

DB2 10 for z/OS

*Managing Security*





DB2 10 for z/OS

*Managing Security*



**Note**

Before using this information and the product it supports, be sure to read the general information under “Notices” at the end of this information.

**First edition (December 2011)**

This edition applies to DB2 10 for z/OS (product number 5605-DB2), DB2 10 for z/OS Value Unit Edition (product number 5697-P31), and to any subsequent releases until otherwise indicated in new editions. Make sure you are using the correct edition for the level of the product.

Specific changes are indicated by a vertical bar to the left of a change. A vertical bar to the left of a figure caption indicates that the figure has changed. Editorial changes that have no technical significance are not noted.

© Copyright IBM Corporation 1982, 2011.

US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

---

# Contents

<b>About this information</b>	<b>ix</b>
Who should read this information	ix
DB2 Utilities Suite	ix
Terminology and citations	ix
Accessibility features for DB2 10 for z/OS	x
How to send your comments	xi
How to read syntax diagrams	xi
 <b>Chapter 1. Getting started with DB2 security</b>	 <b>1</b>
DB2 security solutions	1
What's new in DB2 10 for z/OS security?	1
DB2 data access control	4
ID-based access control within DB2	4
Role-based access control within DB2	5
Ownership-based access control within DB2	6
Access control through multilevel security	6
Access control external to DB2	6
DB2 subsystem access control	6
Managing access requests from local applications	7
Managing access requests from remote applications	7
Data set protection	8
RACF for data protection	8
Data encryption	8
Scenario: Securing data access at Spiffy Computer	9
Determining security objectives	9
Securing manager access to employee data	9
Securing access to payroll operations and management	13
Managing access privileges of other authorities	16
 <b>Chapter 2. Managing access through authorization IDs and roles</b>	 <b>21</b>
Authorization IDs and roles	22
Authorization IDs	22
Roles in a trusted context	23
Privileges and authorities	23
Explicit privileges	24
Implicit privileges through object ownership	30
Administrative authorities	31
Common DB2 administrative authorities	42
Utility authorities for DB2 catalog and directory	44
Privileges by authorization ID and authority	45
Managing administrative authorities	51
Separating the SYSADM authority	52
Migrating the SYSADM authority	55
Creating roles or trusted contexts with the SECADM authority	57
Altering tables with the system DBADM authority	58
Accessing data with the DATAACCESS authority	59
Granting and revoking privileges with the ACCESSCTRL authority	59
Managing explicit privileges	60
Granting privileges to a role	60
Granting privileges to the PUBLIC ID	61
Granting privileges to remote users	62
Granting privileges through views	62
Granting privileges with the GRANT statement	63
Revoking privileges with the REVOKE statement	68
Managing implicit privileges	77

Managing implicit privileges through object ownership. . . . .	77
Managing implicit privileges through plan or package ownership . . . . .	80
Managing implicit privileges through routines. . . . .	87
Retrieving privilege records in the DB2 catalog . . . . .	101
Catalog tables with privilege records . . . . .	101
Retrieving all authorization IDs or roles with granted privileges . . . . .	102
Retrieving multiple grants of the same privilege. . . . .	103
Retrieving all authorization IDs or roles with the DBADM and system DBADM authorities . . . . .	103
Retrieving all IDs or roles with access to the same table . . . . .	104
Retrieving all IDs or roles with access to the same routine . . . . .	105
Retrieving plans or packages with access to the same table . . . . .	105
Retrieving privilege information through views . . . . .	106
Implementing multilevel security with DB2 . . . . .	107
Multilevel security. . . . .	107
Mandatory access checking . . . . .	111
Implementing multilevel security at the object level . . . . .	113
Implementing multilevel security with row-level granularity . . . . .	115
Restricting access to the security label column . . . . .	117
Managing data in a multilevel-secure environment . . . . .	118
Implementing multilevel security in a distributed environment. . . . .	126
<b>Chapter 3. Managing access through RACF . . . . .</b>	<b>127</b>
Establishing RACF protection for DB2 . . . . .	127
Defining DB2 resources to RACF . . . . .	127
Permitting RACF access . . . . .	129
Managing authorization for stored procedures . . . . .	137
Protecting connection requests that use the TCP/IP protocol . . . . .	146
Establishing Kerberos authentication through RACF . . . . .	147
Implementing DB2 support for enterprise identity mapping. . . . .	148
Configuring the z/OS LDAP server . . . . .	149
Setting up RACF for the z/OS LDAP server . . . . .	150
Setting up the EIM domain controller . . . . .	151
Adding the SAF user mapping plug-in data set to LNKLIST . . . . .	152
Implementing DB2 support for distributed identity filters . . . . .	152
Managing connection requests from local applications. . . . .	153
Processing of connection requests . . . . .	154
Using secondary IDs for connection requests . . . . .	155
Processing of sign-on requests. . . . .	157
Using secondary IDs for sign-on requests . . . . .	158
Using sample connection and sign-on exit routines for CICS transactions . . . . .	159
Managing connection requests from remote applications . . . . .	159
Security mechanisms for DRDA and SNA . . . . .	159
Communications database for the server . . . . .	161
Enabling change of user passwords . . . . .	163
Authorization failure code . . . . .	163
Managing inbound SNA-based connection requests . . . . .	163
Managing inbound TCP/IP-based connection requests . . . . .	170
Managing denial-of-service attacks . . . . .	173
Preventing SQL injection attacks . . . . .	173
Managing outbound connection requests . . . . .	174
Translating outbound IDs . . . . .	183
Sending passwords or password phrases . . . . .	185
<b>Chapter 4. Managing access through row permissions and column masks . . . . .</b>	<b>189</b>
Row and column access control . . . . .	189
Row permission . . . . .	190
Column mask . . . . .	191
Rules of row and column access control . . . . .	192
Creating row permissions . . . . .	196
Creating column masks . . . . .	198

	Modifying column masks to reference UDFs . . . . .	200
	Using INSERT on tables with row access control . . . . .	202
	Creating triggers for tables with row and column access control . . . . .	203
	<b>Chapter 5. Managing access through trusted contexts. . . . .</b>	<b>205</b>
	Trusted contexts . . . . .	205
	Trusted connections . . . . .	206
	Defining trusted contexts . . . . .	206
	Creating local trusted connections . . . . .	207
	Establishing remote trusted connections by DB2 for z/OS requesters . . . . .	208
	Establishing remote trusted connections to DB2 for z/OS servers . . . . .	209
	Switching users of a trusted connection . . . . .	210
	Reusing a local trusted connection through the DSN command processor and DB2I . . . . .	211
	Reusing a remote trusted connection by DB2 for z/OS requesters . . . . .	211
	Reusing a remote trusted connection through DB2 for z/OS servers . . . . .	211
	Reusing a local trusted connection through RRSAP . . . . .	212
	Reusing a local trusted connection through the SQL CONNECT statement . . . . .	212
	Defining external security profiles . . . . .	213
	Enabling users to perform actions on behalf of others . . . . .	213
	Performing tasks on objects for other users . . . . .	214
	<b>Chapter 6. Managing access through data definition control . . . . .</b>	<b>215</b>
	Data definition statements . . . . .	215
	Data definition control support . . . . .	215
	Registration tables . . . . .	216
	Installing data definition control support . . . . .	218
	Enabling data definition control . . . . .	218
	Controlling data definition by application name . . . . .	219
	Controlling data definition by application name with exceptions . . . . .	220
	Controlling data definition by object name . . . . .	221
	Controlling data definition by object name with exceptions . . . . .	222
	Registering object sets . . . . .	224
	Disabling data definition control . . . . .	225
	Managing registration tables and indexes . . . . .	225
	Creating registration tables and indexes . . . . .	225
	Naming registration tables and indexes . . . . .	226
	Dropping registration tables and indexes . . . . .	226
	Creating table spaces for registration tables . . . . .	227
	Adding columns to registration tables . . . . .	227
	Updating registration tables . . . . .	227
	<b>Chapter 7. Managing access through exit routines . . . . .</b>	<b>229</b>
	Connection routines and sign-on routines . . . . .	229
	Specifying connection and sign-on routines . . . . .	229
	Sample connection and sign-on routines . . . . .	230
	When connection and sign-on routines are taken . . . . .	231
	Exit parameter list for connection and sign-on routines . . . . .	231
	Authorization ID parameter list for connection and sign-on routines . . . . .	234
	Input values for connection routines . . . . .	235
	Input values for sign-on routines . . . . .	235
	Expected output for connection and sign-on routines . . . . .	236
	Processing in sample connection and sign-on routines . . . . .	236
	Performance considerations for connection and sign-on routines . . . . .	238
	Debugging connection and sign-on routines . . . . .	238
	Session variables in connection and sign-on routines . . . . .	240
	Access control authorization exit routine . . . . .	241
	Specifying the access control authorization routine . . . . .	242
	The default access control authorization routine . . . . .	243
	When access control authorization routine is taken . . . . .	243
	Considerations for the access control authorization routine . . . . .	243

Parameter list for access control authorization routines . . . . .	249
Expected output for access control authorization routines . . . . .	259
Debugging access control authorization routines. . . . .	262
Determining whether the access control authorization routine is active . . . . .	262
RACF access control module . . . . .	262

## **Chapter 8. Protecting data through encryption and RACF . . . . . 263**

Encrypting your data with Secure Socket Layer support . . . . .	263
AT-TLS configuration. . . . .	263
SSL authentication level . . . . .	264
Configuring the DB2 server for SSL . . . . .	267
Configuring the DB2 requester for SSL . . . . .	268
Protecting data sets through RACF . . . . .	269
Adding groups to control DB2 data sets . . . . .	269
Creating generic profiles for data sets . . . . .	270
Authorizing DB2 IDs to use data set profiles . . . . .	271
Enabling DB2 IDs to create data sets . . . . .	272
Encrypting your data through DB2 built-in functions . . . . .	272
Defining columns for encrypted data . . . . .	272
Defining column-level encryption . . . . .	273
Defining value-level encryption . . . . .	275
Using predicates for encrypted data . . . . .	277
Optimizing performance of encrypted data . . . . .	277

## **Chapter 9. Auditing access to DB2 . . . . . 281**

Determining active security measures . . . . .	281
DB2 audit trace. . . . .	282
Authorization IDs traced by auditing . . . . .	282
Audit classes . . . . .	283
Audit trace reports . . . . .	284
Audit trace records . . . . .	285
Limitations of the audit trace . . . . .	285
Starting the audit trace . . . . .	286
Stopping the audit trace. . . . .	286
Collecting audit trace records . . . . .	287
Formatting audit trace records. . . . .	287
Auditing in a distributed data environment . . . . .	288
DB2 audit policy . . . . .	288
Audit category . . . . .	288
Creating and activating audit policies . . . . .	291
Auditing the use of an administrative authority . . . . .	292
Auditing tables without specifying the AUDIT clause . . . . .	293
Additional sources of audit information . . . . .	294
Determining ID privileges and authorities. . . . .	294
Auditing specific IDs or roles . . . . .	295
Auditing specific tables . . . . .	295
Ensuring data accuracy and integrity . . . . .	296
Ensuring data presence and uniqueness . . . . .	296
Protecting data integrity. . . . .	297
Tracking data changes . . . . .	298
Checking for lost and incomplete transactions . . . . .	298
Ensuring data consistency . . . . .	298
Using referential integrity for data consistency . . . . .	299
Using locks for data consistency . . . . .	299
Checking data consistency . . . . .	300

## **Information resources for DB2 for z/OS and related products . . . . . 303**

## **How to obtain DB2 information. . . . . 309**



<b>How to use the DB2 library</b>	<b>311</b>
<b>Notices</b>	<b>315</b>
Programming Interface Information	317
Trademarks	317
<b>Glossary</b>	<b>319</b>
<b>Index</b>	<b>365</b>



---

## About this information

This information provides guidance that you can use to manage security in a DB2® for z/OS® environment.

