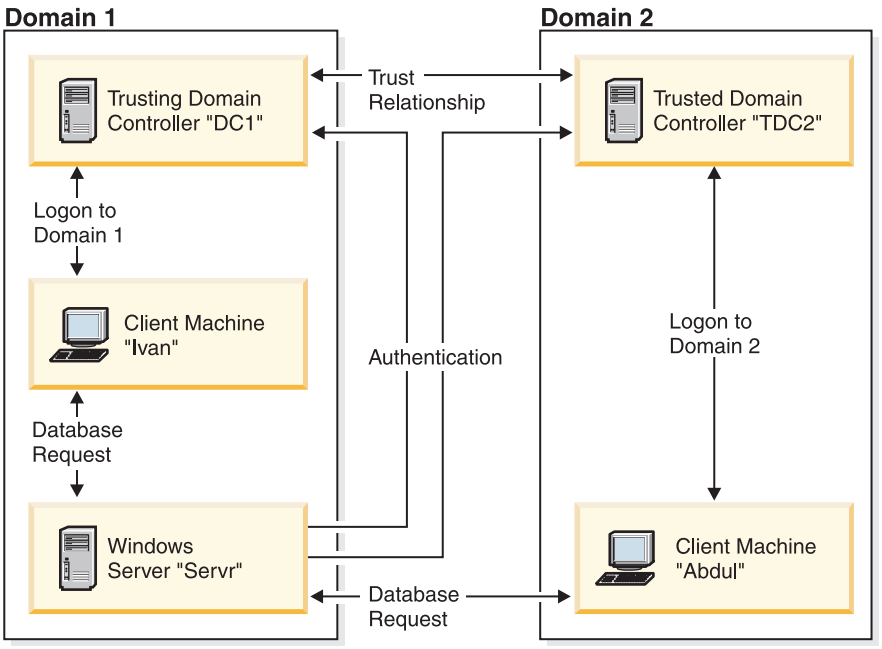


Figure 5. Authentication Using Windows Domains

Each machine has a security database, Security Access Management (SAM). DC1 is the domain controller, in which the client machine, Ivan, and the DB2 server, Servr, are enrolled. TDC2 is a trusted domain for DC1 and the client machine, Abdul, is a member of TDC2's domain.

Authentication scenarios

A scenario with server authentication (Windows)

1. Abdul logs on to the TDC2 domain (that is, he is known in the TDC2 SAM database).
2. Abdul then connects to a DB2 database that is cataloged to reside on SRV3:

   ```
   db2 connect to remotedb user Abdul using fredpw
   ```

3. SRV3 determines where Abdul is known. The API that is used to find this information first searches the local machine (SRV3) and then the domain controller (DC1) before trying any trusted domains. Username Abdul is found on TDC2. This search order requires a single namespace for users and groups.
4. SRV3 then:
   a. Validates the username and password with TDC2.
   b. Finds out whether Abdul is an administrator by asking TDC2.
   c. Enumerates all Abdul's groups by asking TDC2.

A scenario with client authentication and a Windows client machine

1. Dale, the administrator, logs on to SRV3 and changes the authentication for the database instance to Client:

   ```
   db2 update dbm cfg using authentication client
   db2stop
   db2start
   ```

2. Ivan, at a Windows client machine, logs on to the DC1 domain (that is, he is known in the DC1 SAM database).

3. Ivan then connects to a DB2 database that is cataloged to reside on SRV3:
   ```
   DB2 CONNECT to remotedb user Ivan using johnpw
   ```
4. Ivan's machine validates the username and password. The API used to find this information first searches the local machine (Ivan) and then the domain controller (DC1) before trying any trusted domains. Username Ivan is found on DC1.
5. Ivan's machine then validates the username and password with DC1.
6. SRV3 then:
   a. Determines where Ivan is known.
   b. Finds out whether Ivan is an administrator by asking DC1.
   c. Enumerates all Ivan's groups by asking DC1.

Note: Before attempting to connect to the DB2 database, ensure that DB2 Security Service has been started. The Security Service is installed as part of the Windows installation. DB2 is then installed and "registered" as a Windows service however, it is not started automatically. To start the DB2 Security Service, enter the `NET START DB2NTSECSERVER` command.

Support for global groups (on Windows)

The DB2 database system supports global groups.

To use global groups, you must include global groups inside a local group. When the DB2 database manager enumerates all the groups that a person is a member of, it also lists the local groups that the user is a member of indirectly (by the virtue of being in a global group that is itself a member of one or more local groups).

Global groups are used in two possible situations:

- Included inside a local group. Permission must be granted to this local group.
- Included on a domain controller. Permission must be granted to the global group.

Using a backup domain controller with DB2 database systems

If the server you use for your DB2 database system also acts as a backup domain controller, you can improve DB2 database performance and reduce network traffic if you configure the DB2 database system to use the backup domain controller.

You specify the backup domain controller to the DB2 database system by setting the DB2DMNBCKCTLR registry variable.

If you know the name of the domain for which DB2 database server is the backup domain controller, use:

```
db2dmnbckctlr=<domain_name>
```

where `domain_name` must be in upper case.

To have the DB2 database system determine the domain for which the local machine is a backup domain controller, use:

```
DB2DMNBCKCTLR=?
```

Note: The DB2 database manager does not use an existing backup domain controller by default because a backup domain controller can get out-of-sync with the primary domain controller, causing a security exposure. Domain controllers get

out-of-sync when the primary domain controller's security database is updated but the changes are not propagated to a backup domain controller. This can happen if there are network latencies or if the computer browser service is not operational.

User authentication with DB2 on Windows

User name and group name restrictions (Windows)

There are a few limitations that are specific to the Windows environment. Be aware that general DB2 object naming rules also apply.

- User names under Windows are not case sensitive; however, passwords are case sensitive.
- User names and group names can be a combination of upper- and lowercase characters. However, they are usually converted to uppercase when used within the DB2 database. For example, if you connect to the database and create the table schema1.table1, this table is stored as SCHEMA1.TABLE1 within the database. (If you wish to use lowercase object names, issue commands from the command line processor, enclosing the object names in quotation marks, or use third-party ODBC front-end tools.)
- A user can not belong to more than 64 groups.
- The DB2 database manager supports a single namespace. That is, when running in a trusted domains environment, you should not have a user account of the same name that exists in multiple domains, or that exists in the local SAM of the server machine and in another domain.