This information assumes that your DB2 subsystem is running in Version 10 new-function mode. Generally, new functions that are described, including changes to existing functions, statements, and limits, are available only in new-function mode, unless explicitly stated otherwise. Exceptions to this general statement include optimization and virtual storage enhancements, which are also available in conversion mode unless stated otherwise. In Versions 8 and 9, most utility functions were available in conversion mode. However, for Version 10, most utility functions work only in new-function mode.

---

## Who should read this information

This information is primarily intended for security, system, and database administrators. It assumes that the user is familiar with the basic concepts and facilities of DB2 for z/OS (DB2), z/OS, RACF®, and Structured Query Language (SQL).

---

## DB2 Utilities Suite

**Important:** In this version of DB2 for z/OS, the DB2 Utilities Suite is available as an optional product. You must separately order and purchase a license to such utilities, and discussion of those utility functions in this publication is not intended to otherwise imply that you have a license to them.

The DB2 Utilities Suite can work with DB2 Sort and the DFSORT program, which you are licensed to use in support of the DB2 utilities even if you do not otherwise license DFSORT for general use. If your primary sort product is not DFSORT, consider the following informational APARs mandatory reading:

- II14047/II14213: USE OF DFSORT BY DB2 UTILITIES
- II13495: HOW DFSORT TAKES ADVANTAGE OF 64-BIT REAL ARCHITECTURE

These informational APARs are periodically updated.

### **Related information**

DB2 utilities packaging (Utility Guide)

---

## Terminology and citations

When referring to a DB2 product other than DB2 for z/OS, this information uses the product's full name to avoid ambiguity.

The following terms are used as indicated:

**DB2** Represents either the DB2 licensed program or a particular DB2 subsystem.

**OMEGAMON®**

Refers to any of the following products:

- IBM® Tivoli® OMEGAMON XE for DB2 Performance Expert on z/OS

- IBM Tivoli OMEGAMON XE for DB2 Performance Monitor on z/OS
- IBM DB2 Performance Expert for Multiplatforms and Workgroups
- IBM DB2 Buffer Pool Analyzer for z/OS

**C, C++, and C language**

Represent the C or C++ programming language.

**CICS®** Represents CICS Transaction Server for z/OS.

**IMS™** Represents the IMS Database Manager or IMS Transaction Manager.

**MVS™** Represents the MVS element of the z/OS operating system, which is equivalent to the Base Control Program (BCP) component of the z/OS operating system.

**RACF** Represents the functions that are provided by the RACF component of the z/OS Security Server.

---

## Accessibility features for DB2 10 for z/OS

Accessibility features help a user who has a physical disability, such as restricted mobility or limited vision, to use information technology products successfully.

### Accessibility features

The following list includes the major accessibility features in z/OS products, including DB2 10 for z/OS. These features support:

- Keyboard-only operation.
- Interfaces that are commonly used by screen readers and screen magnifiers.
- Customization of display attributes such as color, contrast, and font size

**Tip:** The Information Management Software for z/OS Solutions Information Center (which includes information for DB2 10 for z/OS) and its related publications are accessibility-enabled for the IBM Home Page Reader. You can operate all features using the keyboard instead of the mouse.

### Keyboard navigation

You can access DB2 10 for z/OS ISPF panel functions by using a keyboard or keyboard shortcut keys.

For information about navigating the DB2 10 for z/OS ISPF panels using TSO/E or ISPF, refer to the *z/OS TSO/E Primer*, the *z/OS TSO/E User's Guide*, and the *z/OS ISPF User's Guide*. These guides describe how to navigate each interface, including the use of keyboard shortcuts or function keys (PF keys). Each guide includes the default settings for the PF keys and explains how to modify their functions.

### Related accessibility information

Online documentation for DB2 10 for z/OS is available in the Information Management Software for z/OS Solutions Information Center, which is available at the following website: <http://publib.boulder.ibm.com/infocenter/imzic>

### IBM and accessibility

See the *IBM Accessibility Center* at <http://www.ibm.com/able> for more information about the commitment that IBM has to accessibility.

---

## How to send your comments

Your feedback helps IBM to provide quality information. Please send any comments that you have about this book or other DB2 for z/OS documentation. You can use the following methods to provide comments:

- Send your comments by email to [db2zinfo@us.ibm.com](mailto:db2zinfo@us.ibm.com) and include the name of the product, the version number of the product, and the number of the book. If you are commenting on specific text, please list the location of the text (for example, a chapter and section title or a help topic title).
- You can send comments from the web. Visit the DB2 for z/OS - Technical Resources website at:

<http://www.ibm.com/support/docview.wss?rs=64&uid=swg27011656>

This website has an online reader comment form that you can use to send comments.





- You can also send comments by using the **Feedback** link at the footer of each page in the Information Management Software for z/OS Solutions Information Center at <http://publib.boulder.ibm.com/infocenter/imzic>.

---

## How to read syntax diagrams

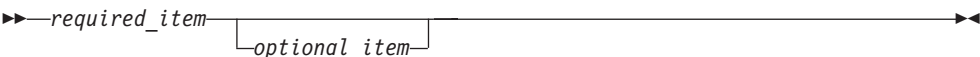
Certain conventions apply to the syntax diagrams that are used in IBM documentation.

Apply the following rules when reading the syntax diagrams that are used in DB2 for z/OS documentation:

- Read the syntax diagrams from left to right, from top to bottom, following the path of the line.
  - The  symbol indicates the beginning of a statement.
  - The  symbol indicates that the statement syntax is continued on the next line.
  - The  symbol indicates that a statement is continued from the previous line.
  - The  symbol indicates the end of a statement.
- Required items appear on the horizontal line (the main path).



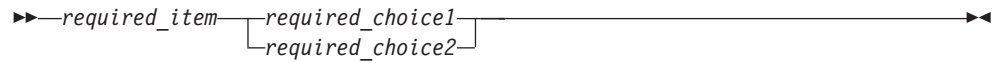
- Optional items appear below the main path.



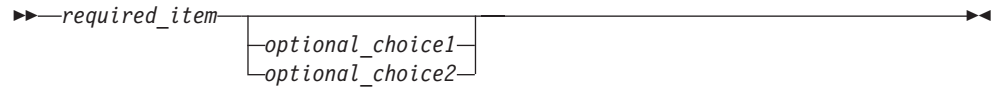
If an optional item appears above the main path, that item has no effect on the execution of the statement and is used only for readability.



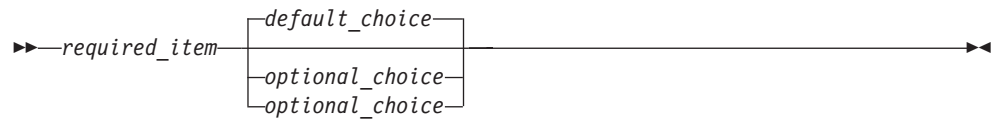
- If you can choose from two or more items, they appear vertically, in a stack. If you *must* choose one of the items, one item of the stack appears on the main path.



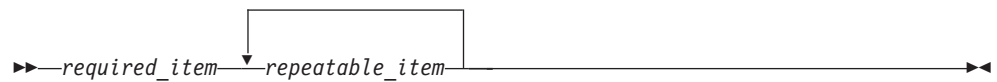
If choosing one of the items is optional, the entire stack appears below the main path.



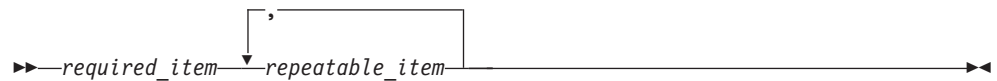
If one of the items is the default, it appears above the main path and the remaining choices are shown below.



- An arrow returning to the left, above the main line, indicates an item that can be repeated.



If the repeat arrow contains a comma, you must separate repeated items with a comma.



A repeat arrow above a stack indicates that you can repeat the items in the stack.

- Sometimes a diagram must be split into fragments. The syntax fragment is shown separately from the main syntax diagram, but the contents of the fragment should be read as if they are on the main path of the diagram.



#### fragment-name:



- With the exception of XPath keywords, keywords appear in uppercase (for example, FROM). Keywords must be spelled exactly as shown. XPath keywords are defined as lowercase names, and must be spelled exactly as shown. Variables appear in all lowercase letters (for example, *column-name*). They represent user-supplied names or values.
- If punctuation marks, parentheses, arithmetic operators, or other such symbols are shown, you must enter them as part of the syntax.

---

## Chapter 1. Getting started with DB2 security

DB2 *security* refers to the protection of sensitive data and system resources by controlling access to DB2 objects, subsystems, and other assets.

DB2 security is set through a security plan, implemented through privilege and authority management, and reinforced through the audit of accesses to protected data. A *security plan* defines the security objectives of your organization and specifies the policies and techniques that you use to meet these objectives. A *security audit* traces all data access and determines whether your security plan works as designed and implemented.

If you are new to DB2 security, skim through the succeeding topics for a brief overview of the techniques that you can use to manage access to your DB2 and protect your data before reading the scenario.

---

### DB2 security solutions

With each new release, DB2 gets faster and more secure.

Over the years, DB2 recognizes and addresses the following security problems:

- Privilege theft or mismanagement
- Application or application server tampering
- Data or log tampering
- Storage media theft
- Unauthorized access to objects

DB2 offers the following security solutions to address the problems:

- Authentication
- Authorization
- Data integrity
- Confidentiality
- System integrity
- Audit

---

### What's new in DB2 10 for z/OS security?

DB2 10 for z/OS provides critical enhancements to security and auditing. These enhancements strengthen DB2 security in the z/OS environment.

#### Increased granularity in DB2 administrative authorities

Any user (authorization ID or role) with the SYSADM authority can access data from any table in an entire DB2 subsystem. Granting the SYSADM authority to too many users may introduce security risks. You can minimize these risks by granting the SYSADM authority to as few users as possible while taking advantage of the following new DB2 administrative authorities. These authorities help improve the granularity in DB2 administrative authorities and meet the increasingly demanding security needs of your business.

## **SECADM**

The SECADM authority can manage security-related objects in DB2 and control access to all database resources. It, however, doesn't have any inherent privilege to access any user data in those databases.

## **ACCESSCTRL**

The ACCESSCTRL authority allows you to grant explicit privileges to authorization IDs or roles by issuing SQL GRANT statements and revoke privileges by issuing SQL REVOKE statements with the BY clause. It does not allow you to grant or revoke the CREATE\_SECURE\_OBJECT privilege or the system DBADM, DATAACCESS, and ACCESSCTRL authorities.

## **DATAACCESS**

The DATAACCESS authority allows you to access data in tables, views, and materialized query tables in a DB2 subsystem. It also allows you to execute plans, packages, functions, and procedures.

## **System DBADM**

The system DBADM authority allows an administrator to manage all databases in a DB2 subsystem. By default, the system DBADM has all the privileges of the DATAACCESS and ACCESSCTRL authorities. If you do not want a user (an authorization ID or role) with the system DBADM authority to grant any explicit privileges, you can specify the WITHOUT ACCESSCTRL clause in the GRANT statement when you grant the authority. If you do not want a user with the system DBADM authority to access any user data in the databases, you can specify the WITHOUT DATAACCESS clause in the GRANT statement when you grant the authority. If necessary, you can still grant explicit privileges (i.e., SELECT) to the system DBADM user to access data or perform grants.

## **SQLADM**

The SQLADM authority allows you to issue the SQL EXPLAIN statements, execute the PROFILE commands, run the RUNSTATS and MODIFY STATISTICS utilities on all user databases, and execute system-defined routines, such as stored procedures or functions, and any packages that are executed within the routines.

## **Separation of DB2 administrative authorities**

The separation of DB2 administrative authorities allows you to comply with the increasingly complex governance or compliance regulations of your business. It also helps reduce the potential security risks that are caused by the general use of the SYSADM authority. Separating the administrative tasks into multiple auditable authorities prevents a single user with the SYSADM authority from having control over several phases of a task and from committing any security fraud. Both DB2 for z/OS and DB2 for Linux, UNIX, and Windows provides the same separation of DB2 administrative authorities.

Depending on how you set the SEPARATE\_SECURITY system parameter on panel DSNTIPP1 during installation or migration, you can separate DB2 security administration from system administration.

If you set SEPARATE\_SECURITY to YES, the SYSADM authority can no longer control access or manage security-related objects (i.e., roles, trusted contexts, row permissions, and column masks). It cannot grant or revoke privileges that are granted by others, either. In addition, the SYSCTRL authority can no longer manage roles or grant or revoke privileges that are granted by others. Instead, the



new SECADM authority will manage all security-related objects and control access to all databases even though it cannot access any data stored in the databases.

If you set `SEPARATE_SECURITY` to `NO` (which is the default), the `SYSADM` authority retains all the existing privileges and responsibilities and gets implicit privileges of the `SECADM` authority. In other words, the `SYSADM` authority continues to be the security administrator and manage all security-related objects, perform grants, and revoke privileges that are granted by others. In addition, it gets implicit insert, update, delete access on the `SYSIBM.SYSAUDITPOLICIES` table and is able to set `CURRENT SQLID` and `BIND OWNER` to any value.

Setting `SEPARATE_SECURITY` to `NO` also allows the `SYSCTRL` authority to get most of the implicit privileges of the `ACCESSCTRL` authority. `SYSCTRL` can manage roles, perform certain grants, and revoke privileges that are granted by others, and set `BIND OWNER` to any value.

## **EXPLAIN system privilege**

The `EXPLAIN` system privilege provides you the ability to issue `EXPLAIN`, `PREPARE`, and `DESCRIBE` statements without requiring the privilege to execute the statements. It allows you to validate your applications and SQL syntaxes, without requiring access to data, before putting them into production.

## **Row and column access control**

Row and column access control is an SQL security solution that enables you to manage access to a table at the level of a row, a column, or both. It allows administrators to establish security policies within DB2 and enforce security controls on all applications and tools (except utilities) that access data in DB2. It enables you to separate applications from security implementation and comply with new security regulations without having to change the existing applications. In addition, row and column access control helps prevent the use of SQL to bypass views or the SQL injection attacks on a DB2 subsystem.

You can implement row access control through row permissions and column access control through column masks.

### **Row permission**

A row permission is a database object that describes a specific row access control rule for a table. In the form of an SQL search condition, the rule specifies the conditions under which a user, group, or role can access the rows of data in the table.

### **Column mask**

A column mask is a database object that describes a specific column access control rule for a column. In the form of an SQL `CASE` expression, the rule specifies the conditions under which a user, group, or role can receive the masked values returned for a column.

## **Audit policy**

An audit policy is a set of criteria that determines the categories to be audited. It helps you configure and control the audit requirements of your security policies and to monitor data access by applications and individual users (authorization IDs or roles), including administrative authorities. You can create different audit policies to address the different security needs of your business. For example, you

can create and activate an audit policy to audit how a DB2 administrative authority is used.

## DB2 data access control

Access to data can originate from users through interactive terminal sessions, local or remote stored procedures, utilities, or IMS or CICS transactions. It can also originate from application programs that run in batch mode, remote applications that use DDF or CLI and JDBC drivers, or web-based applications supported by WebSphere® Application Servers.

Given the variety of access originators, the term *process* is used to represent all access to data. For example, within a DB2 subsystem, a process can be a primary authorization ID, one or more secondary IDs, a role, or an SQL ID.

A process can gain access to DB2 data through several routines. As shown in the following diagram, DB2 provides different ways for you to control access from all but the data set protection route.

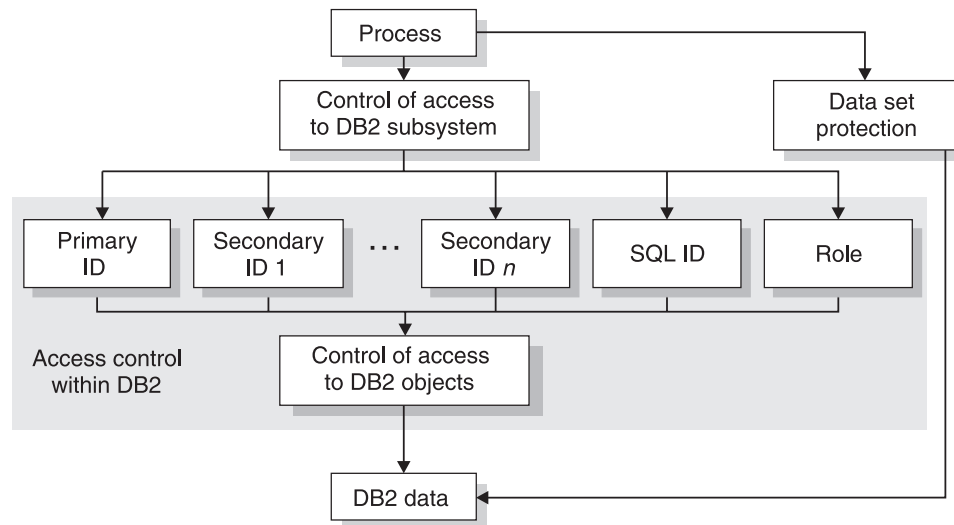


Figure 1. DB2 data access control

One of the ways that DB2 controls access to data is by using authorization IDs or roles. DB2 relies on IDs or roles to determine whether to allow or prohibit certain processes. DB2 assigns privileges and authorities to IDs or roles so that the owning users can take actions on objects. In this sense, it is an ID or a role, not a user, that owns an object. In other words, DB2 does not base access control on a specific user or person who need access. For example, if you allow other users to use your IDs, DB2 recognizes only the IDs, not the people or programs that use them.

### Related concepts

“DB2 subsystem access control” on page 6

## ID-based access control within DB2

DB2 provides a wide range of granularity when you grant privileges to an ID within DB2. You can grant privileges and authorities to groups, secondary IDs, or to roles.

For example, you could, separately and specifically, grant to an ID the privilege to retrieve data from the table, to insert rows, to delete rows, or to update specific columns. By granting or not granting privileges on views of the table, you can specify exactly what an ID can do to the table, down to the granularity of specific fields. You can also grant to an ID specific privileges on databases, plans, packages, and the entire DB2 subsystem. If you grant or revoke privileges on a procedure or procedure package, all versions of that procedure or procedure package have those privileges.

DB2 also defines sets of related privileges, called *administrative authorities*. When you grant one of the administrative authorities to a person's ID, that person has all of the privileges that are associated with that administrative authority. You can efficiently grant many privileges by granting one administrative authority.

You can also efficiently grant multiple privileges by granting the privilege to execute an application plan or a package. When an ID executes a plan or package, the ID implicitly uses all of the privileges that the owner needed when binding the plan or package. Therefore, granting to an ID the privilege to execute a plan or package can provide a finely detailed set of privileges and can eliminate the need to grant other privileges separately.

**Example:** Assume that an application plan issues the INSERT and SELECT statements on several tables. You need to grant INSERT and SELECT privileges only to the plan owner. However, any authorization ID that is later granted the EXECUTE privilege on the plan can perform those same INSERT and SELECT statements by executing the plan. You do not need to explicitly grant the INSERT and SELECT privileges to the ID.

**Recommendation:** Instead of granting privileges to many primary authorization IDs, consider associating each of those primary IDs with the same secondary ID or a role if running in a trusted context. Then grant the privileges to the secondary ID or role. You can associate a primary ID with one or more secondary IDs or roles when the primary ID gains access to the DB2 subsystem. DB2 makes the association within an exit routine. The assignment of privileges to the secondary ID or role is controlled entirely within DB2.

#### **Related concepts**

“Role-based access control within DB2”

“Ownership-based access control within DB2” on page 6

## **Role-based access control within DB2**

A *privilege* enables the user of an ID to execute certain SQL statements or to access the objects of another user. A *role* groups the privileges together so that they can be simultaneously granted to and revoked from multiple users.

A role is a database object that is created in DB2. It is defined through the SQL CREATE ROLE statement and a trusted connection. A role cannot be used outside of a trusted context unless the user in a role grants privileges to an ID.

#### **Related concepts**

“ID-based access control within DB2” on page 4

“Ownership-based access control within DB2”

## **Ownership-based access control within DB2**

Object ownership carries with it a set of related privileges on the object. DB2 provides separate controls for creation and ownership of objects.