Groups and user authentication on Windows

Users are defined on Windows by creating user accounts using the Windows administration tool called the "User Manager". An account containing other accounts, also called members, is a group.

Groups give Windows administrators the ability to grant rights and permissions to the users within the group at the same time, without having to maintain each user individually. Groups, like user accounts, are defined and maintained in the Security Access Manager (SAM) database.

There are two types of groups:

- Local groups. A local group can include user accounts created in the local accounts database. If the local group is on a machine that is part of a domain, the local group can also contain domain accounts and groups from the Windows domain. If the local group is created on a workstation, it is specific to that workstation.
- Global groups. A global group exists only on a domain controller and contains user accounts from the domain's SAM database. That is, a global group can only contain user accounts from the domain on which it is created; it cannot contain any other groups as members. A global group can be used in servers and workstations of its own domain, and in trusting domains.

Trust relationships between domains on Windows

Trust relationships are an administration and communication link between two domains. A trust relationship between two domains enables user accounts and global groups to be used in a domain other than the domain where the accounts are defined.

Account information is shared to validate the rights and permissions of user accounts and global groups residing in the trusted domain without being

authenticated. Trust relationships simplify user administration by combining two or more domains into an single administrative unit.

There are two domains in a trust relationship:

- The trusting domain. This domain trusts another domain to authenticate users for them.
- The trusted domain. This domain authenticates users on behalf of (in trust for) another domain.

Trust relationships are not transitive. This means that explicit trust relationships need to be established in each direction between domains. For example, the trusting domain may not necessarily be a trusted domain.

DB2 database system and Windows security service

In the DB2 database system, the authentication of user names and passwords is integrated with the DB2 System Controller.

The Security Service is only required when a client is connected to a server that is configured for authentication CLIENT.

Installing DB2 on a backup domain controller

In a Windows environment a user can be authenticated at either a primary or a backup controller. This feature is important in large distributed LANs with one central primary domain controller and one or more backup domain controllers (BDC) at each site. Users can be authenticated on the backup domain controller at their site instead of requiring a call to the primary domain controller (PDC) for authentication.

The advantage of having a backup domain controller, in this case, is that users are authenticated faster and the LAN is not as congested as it would have been had there been no BDC.

Authentication can occur at the BDC under the following conditions:

- The DB2 server for Windows is installed on the backup domain controller.
- The DB2DMNBCKCTLR profile registry variable is set appropriately.

If the DB2DMNBCKCTLR profile registry variable is not set or is set to blank, the DB2 server performs authentication at the primary domain controller.

The only valid declared settings for DB2DMNBCKCTLR are ″?″ or a domain name.

If the DB2DMNBCKCTLR profile registry variable is set to a question mark (DB2DMNBCKCTLR=?) then the DB2 server will perform its authentication on the backup domain controller under the following conditions:

- The cachedPrimaryDomain is a registry value set to the name of the domain to which this machine belongs. (You can find this setting under **HKEY_LOCAL_MACHINE--> Software--> Microsoft--> Windows NT--> Current Version--> WinLogon**.)
- The Server Manager shows the backup domain controller as active and available. (That is, the icon for this machine is not greyed out.)
- The registry for the DB2 server indicates that the system is a backup domain controller on the specified domain.

Under normal circumstances the setting DB2DMNBCKCTLR=? will work; however, it will not work in all environments. The information supplied about the servers on the domain is dynamic, and Computer Browser must be running to keep this information accurate and current. Large LANs may not be running Computer Browser and therefore Server Manager's information may not be current. In this case, there is a second method to tell the DB2 server to authenticate at the backup domain controller: set DB2DMNBCKCTLR=*xxx* where *xxx* is the Windows domain name for the DB2 server. With this setting, authentication will occur on the backup domain controller based on the following conditions:

- The cachedPrimaryDomain is a registry value set to the name of the domain to which this machine belongs. (You can find this setting under **HKEY_LOCAL_MACHINE--> Software--> Microsoft--> Windows NT--> Current Version--> WinLogon**.)
- The machine is configured as a backup domain controller for the specified domain. (If the machine is set up as a backup domain controller for another domain, this setting will result in an error.)

Authentication with groups and domain security (Windows)

The DB2 database system allows you to specify either a local group or a global group when granting privileges or defining authority levels.

A user is determined to be a member of a group if the user's account is defined explicitly in the local or global group, or implicitly by being a member of a global group defined to be a member of a local group.

The DB2 database manager supports the following types of groups:

- Local groups
- Global groups
- Global groups as members of local groups.

  The DB2 database manager Renumerates the local and global groups that the user is a member of, using the security database where the user was found. The DB2 database system provides an override that forces group enumeration to occur on the local Windows server where the DB2 database is installed, regardless of where the user account was found. This override can be achieved using the following commands:

  – For global settings:

    ```
    db2set -g DB2_GRP_LOOKUP=local
    ```

  – For instance settings:

    ```
    db2set -i <instance_name> DB2_GRP_LOOKUP=local
    ```

    After issuing this command, you must stop and start the DB2 database instance for the change to take effect. Then create local groups and include domain accounts or global groups in the local group.

  To view all DB2 profile registry variables that are set, type

  ```
  db2set -all
  ```

If the DB2_GRP_LOOKUP profile registry variable is set to local, then DB2 database tries to enumerate the user's groups on the local machine only. If the user is not defined as a member of a local or global group, then group enumeration fails. DB2 does **not** try to enumerate the user's groups on another machine in the domain or on the domain controllers.

If the DB2_GRP_LOOKUP profile registry variable is not set then:

1. The DB2 database system first tries to find the user on the same machine.
2. If the user name is defined locally, the user is authenticated locally.
3. If the user is not found locally, the DB2 database system attempts to find the user name on it domain, and then on trusted domains.

If the DB2 database manager is running on a machine that is a primary or backup domain controller in the resource domain, it is able to locate any domain controller in any trusted domain. This occurs because the names of the domains of backup domain controllers in trusted domains are only known if you are a domain controller.