If you want to prevent users from obtaining implicit privileges from object ownership, you can make a DB2 role the owner of the object. To do this, you need to create the object in a trusted context that is defined with the `ROLE AS OBJECT OWNER AND QUALIFIER` clause.

#### **Related concepts**

“ID-based access control within DB2” on page 4

“Role-based access control within DB2” on page 5

#### **Related tasks**

“Changing object ownership” on page 79

## **Access control through multilevel security**

*Multilevel security*, also known as label-based access control, allows you to classify objects and users with security labels. The security labels are based on hierarchical security levels and non-hierarchical security categories.

DB2 multilevel security solution utilizes the multilevel security feature in the z/OS operating system. It prevents unauthorized users from accessing information at a higher classification than their authorization. It also prevents users from declassifying information.

Using multilevel security with row-level granularity, you can define strong security for DB2 objects and perform security checks, including row-level security checks. Row-level security checks allow you to control which users have authorization to view, modify, or perform other actions on specific rows of data.

#### **Related reference**

“Implementing multilevel security with DB2” on page 107

## **Access control external to DB2**

You can control access to DB2 by using a DB2-supplied exit routine or an exit routine that you write.

If your installation uses one of the access control authorization exit routines, you can use it to control authorization and authentication checking, instead of using other techniques and methods.

#### **Related concepts**

“Access control authorization exit routine” on page 241

---

## **DB2 subsystem access control**

You can control whether a process can gain access to a specific DB2 subsystem from outside of DB2. A common approach is to grant access through RACF or a similar security system.

A RACF system provides several advantages. For example, you can use RACF for the following objectives:

- Identify and verify the identifier that is associated with a process
- Connect those identifiers to RACF group names
- Log and report unauthorized attempts to access protected resources

Profiles for access to DB2 from various environments and DB2 address spaces are defined as resources to RACF. Each request to access DB2 is associated with an ID. RACF determines whether the ID is authorized for DB2 resources. If the ID is authorized, RACF permits access to DB2.

You can also consider using the security capabilities of IMS or CICS to manage access to DB2:

- *IMS terminal security* lets you limit the entry of a transaction code to a particular logical terminal (LTERM) or group of LTERMs in the system. To protect a particular program, you can authorize a transaction code that is to be entered only from any terminal on a list of LTERMs. Alternatively, you can associate each LTERM with a list of the transaction codes that a user can enter from that LTERM. IMS then passes the validated LTERM name to DB2 as the initial primary authorization ID
- *CICS transaction code security* works with RACF to control the transactions and programs that can access DB2. Within DB2, you can use the ENABLE and DISABLE options of the bind operation to limit access to specific CICS subsystems.

#### **Related concepts**

“DB2 data access control” on page 4

## **Managing access requests from local applications**

If you request access to a local DB2 subsystem, your request is often subject to several checks before you are granted access.

If you run DB2 under TSO and use the TSO logon ID as the DB2 primary authorization ID, TSO verifies your ID when you log on. When you gain access to DB2, you can use a self-written or IBM-supplied DSN3@ATH exit routine that is connected to DB2 to perform the following actions:

- Check the authorization ID again
- Change the authorization ID
- Associate the authorization ID with secondary IDs

After these actions are performed, the authorization ID can use the services of an external security system again.

## **Managing access requests from remote applications**

You can require remote users to pass several access checks before they reach DB2. You can use RACF or a similar security subsystem to control access from a remote location.

While controlling access from a remote locations, RACF can do the following:

- Verify an ID that is associated with a remote attachment request and check the ID with a password
- Generate *PassTickets* on the sending side. PassTickets can be used instead of passwords. A PassTicket lets a user gain access to a host system without sending the RACF password across the network.

- Verify a Kerberos ticket if your distributed environment uses Kerberos to manage user access and perform user authentication

You can also control access authentication by using the *DB2 communications database* (CDB). The CDB is a set of tables in the DB2 catalog that are used to establish conversations with remote database management systems. The CDB can translate IDs before it sends them to the remote system.

You can use the RACF DSNR general resource class for DB2 for access authentication. With RACF DSNR, you can control access to the DB2 server by the IDs that are defined to the ssnm.DIST profile with READ. In addition, you can use the port of entry (POE) checking by RACF and the z/OS communications server to protect against unauthorized remote connections to DB2.

---

## Data set protection

The data in a DB2 subsystem is contained in data sets. The data sets can be accessed without going through DB2. To protect your data, you need to control all access routes by using different access control methods and mechanisms. For example, you can determine who can use offline utilities by assigning appropriate access.

### RACF for data protection

If you use RACF, or an equivalent security system, to control access to DB2, consider controlling access to your data sets.

If you want to use RACF for data set protection outside of the DB2 subsystem, define RACF profiles for data sets and permit access to the data sets for certain DB2 IDs.

### Data encryption

If your data is very sensitive, consider encrypting the data. Encryption protects against unauthorized access to data sets and to backup copies outside of the DB2 subsystem.

You have the following encryption options for protecting sensitive data:

- IBM System Storage® DS8000® support for data encryption with the IBM Full Disk Encryption drives
- IBM System Storage TS1130 encryption solution
- Secure Socket Layer (SSL) protocol through the z/OS Communications Server IP Application Transparent Transport Layer (AT-TLS) service
- IBM Encryption Facility for z/OS
- Advanced Encryption Standard (AES) for encrypting userids and passwords over network connections
- DB2 edit procedures or field procedures, which can use the Integrated Cryptographic Service Facility (ICSF)
- IBM Data Encryption for IMS and DB2 Databases tool
- Encryption tools and facilities that used outside of DB2

You can consider compressing your data sets before encrypting the data. Data compression is not a substitute for encryption. In some cases, the compression method does not actually shorten the data. In those cases, the data is left uncompressed and readable. If you encrypt and compress your data, compress it

first. After you obtain the maximum compression, encrypt the result. When you retrieve your data, first decrypt the data. After the data is decrypted, decompress the result.

---

## Scenario: Securing data access at Spiffy Computer

This scenario describes a simple approach to securing local and remote access to the sensitive data of employees, payroll operations, and payroll management at Spiffy Computer Company. It shows how to enforce a security plan by using authorization IDs, roles, privileges, authorities, and the audit trace.

You should base your security plan, techniques, and procedures on your actual security objectives; do not view this sample security plan as an exact model for your security needs. Instead, use it to understand various possibilities and address problem areas that you might encounter when you implement your security plan.

### Determining security objectives

An important step in defining and implementing an effective security plan is to determine your security objectives.

Suppose that the Spiffy Computer Company management team determines the following security objectives:

- Managers can see, but not update, all of the employee data for members of their own departments.
- Managers of managers can see all of the data for employees of departments that report to them.
- The employee table resides at a central location. Managers at remote locations can query the data in the table.
- The payroll operations department makes changes to the employee table. Members of the payroll operations department can update any column of the employee table except for the salary, bonus, and commission columns.
- Members of payroll operations can update any row except for rows that are for members of their own department. Because changes to the table are made only from a central location, distributed access does not affect payroll operations.
- Changes to the salary, bonus, and commission columns are made through a process that involves the payroll update table. When an employee's compensation changes, a member of the payroll operations department can insert rows in the payroll update table. For example, a member of the payroll operations department might insert a row in the compensation table that lists an employee ID and an updated salary. Next, the payroll management group can verify inserted rows and transfer the changes to the employee table.
- No one else can see the employee data. The security plan cannot fully achieve this objective because some ID must occasionally exercise SYSADM authority. While exercising SYSADM authority, an ID can retrieve any data in the system. The security plan uses the trace facility to monitor the use of that power.

### Securing manager access to employee data

As a security measurement, the Spiffy Computer Company sets clear restrictions on how its managers can access employee data.

Specifically, it imposes the following security restrictions on managers:

- Managers can retrieve, but not change, all information in the employee table for members of their own departments.



- Managers of managers have the same privileges for their own departments and for the departments that directly report to them.

## Creating views of employee data

The Spiffy security planners decide to use views for implementing the restrictions on managers' access to employee data.

To create a view of employee data for every employee that reports to a manager, the Spiffy security planners perform the following steps:

1. Add a column that contains manager IDs to DSN8910.DEPT, as shown in the following statement:

```
ALTER TABLE DSN81010.DEPT
  ADD MGRID CHAR(8) FOR SBCS DATA NOT NULL WITH DEFAULT;
```

2. Create a view that selects employee information about employees that work for a given manager, as shown in the following statement:

```
CREATE VIEW DEPTMGR AS
  SELECT * FROM DSN81010.EMP, DSN81010.DEPT
    WHERE WORKDEPT = DEPTNO
      AND MGRID = USER;
```

3. Ensure that every manager has the SELECT privilege on the view.

## Granting managers the SELECT privilege

The security planners for Spiffy Computer Company can take an "individual" approach or a "functional" approach when they grant the SELECT privilege on a view to managers.

With an individual approach, they can grant privileges to individual IDs and revoke them if the user of the ID leaves the company or transfers to another position. With a functional approach, they can create RACF groups, and grant privileges to the group IDs, with the intention of never revoking them. When an individual ID needs those privileges, connect that ID to the group; disconnect the ID when its user leaves or transfers.

The Spiffy security planners know that the functional approach is usually more convenient in the following situations:

- Each function, such as the manager function, requires many different privileges. When functional privileges are revoked from one user, they must be granted to another user.
- Several users need the same set of privileges.
- The privileges are given with the grant option, or the privileges let users create objects that must persist after their original owners leave or transfer. In both cases, revoking the privileges might not be appropriate. The revokes cascade to other users. To change ownership, you might need to drop objects and re-create them.

Some of the Spiffy requirements for securing manager access suggest the functional approach. However, in this case, the function needs only one privilege. The privilege does not carry the grant option, and the privilege does not allow new objects to be created.

Therefore, the Spiffy security planners choose the individual approach, and plan to re-examine their decision later. Spiffy security planners grant all managers the SELECT privilege on the views for their departments.



**Example:** To grant the SELECT privilege on the DEPTMGR view to the manager with ID EMP0060, the planners use the following GRANT statement:

```
GRANT SELECT ON DEPTMGR TO EMP0060;
```

## Managing distributed access

Some Spiffy managers must use views to query data in the central employee table from remote locations. The security plan must ensure that this type of distributed access is secure. Therefore, security administrators must implement a sound plan for distributed access.

### Planning for distributed access:

The Spiffy security planners need to determine how the managers can securely access employee data in a distributed environment.

To secure distributed access to employee data, the Spiffy security planners must address the following questions:

- Which IDs should hold privileges on which views?
- How do the central location and the remote locations divide security responsibilities for IDs?