If the DB2 database manager is not running on a domain controller, then you should issue:

```
db2set -g DB2_GRP_LOOKUP=DOMAIN
```

This command tells the DB2 database system to use a domain controller in its own domain to find the name of a domain controller in the accounts domain. That is, when a DB2 database finds out that a particular user account is defined in domain x, rather than attempting to locate a domain controller for domain x, it sends that request to a domain controller in its own domain. The name of the domain controller in the account domain will be found and returned to the machine the DB2 database is running on. There are two advantages to this method:

1. The nearest domain controller is found when the primary domain controller is unavailable.
2. The nearest domain controller is found when the primary domain controller is geographically remote.

Authentication using an ordered domain list

User IDs may be defined more than once in a trusted domain forest. A trusted domain forest is a collection of domains that are interrelated through a network.

It is possible for a user on one domain to have the same user ID as that for another user on a different domain. This may cause difficulties when attempting to do any of the following:

- Authenticating multiple users having the same user ID but on different domains.
- Group lookup for the purposes of granting and revoking privileges based on groups.
- Validation of passwords.
- Control of network traffic.

To prevent difficulties arising from the possibility of multiple users with the same user ID across a domain forest, you should use an ordered domain list as defined using the db2set and the registry variable DB2DOMAINLIST. When setting the order, the domains to be included in the list are separated by a comma. You must make a conscious decision regarding the order that the domains are searched when authenticating users.

Those user IDs that are present on domains further down the domain list will have to be renamed by you if they are to be authenticated for access.

Control of access can be done through the domain list. For example, if the domain of a user is not in the list, the user will not be allowed to connect.

Note: The DB2DOMAINLIST registry variable is effective only when CLIENT authentication is set in the database manager configuration and is needed if a single signon from a Windows desktop is required in a Windows domain environment. DB2DOMAINLIST is supported by some versions of DB2 servers however DB2DOMAINLIST will not be enforced if neither the client nor the server are in a Windows environment.

Domain security support (Windows)

The following example illustrates how the DB2 database management system can support Windows domain security. The connection works because the user name and local group are on the same domain.

The connection works in the following scenario because the user name and local or global group are on the same domain.

Note that the user name and local or global group do not need to be defined on the domain where the database server is running, but they must be on the same domain as each other.

Table 51. Successful Connection Using a Domain Controller

Domain1	Domain2
A trust relationship exists with Domain2.	• A trust relationship exists with Domain1. • The local or global group `grp2` is defined. • The user name `id2` is defined. • The user name `id2` is part of `grp2`.

The DB2 server runs in this domain. The following DB2 commands are issued from it:
`REVOKE CONNECT ON db FROM public`
`GRANT CONNECT ON db TO GROUP grp2`
`CONNECT TO db USER id2`	

The local or global domain is scanned but `id2` is not found. Domain security is scanned.	
	The user name `id2` is found on this domain. DB2 gets additional information about this user name (that is, it is part of the group `grp2`).
The connection works because the user name and local or global group are on the same domain.	

Acquiring Windows users' group information using an access token

An access token is an object that describes the security context of a process or thread. The information in an access token includes the identity and privileges of the user account associated with the process or thread.

When you log on, the system verifies your password by comparing it with information stored in a security database. If the password is authenticated, the system produces an access token. Every process run on your behalf uses a copy of this access token.

An access token can also be acquired based on cached credentials. After you have been authenticated to the system, your credentials are cached by the operating system. The access token of the last logon can be referenced in the cache when it is not possible to contact the domain controller.

The access token includes information about all of the groups you belong to: local groups and various domain groups (global groups, domain local groups, and universal groups).

Note: Group lookup using client authentication is not supported using a remote connection even though access token support is enabled.

To enable access token support, you must use the db2set command to update the DB2_GRP_LOOKUP registry variable. Your choices when updating this registry variable include:

- TOKEN

  This choice enables access token support to lookup all groups that the user belongs to at both the local machines as well as at the location where the user account is defined (if the account is defined at the domain).
- TOKENLOCAL

  This choice enables access token support to lookup all local groups that the user belongs to on the DB2 database server.
- TOKENDOMAIN

  This choice enables access token support to lookup all groups that the user belongs to at the location where the user account is defined. This location is typically either at the domain or local to the DB2 database server.

You should consider using the DB2_GRP_LOOKUP registry variable and specify the group lookup location to indicate where the DB2 database system should look up groups using the conventional group enumeration methodology. For example,

```
db2set DB2_GRP_LOOKUP=LOCAL,TOKENLOCAL
```

This enables the access token support for enumerating local groups.

```
db2set DB2_GRP_LOOKUP=,TOKEN
```

This enables the access token support for enumerating groups at both the local machines as well as the location where the user ID is defined (if the account is defined at the domain).

```
db2set DB2_GRP_LOOKUP=DOMAIN,TOKENDOMAIN
```

This enables the access token support for enumerating domain groups at the location where the user ID is defined.

Access token support can be enabled with all authentications types except CLIENT authentication.

Windows platform security considerations for users

System administration (SYSADM) authority is granted to any valid DB2 database user account which belongs to the local Administrators group on the machine where the account is defined.

By default in a Windows domain environment, only domain users that belong to the Administrators group at the Domain Controller have SYSADM authority on an instance. Since DB2 always performs authorization at the machine where the account is defined, adding a domain user to the local Administrators group on the server does not grant the domain user SYSADM authority to the group.

Note: In a domain environment such as is found in Windows, DB2 only authenticates the first 64 groups that meet the requirements and restrictions, and to which a user ID belongs. You may have more than 64 groups.

To avoid adding a domain user to the Administrators group at the primary domain controller (PDC), you should create a global group and add the users (both domain and local) that you want to grant SYSADM authority. To do this, enter the following commands:

```
DB2STOP
DB2 UPDATE DBM CFG USING SYSADM_GROUP global_group
DB2START
```

Windows local system account support

On Windows platforms (except Windows ME), the DB2 database system supports applications running under the context of the local system account (LSA) with local implicit connection.

Developers writing applications to be run under this account need to be aware that the DB2 database system has restrictions on objects with schema names starting with "SYS". Therefore if your applications contain DDLs that create DB2 database objects, they should be written such that:

- For static queries, they should be bound with a value for the QUALIFIER options other than the default one.
- For dynamic queries, the objects to be created should be explicitly qualified with a schema name supported by the DB2 database manager, or the CURRENT SCHEMA register must be set to a schema name supported by the DB2 database manager.