The Spiffy security planners answer those questions with the following decisions:

- IDs that are managed at the central location hold privileges on views for departments that are at remote locations. For example, the ID MGRD11 has the SELECT privilege on the view DEPTD11.
- If the manager of Department D11 uses a remote system, the ID at that system must be translated to MGRD11. Then a request is sent to the central system. All other IDs are translated to CLERK before they are sent to the central system.
- The communications database (CDB) manages the translated IDs, like MGRD11.
- An ID from a remote system must be authenticated on any request to the central system.

### Implementing distributed access at the central server:

To enable distributed access to sensitive employee data, the Spiffy security plan requires certain security measures to be implemented at the central server location.

The following actions must occur at the central server location:

- The central DB2 subsystem must authenticate every incoming ID with RACF.
- For SNA connections, the Spiffy security planners must include an entry in table SYSIBM.LUNAMES in the CDB; the entry in the LUNAME column identifies the LU name of every remote location. The entry must specify that connections must be verified.

**Example:** The following table shows an entry in SYSIBM.LUNAMES for LUREMOTE.

*Table 1. The SYSIBM.LUNAMES table at the central location*

LUNAME	USERNAMES	SECURITY_IN	ENCRYPTPSWDS
LUREMOTE	blank	V	N

The value of V for SECURITY\_IN indicates that incoming remote connections must include verification. The value of N for ENCRYPTPSWDS indicates that passwords are not in internal RACF encrypted format.

The security plan treats all remote locations alike, so it does not require encrypted passwords. The option to require encrypted passwords is available only between two DB2 subsystems that use SNA connections.

- For TCP/IP connections, the Spiffy security planners must set the TCP/IP ALREADY VERIFIED field of installation panel DSN TIP5 to NO. This setting ensures that the incoming requests that use TCP/IP are not accepted without authentication.
- The Spiffy security planners must grant all privileges and authorities that are required by the manager of Department D11 to the ID, MGRD11. The security planners must grant similar privileges to IDs that correspond to the remaining managers.

### Implementing distributed access at remote locations:

To enable distributed access to sensitive employee data, the Spiffy security plan requires certain security measures to be implemented at the remote locations.

The following actions must occur at the remote locations to enable distributed access for the Spiffy security plan:

- For SNA connections, the Spiffy security planners must include an entry in table SYSIBM.LUNAMES for the LU name of the central location. The entry must specify an outbound ID translation for attachment requests to that location.

**Example:** The following table shows an entry in SYSIBM.LUNAMES for LUCENTRAL.

Table 2. The SYSIBM.LUNAMES table at the remote location

LUNAME	USERNAMES	SECURITY_OUT
LUCENTRAL	O	P

The value of O for USERNAMES indicates that translation checking is performed on outbound IDs, but not on inbound IDs. The value of P for SECURITY\_OUT indicates that outbound connection requests contain a user password and a RACF PassTicket.

- For TCP/IP connections, the Spiffy security planners must include an entry in table SYSIBM.IPNAMES for the LU name that is used by the central location. The content of the LUNAME column is used to generate RACF PassTickets. The entry must specify outbound ID translation for requests to that location.

**Example:** The following table shows an entry in SYSIBM.IPNAMES for LUCENTRAL.

Table 3. The SYSIBM.IPNAMES table at the remote location

LINKNAME	USERNAMES	SECURITY_OUT	IPADDR
LUCENTRAL		R	central.vnet.ibm.com

- The Spiffy security planners must include entries in table SYSIBM.USERNAMES to translate outbound IDs.

**Example:** The following table shows two entries in SYSIBM.USERNAMES.

Table 4. The SYSIBM.USERNAMES table at the remote location

TYPE	AUTHID	LINKNAME	NEWAUTHID
O	MEL1234	LUCENTRAL	MGRD11
O	blank	LUCENTRAL	CLERK

MEL1234 is translated to MGRD11 before it is sent to the LU that is specified in

the LINKNAME column. All other IDs are translated to CLERK before they are sent to that LU.

**Exception:** For a product other than DB2 for z/OS, the actions at the remote location might be different. If you use a different product, check the documentation for that product. The remote product must satisfy the requirements that are imposed by the central subsystem.

### **Auditing manager access**

The Spiffy payroll data is extremely sensitive. The security plan requires the audit trace to be automatically started for all classes whenever DB2 is started.

To ensure that an audit record exists for every access to the employee table, the Spiffy security planners create an audit policy for the employee table. Every week, the security planners scan the records and determine the number of accesses by each manager.

The report highlights any number of accesses outside an expected range. The Spiffy system operator makes a summary of the reports every two months, and scans it for unusual patterns of access. A large number of accesses or an unusual pattern might reveal use of a manager's logon ID by an unauthorized employee.

### **Related concepts**

Chapter 9, "Auditing access to DB2," on page 281

"DB2 audit policy" on page 288

## **Securing access to payroll operations and management**

As a security measurement, the Spiffy security plan sets clear restrictions on how members of the payroll operations department access and handle sensitive payroll information.

The plan imposes the following restrictions on members of the payroll operations department:

- Members of the payroll operations department can update any column of the employee table except for SALARY, BONUS, and COMM.
- Members of payroll operations can update any row except for rows that are for members of their own department.

Because changes to the table are made only from the central location, distributed access does not affect payroll operations.

### **Creating views of payroll operations**

The Spiffy security planners decide to use views for implementing the security objectives for members of the payroll operations department.

The PAYDEPT view shows all the columns of the employee table except for job, salary, bonus, and commission. The view does not show the rows for members of the payroll operations department.

**Example:** The WORKDEPT value for the payroll operations department is P013. The owner of the employee table uses the following statement to create the PAYDEPT view:

```
CREATE VIEW PAYDEPT AS
  SELECT EMPNO, FIRSTNME, MIDINIT, LASTNAME, WORKDEPT,
         PHONENO, HIREDATE, JOB, EDLEVEL, SEX, BIRTHDATE
  FROM DSN81010.EMP
  WHERE WORKDEPT<>'P013'
  WITH CHECK OPTION;
```

The CHECK OPTION ensures that every row that is inserted or updated through the view conforms to the definition of the view.

A second view, the PAYMGR view, gives Spiffy payroll managers access to any record, including records for the members of the payroll operations department.

**Example:** The owner of the employee table uses the following statement to create the PAYMGR view:

```
CREATE VIEW PAYMGR AS
  SELECT EMPNO, FIRSTNME, MIDINIT, LASTNAME, WORKDEPT,
         PHONENO, HIREDATE, JOB, EDLEVEL, SEX, BIRTHDATE
  FROM DSN81010.EMP
  WITH CHECK OPTION;
```

Neither PAYDEPT nor PAYMGR provides access to compensation amounts. When a row is inserted for a new employee, the compensation amounts remain null. An update process can change these values at a later time. The owner of the employee table creates, owns, and grants privileges on both views.

### Securing compensation accounts with update tables

The Spiffy security plan does not allow members of payroll operations to update compensation amounts directly. Instead, a separate payroll update table contains the employee ID, job, salary, bonus, and commission.

Members of payroll operations make all job, salary, and bonus changes to the payroll update table, except those for their own department. After they verify the prospective changes, the managers of payroll operations run an application program. The program reads the payroll update table and makes the corresponding changes to the employee table. Only the payroll update program has the privilege of updating job, salary, and bonus in the employee table.

The Spiffy Computer Company calculates commission amounts separately by using a complicated formula. The formula considers the employee's job, department, years of service with the company, and responsibilities for various projects. The formula is embedded in the commission program, which is run regularly to insert new commission amounts in the payroll update table. The plan owner must have the SELECT privilege on the employee table and other tables to run the commission program.

### Securing compensation updates with other measures

By separating potential salary changes into the payroll update table, the Spiffy security planners allow payroll management to verify changes before they go into effect.

At Spiffy Computer Company, managers check the changes against a written change request that is signed by a required level of management. The Spiffy security planners consider that check to be the most important control on salary updates, but the plan also includes the following controls:

- The employee ID in the payroll update table is a foreign key column that refers to the employee ID in the employee table. Enforcing the referential constraint prevents an employee ID from being changed to an invalid value.
- The employee ID in the payroll update table is also a primary key for that table. Therefore, the values in the employee ID column must be unique. Because of enforced uniqueness, every change that is made for any one employee during a given operating period must appear in the same row of the table. No two rows can carry conflicting changes.

The Spiffy security plan documents an allowable range of salaries, bonuses, and commissions for each job level. To keep the values within the allowable ranges, the Spiffy security planners use table check constraints for the salaries, bonuses, and commissions. The planners use this approach because it is both simple and easy to control.

In a similar situation, you might also consider the following ways to ensure that updates and inserts stay within certain ranges:

- Keep the ranges in a separate DB2 table. To verify changes, query the payroll update table and the table of ranges. Retrieve any rows for which the planned update is outside the allowed range.
- Build the ranges into a validation routine. Apply the validation routine to the payroll update table to automatically reject any insert or update that is outside the allowed range.
- Embody the ranges in a view of the payroll table, using WITH CHECK OPTION, and make all updates to the view. The ID that owns the employee table also owns the view.
- Create a trigger to prevent salaries, bonuses, and commissions from increasing by more than the percent that is allowed for each job level.

### **Granting privileges to payroll operations and management**

The Spiffy security plan strongly suggests the functional approach for the payroll operations department.

The functional approach meets the security needs of the payroll operations for the following reasons:

- Payroll operations members require several privileges, including the SELECT, INSERT, UPDATE, and DELETE privileges on the PAYDEPT view.
- Several members of the department require the same set of privileges.
- If members of the department leave, others are hired or transferred to replace the departing members.

Therefore, the security plan calls for the creation of two RACF groups, with one for the payroll operations and another for the payroll management.

#### **Creating a RACF group for payroll operations:**

The Spiffy security plan calls for the creation of a RACF group for the payroll operations department. DB2USER can define the group and retain its ownership, or it can assign the ownership to an ID that is used by payroll management.

The owner of the employee table can grant the privileges that the group requires. The owner grants all required privileges to the group ID, with the intent not to revoke them. The primary IDs of new members of the department are connected to

the group ID, which becomes a secondary ID for each of them. The primary IDs of members who leave the department are disconnected from the group ID.

**Example:** The following statement grants the SELECT, INSERT, UPDATE, and DELETE privileges on the PAYDEPT view to the payroll operations group ID PAYOPS:

```
GRANT SELECT, INSERT, UPDATE, DELETE ON PAYDEPT TO PAYOPS;
```

This statement grants the privileges without the GRANT OPTION to keep members of payroll operations from granting privileges to other users.

### **Creating a RACF group for payroll management:**

The Spiffy payroll managers require different privileges and a different RACF group ID. The security planners add a RACF group for payroll managers and name it PAYMGRS.

The security planners associate the payroll managers' primary IDs with the PAYMGRS secondary ID. Next, privileges on the PAYMGR view, the compensation application, and the payroll update application are granted to PAYMGRS. The payroll update application must have the appropriate privileges on the update table.