Group information for the LSA is gathered at the first group lookup request after the DB2 database instance is started and is not refreshed until the instance is restarted.

Note: Applications running under the context of the local system account (LSA) are supported on all Windows platforms, except Windows ME.

Extended Windows security using DB2ADMNS and DB2USERS groups

For the server version of the DB2 database manager, extended security is implicitly *enabled* by default. However, for the client version, extended security is implicitly *disabled* by default; you *must* explicitly select extended security during installation to have it enabled.

To enable extended security during DB2 installation on a client, select the **Enable operating system security** check box on the **Enable operating system security for the DB2 object** panel. The installer creates two new groups, DB2ADMNS and DB2USERS. DB2ADMNS and DB2USERS are the default group names; optionally, you can specify different names for these groups at installation time (if you select silent install, you can change these names within the install response file). If you choose to use groups that already exist on your system, be aware that the privileges of these groups will be modified. They will be given the privileges, as required, listed in the table, below. It is important to understand that these groups are used for protection at the *operating-system* level and are in no way associated with DB2 authority levels, such as SYSADM, SYSMAINT, and SYSCTRL. However, instead of using the default Administrator's group, your database administrator

can use the DB2ADMNS group for one or all of the DB2 authority levels, at the discretion of the installer or administrator. If you are specifying a SYSADM group, then that should be the DB2ADMNS group. This can be established during installation or subsequently, by an administrator.

Note: You can specify your DB2 Administrators Group (DB2ADMNS or the name you chose during installation) and DB2 Users Group (DB2USERS or the name you chose during installation) either as local groups or as domain groups. Both groups must be of the same type, so either both local or both domain.

If you change the computer name, and the computer groups DB2ADMNS and DB2USERS are local computer groups, you must update the DB2_ADMINGROUP and DB2_USERSGROUP global registries. To update the registry variables after renaming and restarting the computer run the following command:

1. Open a command prompt.
2. Run the db2extsec command to update security settings:

   ```
   db2extsec -a new computer name\DB2ADMNS -u new computer name\DB2USERS
   ```

Note: If extended security is enabled in DB2 database products on Windows Vista, only users that belong to the DB2ADMNS group can run the graphical DB2 administration tools. In addition, members of the DB2ADMNS group need to launch the tools with full administrator privileges. This is accomplished by right-clicking on the shortcut and then choosing "Run as administrator".

Abilities acquired through the DB2ADMNS and DB2USERS groups

The DB2ADMNS and DB2USERS groups provide members with the following abilities:

- DB2ADMNS

  Full control over all DB2 objects (see the list of protected objects, below)
- DB2USERS

  Read and Execute access for all DB2 objects located in the installation and instance directories, but no access to objects under the database system directory and limited access to IPC resources

  For certain objects, there may be additional privileges available, as required (for example, write privileges, add or update file privileges, and so on). Members of this group have no access to objects under the database system directory.

  Note: The meaning of Execute access depends on the object; for example, for a **.dll** or **.exe** file having Execute access means you have authority to execute the file, however, for a directory it means you have authority to traverse the directory.

Ideally, all DB2 administrators should be members of the DB2ADMNS group (as well as being members of the local Administrators group), but this is not a strict requirement. Everyone else who requires access to the DB2 database system *must* be a member of the DB2USERS group. To add a user to one of these groups:

1. Launch the Users and Passwords Manager tool.
2. Select the user name to add from the list.
3. Click Properties. In the Properties window, click the Group membership tab.
4. Select the Other radio button.
5. Select the appropriate group from the drop-down list.

Adding extended security after installation (db2extsec command)

If the DB2 database system was installed without extended security enabled, you can enable it by executing the command **db2extsec** (called **db2secv82** in earlier releases). To execute the **db2extsec** command you must be a member of the local Administrators group so that you have the authority to modify the ACL of the protected objects.

You can run the **db2extsec** command multiple times, if necessary, however, if this is done, you cannot disable extended security unless you issue the **db2extsec –r** command immediately after *each* execution of **db2extsec**.

Removing extended security

CAUTION:
Do not remove extended security after it has been enabled unless absolutely necessary.

You can remove extended security by running the command **db2extsec -r**, however, this will only succeed if no other database operations (such as creating a database, creating a new instance, adding table spaces, and so on) have been performed after enabling extended security. The safest way to remove the extended security option is to uninstall the DB2 database system, delete all the relevant DB2 directories (including the database directories) and then reinstall the DB2 database system without extended security enabled.

Protected objects

The *static* objects that can be protected using the DB2ADMNS and DB2USERS groups are:

- File system
  - File
  - Directory
- Services
- Registry keys

The *dynamic* objects that can be protected using the DB2ADMNS and DB2USERS groups are:

- IPC resources, including:
  - Pipes
  - Semaphores
  - Events
- Shared memory

Privileges owned by the DB2ADMNS and DB2USERS groups

The privileges assigned to the DB2ADMNS and DB2USERS groups are listed in the following table:

Table 52. Privileges for DB2ADMNS and DB2USERS groups

Privilege	DB2ADMNS	DB2USERS	Reason
Create a token object (SeCreateTokenPrivilege)	Y	N	Token manipulation (required for certain token manipulation operations and used in authentication and authorization)
Replace a process level token (SeAssignPrimaryTokenPrivilege)	Y	N	Create process as another user
Increase quotas (SeIncreaseQuotaPrivilege)	Y	N	Create process as another user
Act as part of the operating system (SeTcbPrivilege)	Y	N	LogonUser (required prior to Windows XP in order to execute the LogonUser API for authentication purposes)
Generate security audits (SeSecurityPrivilege)	Y	N	Manipulate audit and security log
Take ownership of files or other objects (SeTakeOwnershipPrivilege)	Y	N	Modify object ACLs
Increase scheduling priority (SeIncreaseBasePriorityPrivilege)	Y	N	Modify the process working set
Backup files and directories (SeBackupPrivilege)	Y	N	Profile/Registry manipulation (required to perform certain user profile and registry manipulation routines: LoadUserProfile, RegSaveKey(Ex), RegRestoreKey, RegReplaceKey, RegLoadKey(Ex))
Restore files and directories (SeRestorePrivilege)	Y	N	Profile/Registry manipulation (required to perform certain user profile and registry manipulation routines: LoadUserProfile, RegSaveKey(Ex), RegRestoreKey, RegReplaceKey, RegLoadKey(Ex))
Debug programs (SeDebugPrivilege)	Y	N	Token manipulation (required for certain token manipulation operations and used in authentication and authorization)
Manage auditing and security log (SeAuditPrivilege)	Y	N	Generate auditing log entries
Log on as a service (SeServiceLogonRight)	Y	N	Run DB2 as a service
Access this computer from the network (SeNetworkLogonRight)	Y	Y	Allow network credentials (allows the DB2 database manager to use the LOGON32_LOGON_NETWORK option to authenticate, which has performance implications)
Impersonate a client after authentication (SeImpersonatePrivilege)	Y	N	Client impersonation (required for Windowsto allow use of certain APIs to impersonate DB2 clients: ImpersonateLoggedOnUser, ImpersonateSelf, RevertToSelf, and so on)
Lock pages in memory (SeLockMemoryPrivilege)	Y	N	Large Page support
Create global objects (SeCreateGlobalPrivilege)	Y	Y	Terminal Server support (required on Windows)

Considerations for Vista: User Access Control feature

The User Access Control (UAC) feature of Windows Vista impacts the DB2 database system in the following ways.

Starting applications with full administrative privileges

On Vista, by default, applications start with only standard user rights, even if the user is a local administrator. To start an application with further privileges, you need to launch the command from a command window that is running with full administrative privileges. The DB2 installation process creates a shortcut called ″Command window - Administrator″ specifically for Vista users. It is recommended that you launch this shortcut if you want to run administrative commands.

If you do not have full administrative privileges and you attempt to perform DB2 administration tasks from a command prompt or graphical tool on Windows Vista, you can encounter various error messages implying that your access is denied and the tasks will fail to complete successfully.

To determine whether the action you are performing is considered to be an administration task, check whether any of the following are true:

- It requires SYSADM, SYSCTRL or SYSMAINT authority
- It modifies registry keys under the HKLM branch in the registry
- It writes to the directories under the Program Files directory

For example, the following actions are all considered to be administration tasks:

- Creating and dropping DB2 instances
- Starting and stopping DB2 instances
- Creating databases
- Updating database manager configuration parameters or DB2 Administration Server (DAS) configuration parameters
- Updating CLI configuration parameters and configuring system data source names (DSN)
- Starting the DB2 trace facility
- Running the db2pd utility
- Changing DB2 profile registry variables

To resolve the problem, you must perform DB2 administration tasks from a command prompt or graphical tool that is running with full administrator privileges. To launch a command prompt or graphical tool with full administrator privileges, right-click on the shortcut and then select **Run as administrator**.

Note: If extended security is enabled, you also need to be a member of the DB2ADMNS group in order to launch the graphical administration tools (such as the Command Editor or Control Center).

User data location

User data (for example, files under instance directories) is stored in ProgramData\IBM\DB2*copy_name*, where *copy_name* is the name of the DB2 copy (by default, DB2COPY1 is the name of the first copy installed). On Windows versions other than Vista, user data is stored in Documents and Settings\All Users\Application Data\IBM\DB2*copy_name*.

DB2 and UNIX security

UNIX platform security considerations for users

The DB2 database does not support root acting directly as a database administrator. You should use su - <instance owner> as the database administrator.

For security reasons, in general, do not use the instance name as the Fenced ID. However, if you are not planning to use fenced UDFs or stored procedures, you can set the Fenced ID to the instance name instead of creating another user ID.

The recommendation is to create a user ID that is recognized as being associated with this group. The user for fenced UDFs and stored procedures is specified as a parameter of the instance creation script (db2icrt ... -u <FencedID>). This is not required if you install the DB2 Clients or the DB2 Software Developer's Kit.

Location of the instance directory

For root installations on Linux and UNIX, the db2icrt command creates the main SQL library (sqllib) directory under the home directory of the instance owner.

On Windows operating systems, the instance directory is located in the /sqllib sub-directory, in the directory where the DB2 database system was installed.

DB2 and Linux security

Change password support (Linux)

DB2 database products provide support for changing passwords on Linux operating systems.

This support is implemented through the use of security plug-in libraries called IBMOSchgpwdclient.so and IBMOSchgpwdserver.so.

To enable password change support on Linux, set the database manager configuration parameter **CLNT_PW_PLUGIN** to IBMOSchgpwdclient and **SRVCON_PW_PLUGIN** to IBMOSchgpwdserver.

You must also create a PAM configuration file called "db2" in the /etc/pam.d directory.

Deploying a change password plug-in (Linux)

To enable support for changing passwords in DB2 database products on Linux, you must configure the DB2 instance to use the security plug-ins IBMOSchgpwdclient and IBMOSchgpwdserver.

The plug-in libraries are located in the following directories:

- *INSTHOME*/sqllib/*securityXX*/plugin/client/IBMOSchgpwdclient.so
- *INSTHOME*/sqllib/*securityXX*/plugin/server/IBMOSchgpwdserver.so

where *INSTHOME* is the home directory of the instance owner and *securityXX* is either security32 or security64, depending on the bit-width of the instance.

To deploy the security plug-ins in a DB2 instance, perform the following steps:

1. Log in as a user with root authority.

2. Create a PAM configuration file: /etc/pam.d/db2

   Ensure that the file contains the appropriate set of rules, as defined by your system administrator. For example:

   ```
   auth     required pam_unix2.so    nullok
   account  required pam_unix2.so
   password required pam_pwcheck.so  nullok tries=1
   password required pam_unix2.so    nullok use_authtok use_first_pass
   session  required pam_unix2.so
   ```

3. Enable the security plug-ins in the DB2 instance:
   a. Update the database manager configuration parameter **SRVCON_PW_PLUGIN** with the value IBMOSchgpwdserver:

      ```
      db2 update dbm cfg using srvcon_pw_plugin IBMOSchgpwdserver
      ```

   b. Update the database manager configuration parameter **CLNT_PW_PLUGIN** with the value IBMOSchgpwdclient:

      ```
      db2 update dbm cfg using CLNT_PW_PLUGIN IBMOSchgpwdclient
      ```

   c. Ensure that either the database manager configuration parameter **SRVCON_AUTH** is set to a value of CLIENT, SERVER, SERVER_ENCRYPT, DATA_ENCRYPT, or DATA_ENCRYPT_CMP, or the database manager configuration parameter **SRVCON_AUTH** is set to a value of NOT_SPECIFIED and **AUTHENTICATION** is set to a value of CLIENT, SERVER, SERVER_ENCRYPT, DATA_ENCRYPT, or DATA_ENCRYPT_CMP.