**Example:** The following statement grants the SELECT, INSERT, UPDATE, and DELETE privileges on the PAYMGR view to the payroll managers' group ID PAYMGRS:

```
GRANT SELECT, INSERT, UPDATE, DELETE ON PAYMGR TO PAYMGRS;
```

**Example:** The following statement grants the EXECUTE privilege on the compensation application:

```
GRANT EXECUTE ON PLAN COMPENS TO PAYMGRS;
```

### **Auditing payroll operations and management**

You can create an audit policy for the payroll update table to audit payroll operation and management activities.

The audit trace records the number of accesses by the payroll operations and payroll management groups. The Spiffy security planners scan the reports of payroll access for large numbers or unusual patterns of access.

#### **Related concepts**

Chapter 9, "Auditing access to DB2," on page 281

"DB2 audit policy" on page 288

## **Managing access privileges of other authorities**

In addition to the privileges for the managers and the payroll operation and management personnel, the security plan considers the privileges for other roles.

### **Managing access by the DBADM authority**

An ID with the DBADM authority on a database has many privileges on that database and its tables. These privileges include the SELECT, INSERT, DELETE, UPDATE, and ALTER statements on any table in the database. They also include the CREATE and DROP statements on indexes for those tables.



For security reasons, the Spiffy security planners prefer not to grant all of the privileges that come with DBADM authority on DSN8D10A. DSN8D10A is the database that holds the employee table and the payroll update table.

The Spiffy security planners prefer to grant DBCTRL authority on the database because granting DBCTRL authority does not expose as many security risks as granting DBADM authority. DBCTRL authority allows an ID to support the database without allowing the ID to retrieve or change the data in the tables. However, database DSN8D10A contains several additional tables. These additional tables require some of the privileges that are included in DBADM authority but not included in DBCTRL authority.

The Spiffy security planners decide to compromise between the greater security of granting DBCTRL authority and the greater flexibility of granting DBADM authority. To balance the benefits of each authority, the Spiffy security planners create an administrative ID with some, but not all of the DBADM privileges. The security plan calls for a RACF group ID with the following authorities and privileges:

- DBCTRL authority over DSN8D81A
- The INDEX privilege on all tables in the database except the employee table and the payroll update table
- The SELECT, INSERT, UPDATE, and DELETE privileges on certain tables, excluding the employee table and the payroll update table

An ID with SYSADM authority grants the privileges to the group ID.

In a similar situation, you also might consider putting the employee table and the payroll update table in a separate database. Then you can grant DBADM authority on DSN8D10A, and grant DBCTRL authority on the database that contains the employee table and the payroll update table.

#### **Related reference**

“System DBADM” on page 40

“DBADM” on page 38

### **Managing access by the SYSADM authority**

An ID with SYSADM authority can access data from any table in the entire DB2 subsystem, including the employee table and the payroll update table. The Spiffy security planners want to minimize the security risk by granting the SYSADM authority to as few users as possible.

The planners know that the subsystem might require SYSADM authority only for certain tasks and only for relatively short periods. They also know that the privileges that are associated with the SYSADM authority give an ID control over all of the data in a subsystem.

To limit the number of users with SYSADM authority, the Spiffy security plan grants the authority to DB2OWNER, the ID that is responsible for DB2 security. That does not mean that only IDs that are connected to DB2OWNER can exercise privileges that are associated with SYSADM authority. Instead, DB2OWNER can grant privileges to a group, connect other IDs to the group as needed, and later disconnect them.

The Spiffy security planners prefer to have multiple IDs with SYSCTRL authority instead of multiple IDs with SYSADM authority. IDs with SYSCTRL authority can exercise most of the SYSADM privileges and can assume much of the day-to-day

work. IDs with SYSCTRL authority cannot access data directly or run plans unless the privileges for those actions are explicitly granted to them. However, they can run utilities, examine the output data sets, and grant privileges that allow other IDs to access data. Therefore, IDs with SYSCTRL authority can access some sensitive data, but they cannot easily access the data. As part of the Spiffy security plan, DB2OWNER grants SYSCTRL authority to selected IDs.

The Spiffy security planners also use ROLES, RACF group IDs, and secondary IDs to relieve the need to have SYSADM authority continuously available. SYSADM grants the necessary privileges to a ROLE, RACF group ID, or secondary ID. IDs that have this ROLE, RACF group ID, or secondary ID can then bind plans and packages it owns.

### **Managing access by object owners**

The Spiffy security plan must consider the ID that owns and grants privileges on the tables, views, and programs. The ID that owns these objects has many implicit privileges on the objects. The owner of the objects can also grant privileges on the objects to other users.

The Spiffy security planners want to limit the number of IDs that have privileges on the employee table and the payroll update table to the smallest convenient value. To meet that objective, they decide that the owner of the employee table should issue all of the CREATE VIEW and GRANT statements. They also decide to have the owner of the employee table own the plans and packages that are associated with employee data. The employee table owner implicitly has the following privileges, which the plans and packages require:

- The owner of the payroll update program must have the SELECT privilege on the payroll update table and the UPDATE privilege on the employee table.
- The owner of the commission program must have the UPDATE privilege on the payroll update table and the SELECT privilege on the employee table.
- The owners of several other payroll programs must have the proper privileges to do payroll processing, such as printing payroll checks, writing summary reports, and so on.

To bind these plans and packages, an ID must have the BIND or BINDADD privileges. The list of privileges that are required by the owner of the employee table suggests the functional approach. The Spiffy security planners create a RACF group for the owner of the employee table.

### **Managing access by other users**

Users must be authorized to access the employee table or the payroll table. Exceptions occur when any unauthorized user tries to access the tables.

The following users are authorized to access the employee and payroll tables:

- Department managers
- Members of the payroll operations department
- Payroll managers
- The payroll update program

The audit report lists each exception in full. Auditors check each exception to determine whether it was a planned operation by the users with SYSADM or DBADM authority, or the employee table owner.



The audit report also lists denials of access to the tables. Those denials represent attempts by unauthorized IDs to use the tables. Some are possibly accidental; others can be attempts to violate the security system.

After running the periodic reports, the security planners archive the audit records. The archives provide a complete audit trail of access to the employee data through DB2.



---

## Chapter 2. Managing access through authorization IDs and roles

DB2 controls access to its objects and data through authorization identifiers (IDs) and roles and the privileges that are assigned to them. Each privilege and its associated authorities enable you to take specific actions on an object. Therefore, you can manage access to DB2 objects through authorization IDs and roles.

As the following diagram shows, you can grant privileges and authorities to IDs or roles and control access to data and processes in several primary ways:

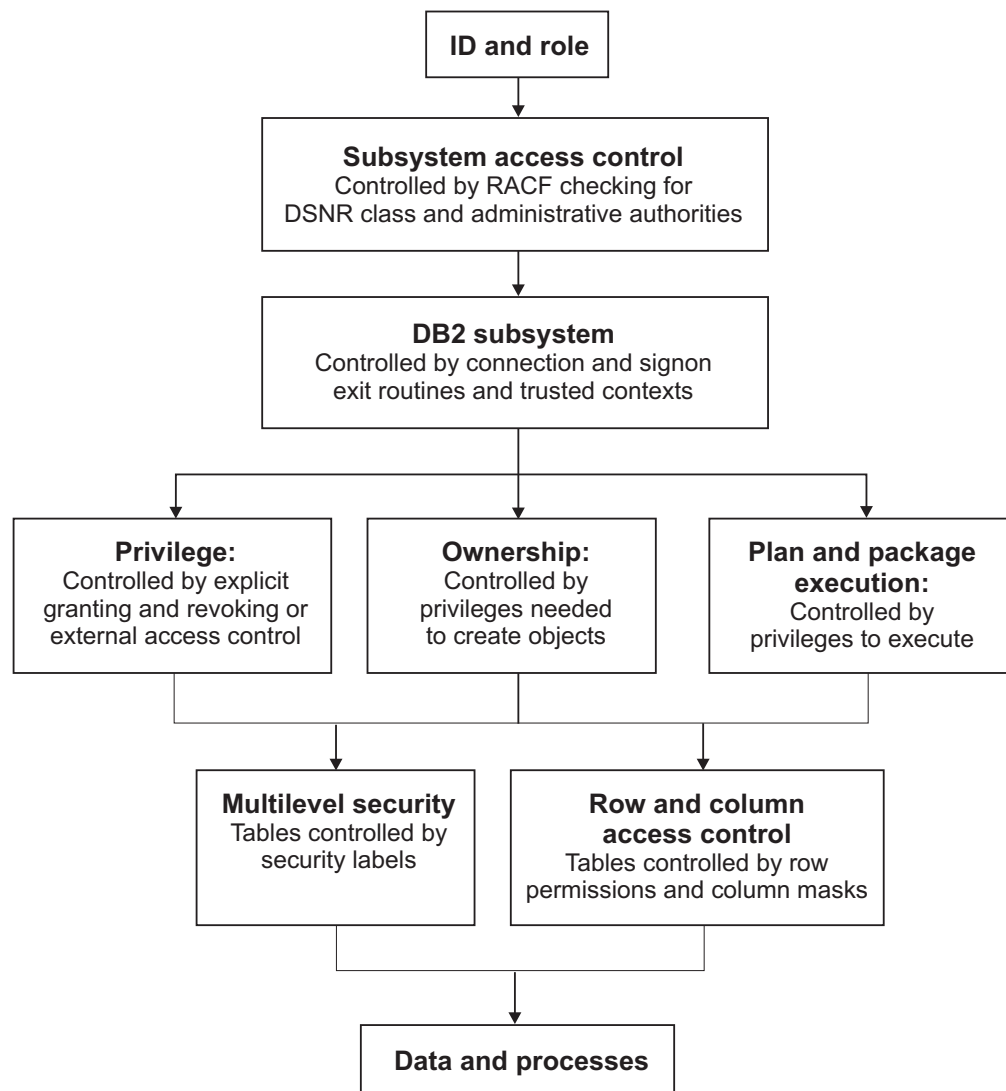


Figure 2. Access to objects and data within DB2

1. Managing access to DB2 through RACF and subsystem access authorization.
2. Managing access to DB2 subsystem through connection and sign-on routines or trusted contexts.

3. Granting and revoking explicit privileges through authorization IDs and roles or through external access control.

DB2 has primary authorization IDs, secondary authorization IDs, roles, and SQL IDs. Some privileges can be exercised by only one type of ID or a role; other privileges can be exercised by multiple IDs or roles. The DB2 catalog records the privileges that IDs are granted and the objects that IDs own.

4. Managing implicit privileges through ownership of objects other than plans and packages.
5. Managing implicit privileges through ownership of plans and packages.
6. Controlling access through security labels on tables.
7. Activating and deactivating row and column access control on tables.

Certain privileges and authorities are assigned when you install DB2. You can reassign these authorities by changing the DSNZPARM subsystem parameter.