Appendix A. Overview of the DB2 technical information

DB2 technical information is available through the following tools and methods:

- DB2 Information Center
  - Topics (Task, concept and reference topics)
  - Help for DB2 tools
  - Sample programs
  - Tutorials
- DB2 books
  - PDF files (downloadable)
  - PDF files (from the DB2 PDF DVD)
  - printed books
- Command line help
  - Command help
  - Message help

Note: The DB2 Information Center topics are updated more frequently than either the PDF or the hard-copy books. To get the most current information, install the documentation updates as they become available, or refer to the DB2 Information Center at ibm.com®.

You can access additional DB2 technical information such as technotes, white papers, and IBM Redbooks® publications online at ibm.com. Access the DB2 Information Management software library site at http://www.ibm.com/software/data/sw-library/.

Documentation feedback

We value your feedback on the DB2 documentation. If you have suggestions for how to improve the DB2 documentation, send an email to db2docs@ca.ibm.com. The DB2 documentation team reads all of your feedback, but cannot respond to you directly. Provide specific examples wherever possible so that we can better understand your concerns. If you are providing feedback on a specific topic or help file, include the topic title and URL.

Do not use this email address to contact DB2 Customer Support. If you have a DB2 technical issue that the documentation does not resolve, contact your local IBM service center for assistance.

DB2 technical library in hardcopy or PDF format

The following tables describe the DB2 library available from the IBM Publications Center at www.ibm.com/shop/publications/order. English DB2 Version 9.5 manuals in PDF format and translated versions can be downloaded from www.ibm.com/support/docview.wss?rs=71&uid=swg2700947.

Although the tables identify books available in print, the books might not be available in your country or region.

Table 53. DB2 technical information

Name	Form Number	Available in print
Administrative API Reference	SC23-5842-00	Yes
Administrative Routines and Views	SC23-5843-00	No
Call Level Interface Guide and Reference, Volume 1	SC23-5844-00	Yes
Call Level Interface Guide and Reference, Volume 2	SC23-5845-00	Yes
Command Reference	SC23-5846-00	Yes
Data Movement Utilities Guide and Reference	SC23-5847-00	Yes
Data Recovery and High Availability Guide and Reference	SC23-5848-00	Yes
Data Servers, Databases, and Database Objects Guide	SC23-5849-00	Yes
Database Security Guide	SC23-5850-00	Yes
Developing ADO.NET and OLE DB Applications	SC23-5851-00	Yes
Developing Embedded SQL Applications	SC23-5852-00	Yes
Developing Java Applications	SC23-5853-00	Yes
Developing Perl and PHP Applications	SC23-5854-00	No
Developing User-defined Routines (SQL and External)	SC23-5855-00	Yes
Getting Started with Database Application Development	GC23-5856-00	Yes
Getting Started with DB2 installation and administration on Linux and Windows	GC23-5857-00	Yes
Internationalization Guide	SC23-5858-00	Yes
Message Reference, Volume 1	GI11-7855-00	No
Message Reference, Volume 2	GI11-7856-00	No
Migration Guide	GC23-5859-00	Yes
Net Search Extender Administration and User's Guide **Note:** The content of this document is not included in the DB2 Information Center	SC23-8509-00	Yes
Partitioning and Clustering Guide	SC23-5860-00	Yes
Query Patroller Administration and User's Guide	SC23-8507-00	Yes
Quick Beginnings for IBM Data Server Clients	GC23-5863-00	No
Quick Beginnings for DB2 Servers	GC23-5864-00	Yes

Table 53. DB2 technical information (continued)

Name	Form Number	Available in print
Spatial Extender and Geodetic Data Management Feature User's Guide and Reference	SC23-8508-00	Yes
SQL Reference, Volume 1	SC23-5861-00	Yes
SQL Reference, Volume 2	SC23-5862-00	Yes
System Monitor Guide and Reference	SC23-5865-00	Yes
Text Search Guide	SC23-5866-00	Yes
Troubleshooting Guide	GI11-7857-00	No
Tuning Database Performance	SC23-5867-00	Yes
Visual Explain Tutorial	SC23-5868-00	No
What's New	SC23-5869-00	Yes
Workload Manager Guide and Reference	SC23-5870-00	Yes
pureXML Guide	SC23-5871-00	Yes
XQuery Reference	SC23-5872-00	No

Table 54. DB2 Connect-specific technical information

Name	Form Number	Available in print
Quick Beginnings for DB2 Connect Personal Edition	GC23-5839-00	Yes
Quick Beginnings for DB2 Connect Servers	GC23-5840-00	Yes
DB2 Connect User's Guide	SC23-5841-00	Yes

Table 55. Information Integration technical information

Name	Form Number	Available in print
Information Integration: Administration Guide for Federated Systems	SC19-1020-01	Yes
Information Integration: ASNCLP Program Reference for Replication and Event Publishing	SC19-1018-02	Yes
Information Integration: Configuration Guide for Federated Data Sources	SC19-1034-01	No
Information Integration: SQL Replication Guide and Reference	SC19-1030-01	Yes
Information Integration: Introduction to Replication and Event Publishing	SC19-1028-01	Yes

Ordering printed DB2 books

If you require printed DB2 books, you can buy them online in many but not all countries or regions. You can always order printed DB2 books from your local IBM representative. Keep in mind that some softcopy books on the *DB2 PDF Documentation* DVD are unavailable in print. For example, neither volume of the *DB2 Message Reference* is available as a printed book.

Printed versions of many of the DB2 books available on the DB2 PDF Documentation DVD can be ordered for a fee from IBM. Depending on where you are placing your order from, you may be able to order books online, from the IBM Publications Center. If online ordering is not available in your country or region, you can always order printed DB2 books from your local IBM representative. Note that not all books on the DB2 PDF Documentation DVD are available in print.

Note: The most up-to-date and complete DB2 documentation is maintained in the DB2 Information Center at http://publib.boulder.ibm.com/infocenter/db2luw/v9r5.

To order printed DB2 books:

- To find out whether you can order printed DB2 books online in your country or region, check the IBM Publications Center at http://www.ibm.com/shop/publications/order. You must select a country, region, or language to access publication ordering information and then follow the ordering instructions for your location.
- To order printed DB2 books from your local IBM representative:
  1. Locate the contact information for your local representative from one of the following Web sites:
     - The IBM directory of world wide contacts at www.ibm.com/planetwide
     - The IBM Publications Web site at http://www.ibm.com/shop/publications/order. You will need to select your country, region, or language to the access appropriate publications home page for your location. From this page, follow the "About this site" link.
  2. When you call, specify that you want to order a DB2 publication.
  3. Provide your representative with the titles and form numbers of the books that you want to order. For titles and form numbers, see "DB2 technical library in hardcopy or PDF format" on page 245.

Displaying SQL state help from the command line processor

DB2 returns an SQLSTATE value for conditions that could be the result of an SQL statement. SQLSTATE help explains the meanings of SQL states and SQL state class codes.

To invoke SQL state help, open the command line processor and enter:

```
? sqlstate or ? class code
```

where *sqlstate* represents a valid five-digit SQL state and *class code* represents the first two digits of the SQL state.

For example, `? 08003` displays help for the 08003 SQL state, and `? 08` displays help for the 08 class code.

Accessing different versions of the DB2 Information Center

For DB2 Version 9.5 topics, the DB2 Information Center URL is http://publib.boulder.ibm.com/infocenter/db2luw/v9r5/

For DB2 Version 9 topics, the DB2 Information Center URL is http://publib.boulder.ibm.com/infocenter/db2luw/v9/

For DB2 Version 8 topics, go to the Version 8 Information Center URL at: http://publib.boulder.ibm.com/infocenter/db2luw/v8/

Displaying topics in your preferred language in the DB2 Information Center

The DB2 Information Center attempts to display topics in the language specified in your browser preferences. If a topic has not been translated into your preferred language, the DB2 Information Center displays the topic in English.

- To display topics in your preferred language in the Internet Explorer browser:
  1. In Internet Explorer, click the **Tools** —> **Internet Options** —> **Languages...** button. The Language Preferences window opens.
  2. Ensure your preferred language is specified as the first entry in the list of languages.
     - To add a new language to the list, click the **Add...** button.

       Note: Adding a language does not guarantee that the computer has the fonts required to display the topics in the preferred language.
     - To move a language to the top of the list, select the language and click the **Move Up** button until the language is first in the list of languages.
  3. Clear the browser cache and then refresh the page to display the DB2 Information Center in your preferred language.
- To display topics in your preferred language in a Firefox or Mozilla browser:
  1. Select the button in the **Languages** section of the **Tools** —> **Options** —> **Advanced** dialog. The Languages panel is displayed in the Preferences window.
  2. Ensure your preferred language is specified as the first entry in the list of languages.
     - To add a new language to the list, click the **Add...** button to select a language from the Add Languages window.
     - To move a language to the top of the list, select the language and click the **Move Up** button until the language is first in the list of languages.
  3. Clear the browser cache and then refresh the page to display the DB2 Information Center in your preferred language.

On some browser and operating system combinations, you might have to also change the regional settings of your operating system to the locale and language of your choice.

Updating the DB2 Information Center installed on your computer or intranet server

If you have installed the DB2 Information Center locally, you can download and install updates that IBM might make available.

Updating your locally-installed DB2 Information Center requires that you:

1. Stop the DB2 Information Center on your computer, and restart the Information Center in stand-alone mode. Running the Information Center in stand-alone mode prevents other users on your network from accessing the Information Center, and allows you to download and apply updates.
2. Use the Update feature to see what updates are available. If there are updates that you would like to install, you can use the Update feature to download and install them

   Note: If your environment requires installing the DB2 Information Center updates on a machine that is not connected to the internet, you have to mirror the update site to a local file system using a machine that is connected to the internet and has the DB2 Information Center installed. If many users on your network will be installing the documentation updates, you can reduce the time required for individuals to perform the updates by also mirroring the update site locally and creating a proxy for the update site.
   If update packages are available, use the Update feature to download the packages. However, the Update feature is only available in stand-alone mode.
3. Stop the stand-alone Information Center, and restart the DB2 Information Center on your computer.

Note: On Windows Vista, the commands listed below must be run as an administrator. To launch a command prompt or graphical tool with full administrator privileges, right-click on the shortcut and then select **Run as administrator**.

To update the DB2 Information Center installed on your computer or intranet server:

1. Stop the DB2 Information Center.
   - On Windows, click **Start → Control Panel → Administrative Tools → Services**. Then right-click on **DB2 Information Center** service and select **Stop**.
   - On Linux, enter the following command:
     ```
     /etc/init.d/db2icdv95 stop
     ```
2. Start the Information Center in stand-alone mode.
   - On Windows:
     a. Open a command window.
     b. Navigate to the path where the Information Center is installed. By default, the DB2 Information Center is installed in the <Program Files>\IBM\DB2 Information Center\Version 9.5 directory, where <Program Files> represents the location of the Program Files directory.
     c. Navigate from the installation directory to the doc\bin directory.
     d. Run the help_start.bat file:
        ```
        help_start.bat
        ```
   - On Linux:

a. Navigate to the path where the Information Center is installed. By default, the DB2 Information Center is installed in the /opt/ibm/db2ic/V9.5 directory.
b. Navigate from the installation directory to the doc/bin directory.
c. Run the help_start script:

```
help_start
```

The systems default Web browser launches to display the stand-alone Information Center.