As a security planner, you must be aware of these ways to manage privileges and authorities through authorization IDs and roles before you write a security plan. After you decide how to authorize access to data, you can implement it through your security plan.

---

## Authorization IDs and roles

You can control access to DB2 objects by assigning privileges and authorities to an authorization ID or a role.

### Authorization IDs

Every process that connects to or signs on to DB2 is represented by one or more DB2 short identifiers (IDs), which are called *authorization IDs*. Authorization IDs are assigned to a process by default procedures or by user-written exit routines.

When authorization IDs are assigned, every process receives exactly one ID that is called the *primary authorization ID*. All other IDs are *secondary authorization IDs*. Furthermore, one ID (either primary or secondary) is designated as the current *SQL ID*. You can change the value of the SQL ID during your session. More details about these IDs are as follows:

**Role** A role is available within a trusted context. You can define a role and assign it to authorization IDs in a trusted context. When associated with a role and using the trusted connection, an authorization ID inherits all the privileges granted to that role.

#### Primary authorization ID

Generally, the primary authorization ID identifies a process. For example, statistics and performance trace records use a primary authorization ID to identify a process.

#### Secondary authorization ID

A secondary authorization ID, which is optional, can hold additional privileges that are available to the process. For example, a secondary authorization ID can be a Resource Access Control Facility (RACF) group ID.

#### SQL ID

An SQL ID holds the privileges that are exercised when certain dynamic SQL statements are issued. The SQL ID can be set equal to the primary ID or any of the secondary IDs. If an authorization ID of a process has the

SYSADM authority and if the SEPARATE SECURITY system parameter on panel DSNTIPP1 is set to NO during installation, the process can set its SQL ID to any authorization ID. If the SEPARATE SECURITY parameter is set to YES, the SYSADM authority can set it to one of the secondary IDs only. This rule applies even when SET CURRENT SQLID is a static statement. CURRENT SQLID cannot be set to a role.

#### **RACF ID**

The RACF ID is generally the source of the primary and secondary authorization IDs (RACF groups). When you use the RACF Access Control Module or multilevel security, the RACF ID is used directly.

## **Roles in a trusted context**

A *role* is a database entity that groups one or more privileges together in a trusted context. System administrators can use roles to control access to enterprise objects in a way that parallels the structure of the enterprise.

A role is available only in a trusted context. A *trusted context* is an independent database entity that you can define based on a system authorization ID and connection trust attributes. The trust attributes specify a set of characteristics about a specific connection. These attributes include the IP address, domain name, or SERVAUTH security zone name of a remote client and the job or task name of a local client.

DB2 for z/OS extends the trusted context concept to allow for the assignment of a role to a trusted context. An authorization ID that uses the trusted context can inherit the privileges that are assigned to this role, in addition to the privileges that are granted to the ID. An authorization ID can have only one role in a trusted context at any given time.

Using roles provides the flexibility for managing context-specific privileges and simplifies the processing of authorization. Specific roles can be assigned to the authorization IDs that use the trusted connection. When your authorization ID is associated with an assigned role in the trusted context, you inherit all privileges that are granted by that role, instead of those by the default role, because the role-based privileges override the privileges that are associated with the default role.

#### **Related concepts**

“Trusted contexts” on page 205

“Trusted connections” on page 206

#### **Related tasks**

“Defining trusted contexts” on page 206

“Creating local trusted connections” on page 207

“Establishing remote trusted connections by DB2 for z/OS requesters” on page 208

“Establishing remote trusted connections to DB2 for z/OS servers” on page 209

---

## **Privileges and authorities**

You can control access within DB2 by granting or revoking privileges and related authorities that you assign to authorization IDs or roles. A *privilege* enables its holder to perform a specific operation, sometimes on a specific object.

Privileges can be explicit or implicit. An *explicit privilege* is a specific type of privilege. Each explicit privilege has a name and is the result of a GRANT statement or a REVOKE statement.

An *implicit privilege* comes from the ownership of objects, including plans and packages. For example, users are granted implicit privileges on objects that are referenced by a plan or package when they are authorized to execute the plan or package.

An administrative *authority* is a set of privileges, often covering a related set of objects. Authorities often include privileges that are not explicit, have no name, and cannot be specifically granted. For example, when an ID is granted the SYSOPR administrative authority, the ID is implicitly granted the ability to terminate any utility job.

## Explicit privileges

You can explicitly grant privileges on objects to authorization IDs or roles.

You can explicitly grant privileges on the following objects:

- Collections
- Databases
- Distinct types or JAR
- Functions or procedures
- Packages
- Plans
- Routines
- Schemas
- Sequences
- Systems
- Tables and views
- Usage
- Use

### Related concepts

“Privileges by authorization ID and authority” on page 45

### Related reference

“Implicit privileges through object ownership” on page 30

“Administrative authorities” on page 31

“Utility authorities for DB2 catalog and directory” on page 44

## Explicit collection privileges

You can explicitly grant privileges on collections.

### GUI

DB2 supports the following collection privileges:

Table 5. Explicit collection privileges

Collection privilege	Operations allowed for a named package collection
CREATE IN	The BIND PACKAGE subcommand, to name the collection

### GUI

## Explicit database privileges

You can explicitly grant privileges on databases.

GUIP

DB2 supports the following database privileges:

Table 6. Explicit database privileges

Database privilege	Operations allowed on a named database
CREATETAB	The CREATE TABLE statement, to create tables in the database.
CREATETS	The CREATE TABLESPACE statement, to create table spaces in the database
DISPLAYDB	The DISPLAY DATABASE command, to display the database status
DROP	The DROP and ALTER DATABASE statements, to drop or alter the database
IMAGCOPY	The QUIESCE, COPY, and MERGECOPY utilities, to prepare for, make, and merge copies of table spaces in the database; the MODIFY RECOVERY utility, to remove records of copies
LOAD	The LOAD utility, to load tables in the database
RECOVERDB	The RECOVER, REBUILD INDEX, and REPORT utilities, to recover objects in the database and report their recovery status
REORG	The REORG utility, to reorganize objects in the database
REPAIR	The REPAIR and DIAGNOSE utilities (except REPAIR DBD and DIAGNOSE WAIT) to generate diagnostic information about, and repair data in, objects in the database
STARTDB	The START DATABASE command, to start the database
STATS	The RUNSTATS, CHECK, LOAD, REBUILD INDEX, REORG INDEX, and REORG TABLESPACE, and MODIFY STATISTICS utilities, to gather statistics, check indexes and referential constraints for objects in the database, and delete unwanted statistics history records from the corresponding catalog tables
STOPDB	The STOP DATABASE command, to stop the database

Database privileges that are granted on DSNDB04 apply to all implicitly created databases. For example, if you have the DBADM authority on DSNDB04, you can select data from any table in any implicitly created database. If you have the STOPDB privilege on DSNDB04, you can stop any implicitly created database. However, you cannot grant the same authorities or privileges to others on any implicitly created database.

GUIP

## Explicit package privileges

You can explicitly grant privileges on packages.

GUIP

DB2 supports the following package privileges:

Table 7. Explicit package privileges

Package privilege	Operations allowed for a named package
BIND	The BIND, REBIND, and FREE PACKAGE subcommands, and the DROP PACKAGE statement, to bind or free the package, and, depending on the installation option BIND NEW PACKAGE, to bind a new version of a package
COPY	The COPY option of BIND PACKAGE, to copy a package
EXECUTE	Inclusion of the package in the PKLIST option of BIND PLAN
GRANT ALL	All package privileges

#### GUIP

### Explicit plan privileges

You can explicitly grant privileges on plans.

#### GUIP

DB2 supports the following plan privileges:

Table 8. Explicit plan privileges

Plan privilege	Subcommands allowed for a named application plan
BIND	BIND, REBIND, and FREE PLAN, to bind or free the plan
EXECUTE	RUN, to use the plan when running the application

#### GUIP

### Explicit routine privileges

You can explicitly grant privileges on routines.

#### GUIP

DB2 supports the following routine privileges:

Table 9. Explicit routine privileges

Routine privileges	Objects available for usage
EXECUTE ON FUNCTION	A user-defined function
EXECUTE ON PROCEDURE	A stored procedure

#### GUIP

### Explicit schema privileges

You can explicitly grant privileges on schemas.

#### GUIP

DB2 supports the following schema privileges:



Table 10. Explicit schema privileges

Schema privileges	Operations available for usage
CREATEIN	Create distinct types, user-defined functions, triggers, and stored procedures in the designated schemas
ALTERIN	Alter user-defined functions or stored procedures, or specify a comment for distinct types, user-defined functions, triggers, and stored procedures in the designated schemas
DROPIN	Drop distinct types, user-defined functions, triggers, and stored procedures in the designated schemas

## GUIP

### Explicit system privileges

You can explicitly grant privileges on systems.

## GUIP

DB2 supports the following system privileges:

Table 11. Explicit system privileges

System privilege	Operations allowed on the system
ARCHIVE	The ARCHIVE LOG command, to archive the current active log, the DISPLAY ARCHIVE command, to give information about input archive logs, the SET LOG command, to modify the checkpoint frequency specified during installation, and the SET ARCHIVE command, to control allocation and deallocation of tape units for archive processing.
BINDADD	The BIND subcommand with the ADD option, to create new plans and packages
BINDAGENT	The BIND, REBIND, and FREE subcommands, and the DROP PACKAGE statement, to bind, rebind, or free a plan or package, or copy a package, on behalf of the grantor. The BINDAGENT privilege is intended for separation of function, not for added security. A bind agent with the EXECUTE privilege might be able to gain all the authority of the grantor of BINDAGENT.
BSDS	The RECOVER BSDS command, to recover the bootstrap data set
CREATEALIAS	The CREATE ALIAS statement, to create an alias for a table or view name
CREATEDBA	The CREATE DATABASE statement, to create a database and have DBADM authority over it
CREATEDBC	The CREATE DATABASE statement, to create a database and have DBCTRL authority over it
CREATESG	The CREATE STOGROUP statement, to create a storage group
CREATE_SECURE_OBJECT	The CREATE and ALTER statements, to create secure objects, such as a secure trigger or a user-defined function. If a trigger is defined for tables that are enforced with row or column access control, it must be secure. If a user-defined function is referenced in the definition of a row permission or column mask, it must be secure. In addition, if a user-defined function is invoked in a query and its arguments reference columns with column masks, the user-defined function must be secure.