3. Click the Update button (). On the right hand panel of the Information Center, click Find Updates. A list of updates for existing documentation displays.
4. To initiate the download process, check the selections you want to download, then click Install Updates.
5. After the download and installation process has completed, click Finish.
6. Stop the stand-alone Information Center.
   - On Windows, navigate to the installation directory's doc\bin directory, and run the help_end.bat file:

     ```
     help_end.bat
     ```

     Note: The help_end batch file contains the commands required to safely terminate the processes that were started with the help_start batch file. Do not use `Ctrl-C` or any other method to terminate help_start.bat.
   - On Linux, navigate to the installation directory's doc/bin directory, and run the help_end script:

     ```
     help_end
     ```

     Note: The help_end script contains the commands required to safely terminate the processes that were started with the help_start script. Do not use any other method to terminate the help_start script.
7. Restart the DB2 Information Center.
   - On Windows, click **Start → Control Panel → Administrative Tools → Services**. Then right-click on **DB2 Information Center** service and select **Start**.
   - On Linux, enter the following command:

     ```
     /etc/init.d/db2icdv95 start
     ```

The updated DB2 Information Center displays the new and updated topics.

DB2 tutorials

The DB2 tutorials help you learn about various aspects of DB2 products. Lessons provide step-by-step instructions.

Before you begin

You can view the XHTML version of the tutorial from the Information Center at http://publib.boulder.ibm.com/infocenter/db2help/.

Some lessons use sample data or code. See the tutorial for a description of any prerequisites for its specific tasks.

DB2 tutorials

To view the tutorial, click on the title.

"pureXML™" in *pureXML Guide*
Set up a DB2 database to store XML data and to perform basic operations with the native XML data store.

"Visual Explain" in *Visual Explain Tutorial*
Analyze, optimize, and tune SQL statements for better performance using Visual Explain.

DB2 troubleshooting information

A wide variety of troubleshooting and problem determination information is available to assist you in using DB2 products.

DB2 documentation
Troubleshooting information can be found in the DB2 Troubleshooting Guide or the Support and Troubleshooting section of the DB2 Information Center. There you will find information on how to isolate and identify problems using DB2 diagnostic tools and utilities, solutions to some of the most common problems, and other advice on how to solve problems you might encounter with your DB2 products.

DB2 Technical Support Web site
Refer to the DB2 Technical Support Web site if you are experiencing problems and want help finding possible causes and solutions. The Technical Support site has links to the latest DB2 publications, TechNotes, Authorized Program Analysis Reports (APARs or bug fixes), fix packs, and other resources. You can search through this knowledge base to find possible solutions to your problems.

Access the DB2 Technical Support Web site at http://www.ibm.com/software/data/db2/udb/support.html

Terms and Conditions

Permissions for the use of these publications is granted subject to the following terms and conditions.

Personal use: You may reproduce these Publications for your personal, non commercial use provided that all proprietary notices are preserved. You may not distribute, display or make derivative work of these Publications, or any portion thereof, without the express consent of IBM.

Commercial use: You may reproduce, distribute and display these Publications solely within your enterprise provided that all proprietary notices are preserved. You may not make derivative works of these Publications, or reproduce, distribute or display these Publications or any portion thereof outside your enterprise, without the express consent of IBM.

Except as expressly granted in this permission, no other permissions, licenses or rights are granted, either express or implied, to the Publications or any information, data, software or other intellectual property contained therein.

IBM reserves the right to withdraw the permissions granted herein whenever, in its discretion, the use of the Publications is detrimental to its interest or, as determined by IBM, the above instructions are not being properly followed.

You may not download, export or re-export this information except in full compliance with all applicable laws and regulations, including all United States export laws and regulations.

IBM MAKES NO GUARANTEE ABOUT THE CONTENT OF THESE PUBLICATIONS. THE PUBLICATIONS ARE PROVIDED ″AS-IS″ AND WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, AND FITNESS FOR A PARTICULAR PURPOSE.

Appendix B. Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country/region or send inquiries, in writing, to:

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106, Japan

The following paragraph does not apply to the United Kingdom or any other country/region where such provisions are inconsistent with local law:
INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions; therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

This document may provide links or references to non-IBM Web sites and resources. IBM makes no representations, warranties, or other commitments whatsoever about any non-IBM Web sites or third-party resources that may be referenced, accessible from, or linked from this document. A link to a non-IBM Web site does not mean that IBM endorses the content or use of such Web site or

its owner. In addition, IBM is not a party to or responsible for any transactions you may enter into with third parties, even if you learn of such parties (or use a link to such parties) from an IBM site. Accordingly, you acknowledge and agree that IBM is not responsible for the availability of such external sites or resources, and is not responsible or liable for any content, services, products, or other materials on or available from those sites or resources. Any software provided by third parties is subject to the terms and conditions of the license that accompanies that software.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information that has been exchanged, should contact:

IBM Canada Limited
Office of the Lab Director
8200 Warden Avenue
Markham, Ontario
L6G 1C7
CANADA

Such information may be available, subject to appropriate terms and conditions, including in some cases payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement, or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems, and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements, or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility, or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information may contain examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious, and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information may contain sample application programs, in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM for the purposes of developing, using, marketing, or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.

Each copy or any portion of these sample programs or any derivative work must include a copyright notice as follows:

© (*your company name*) (*year*). Portions of this code are derived from IBM Corp. Sample Programs. © Copyright IBM Corp. *_enter the year or years_*. All rights reserved.

Trademarks

Company, product, or service names identified in the documents of the DB2 Version 9.5 documentation library may be trademarks or service marks of International Business Machines Corporation or other companies. Information on the trademarks of IBM Corporation in the United States, other countries, or both is located at http://www.ibm.com/legal/copytrade.shtml.

The following terms are trademarks or registered trademarks of other companies and have been used in at least one of the documents in the DB2 documentation library:

Microsoft, Windows, Windows NT®, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Intel®, Intel logo, Intel Inside® logo, Intel Centrino®, Intel Centrino logo, Celeron®, Intel Xeon®, Intel SpeedStep®, Itanium® and Pentium® are trademarks of Intel Corporation in the United States, other countries, or both.

Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

Adobe®, the Adobe logo, PostScript®, and the PostScript logo are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, and/or other countries.

Other company, product, or service names may be trademarks or service marks of others.

Index

A

B

C

D

E

F

G

H

I

K

L

M

N

O

P

Q

R

S

T

U

V

W

Printed in USA

SC23-5850-00

Spine information:

DB2 Version 9.5 for Linux, UNIX, and Windows

Database Security Guide