Table 11. Explicit system privileges (continued)

System privilege	Operations allowed on the system
CREATETMTAB	The CREATE GLOBAL TEMPORARY TABLE statement, to define a created temporary table
DEBUGSESSION	The DEBUGINFO connection attribute, to control debug session activity for SQL stored procedures, non-inline SQL functions, and Java stored procedures
DISPLAY	The DISPLAY ARCHIVE, DISPLAY BUFFERPOOL, DISPLAY DATABASE, DISPLAY LOCATION, DISPLAY LOG, DISPLAY THREAD, and DISPLAY TRACE commands, to display system information
EXPLAIN	<ul style="list-style-type: none"> <li>The SQL EXPLAIN PLAN and EXPLAIN ALL statements, to issue the statements without requiring additional privileges</li> <li>The SQL PREPARE and DESCRIBE TABLE statements, to prepare and describe the statements without requiring additional privileges on the object</li> <li>The BIND command, to allow users to specify the EXPLAIN(ONLY) and SQLERROR(CHECK) options without creating a plan or package</li> <li>Dynamic SQL statements that have the special register CURRENT EXPLAIN MODE set to EXPLAIN, to allow the capture of information about the statements, without executing them</li> </ul> <p>An authorization ID or role with any of the following authority or privilege can grant the EXPLAIN privilege:</p> <ul style="list-style-type: none"> <li>The SECADM authority</li> <li>The ACCESSCTRL authority</li> <li>The SYSADM authority if the SEPARATE SECURITY system parameter is set to NO at the installation</li> <li>The EXPLAIN privilege with the WITH GRANT OPTION.</li> </ul>
MONITOR1	Receive trace data that is not potentially sensitive
MONITOR2	Receive all trace data
RECOVER	The RECOVER INDOUBT command, to recover threads
STOPALL	The STOP DB2 command, to stop DB2
STOSPACE	The STOSPACE utility, to obtain data about space usage
TRACE	The START TRACE, STOP TRACE, and MODIFY TRACE commands, to control tracing

#### GUPI

### Explicit table and view privileges

You can explicitly grant privileges on tables and views.

#### GUPI

DB2 supports the following table and view privileges:

Table 12. Explicit table and view privileges

Table or view privilege	SQL statements allowed for a named table or view
ALTER	ALTER TABLE, to change the table definition
DELETE	DELETE, to delete rows
INDEX	CREATE INDEX, to create an index on the table
INSERT	INSERT, to insert rows
REFERENCES	ALTER or CREATE TABLE, to add or remove a referential constraint that refers to the named table or to a list of columns in the table
SELECT	SELECT, to retrieve data
TRIGGER	CREATE TRIGGER, to define a trigger on a table
UPDATE	UPDATE, to update all columns or a specific list of columns
GRANT ALL	SQL statements of all privileges

#### GUPI

### Explicit usage privileges

You can explicitly grant privileges on usage.

#### GUPI

DB2 supports the following usage privileges:

Table 13. Explicit usage privileges

Usage privileges	Objects available for usage
USAGE ON DISTINCT TYPE	A distinct type
USAGE ON JAR (Java class for a routine)	A Java class
USAGE ON SEQUENCE	A sequence

#### GUPI

### Explicit use privileges

You can explicitly grant privileges on use.

#### GUPI

DB2 supports the following use privileges:

Table 14. Explicit use privileges

Use privileges	Objects available for use
USE OF BUFFERPOOL	A buffer pool
USE OF STOGROUP	A storage group
USE OF TABLESPACE	A table space

## Implicit privileges through object ownership

When you create a DB2 object by issuing an SQL statement, you establish its name and its ownership. By default, the owner implicitly holds certain privileges on the object.

**GUIP** However, this general rule does not apply to a plan or package that is not created with SQL CREATE statements. In other words, when you own an object other than a plan or package, you have implicit privileges over the object. The following table describes the implicit privileges of ownership for each type of object:

*Table 15. Implicit privileges of ownership by object type*

Object type	Implicit privileges of ownership
Alias	To drop the alias
Database	DBCTRL or DBADM authority over the database, depending on the privilege (CREATEDBC or CREATEDBA) that is used to create it. DBCTRL authority does <b>not</b> include the privilege to access data in tables in the database.
Distinct type	To use or drop a distinct type
Index	To alter, comment on, or drop the index
JAR (Java class for a routine)	To replace, use, or drop the JAR
Package	To bind, rebind, free, copy, execute, drop, or comment on the package
Plan	To bind, rebind, free, execute, or comment on the plan
Role	To create, alter, commit, drop, or comment on the role
Sequence	To alter, comment on, use, or drop the sequence
Storage group	To alter or drop the group and to name it in the USING clause of a CREATE INDEX or CREATE TABLESPACE statement
Stored procedure	To execute, alter, drop, start, stop, or display a stored procedure
Synonym	To use or drop the synonym
Table	<ul style="list-style-type: none"> <li>• To alter or drop the table or any indexes on it</li> <li>• To lock the table, comment on it, or label it</li> <li>• To create an index or view for the table</li> <li>• To select or update any column (if there is no row permission or column mask defined or if the row permission and the column mask definition allows the access)</li> <li>• To insert, delete, select, or update any row (if there is no row permission defined or if the row permission definition allows the access)</li> <li>• To use the LOAD utility for the table</li> <li>• To define referential constraints on any table or set of columns</li> <li>• To create a trigger on the table</li> <li>• To comment on the table</li> </ul>
Table space	To alter or drop the table space and to name it in the IN clause of a CREATE TABLE statement
Trusted context	To create, alter, commit, revoke, or comment on the trusted context
User-defined functions	To execute, alter, drop, start, stop, or display a user-defined function

Table 15. Implicit privileges of ownership by object type (continued)

Object type	Implicit privileges of ownership
View	<ul style="list-style-type: none"> <li>To drop, comment on, or label the view, or to select any row or column</li> <li>To execute UPDATE, INSERT, or DELETE on the view if the view is defined with the INSTEAD OF TRIGGER clause</li> </ul>

#### GUIP

#### Related concepts

“Explicit privileges” on page 24

“Privileges by authorization ID and authority” on page 45

#### Related reference

“Administrative authorities”

“Utility authorities for DB2 catalog and directory” on page 44

## Administrative authorities

Within DB2, privileges are grouped into administrative authorities, and each administrative authority is vested with a specific set of privileges.

#### GUIP

The following table lists all of the DB2 for z/OS administrative authorities and the grantable privileges that each of them has.

Table 16. Administrative authorities and grantable privileges

Authority	Included authorities	Additional grantable privileges
ACCESSCTRL	None	Privileges on all catalog tables: SELECT  Privileges on updatable catalog tables (except SYSIBM.SYSAUDITPOLICIES): DELETE    INSERT    UPDATE  Privileges on security: GRANT    REVOKE

Table 16. Administrative authorities and grantable privileges (continued)

Authority	Included authorities	Additional grantable privileges
DATAACCESS	None	<p>System privileges:</p> <p>DEBUGSESSION</p> <p>Privileges on all user tables, views, and MQTs:</p> <p>DELETE    INSERT    SELECT    UPDATE</p> <p>Privileges on all plans, packages, and routines:</p> <p>EXECUTE</p> <p>Privileges on all user databases:</p> <p>LOAD    RECOVERDB    REORG    REPAIR</p> <p>Privileges on all JARs:</p> <p>USAGE</p> <p>Privileges on all sequences:</p> <p>USAGE</p> <p>Privileges on all distinct types:</p> <p>USAGE</p> <p>Privileges on all catalog tables:</p> <p>SELECT</p> <p>Privileges on updatable catalog tables (except SYSIBM.SYSAUDITPOLICIES):</p> <p>DELETE    INSERT    UPDATE</p>
DBADM	DBCTRL, DBMAINT	<p>Privileges on tables in a database:</p> <p>ALTER            DELETE    INDEX    INSERT REFERENCES    SELECT    TRIGGER    UPDATE</p>
DBCTRL	DBMAINT	<p>Privileges on a database:</p> <p>DROP    LOAD    RECOVERDB    REORG REPAIR</p>
DBMAINT	None	<p>Privileges on a database:</p> <p>CREATETAB    CREATETS    DISPLAYDB IMAGCOPY    STATS    STARTDB    STOPDB</p>
Installation SYSADM	SYSADM, SYSCTRL, DBADM, Installation SYSOPR, SYSOPR, PACKADM, DBCTRL, DBMAINT, SECADM, System DBADM, SQLADM, ACCESSCTRL, DATAACCESS	<p>Privileges on security:</p> <p>GRANT    REVOKE</p>
Installation SYSOPR	SYSOPR	<p>Privileges:</p> <p>ARCHIVE    STARTDB(<i>cannot alter access mode</i>)</p>

Table 16. Administrative authorities and grantable privileges (continued)

Authority	Included authorities	Additional grantable privileges
PACKADM	None	Privileges on a collection: CREATEIN  Privileges on all packages in a collection: BIND COPY EXECUTE
SECADM	ACCESSCTRL	Privileges on all catalog tables: SELECT  Privileges on all updatable catalog tables: DELETE INSERT UPDATE  Privileges on security: GRANT REVOKE  Privileges on security-related objects: ALTER CREATE DROP
SQLADM	None	System privileges: EXPLAIN MONITOR1 MONITOR2  Privileges on all catalog tables: SELECT  Privileges on updatable catalog tables (except SYSIBM.SYSAUDITPOLICIES): DELETE INSERT UPDATE
SYSADM	SYSCTRL, DBADM, Installation SYSOPR, SYSOPR, PACKADM, DBCTRL, DBMAINT, SECADM, System DBADM, SQLADM, ACCESSCTRL, DATAACCESS	Privileges on all plans: EXECUTE  Privileges on all routines: EXECUTE  Privileges on all packages: All privileges  Privileges on distinct types: USAGE  Privileges on sequences: USAGE  System privileges: DEBUGSESSION  EXPLAIN privilege

Table 16. Administrative authorities and grantable privileges (continued)

Authority	Included authorities	Additional grantable privileges																											
SYSCTRL	Installation SYSOPR, SYSOPR, DBCTRL, DBMAINT, ACCESSCTRL (except the ability to grant certain authorities, such as DBADM, SYSADM, PACKADM, and certain privileges, such as DELETE, INSERT, SELECT, and UPDATE on user tables or views, EXECUTE on plans, packages, functions, or stored procedures, PACKADM on collections, and USAGE on distinct types, JARs, and sequences)	<div>System privileges:</div> <table><tr><td>BINDADD</td><td>BINDAGENT</td><td>DBDS</td></tr><tr><td>CREATEALIAS</td><td>CREATEDBA</td><td>CREATEDBC</td></tr><tr><td>CREATESG</td><td>CREATEMTAB</td><td>MONITOR1</td></tr><tr><td>MONITOR2</td><td>STOSPACE</td><td></td></tr></table> <div>Privileges on all tables:</div> <table><tr><td>ALTER</td><td>INDEX</td><td>REFERENCES</td><td>TRIGGER</td></tr></table> <div>Privileges on all catalog tables:</div> <div>SELECT</div> <div>Privileges on updatable catalog tables (except SYSIBM.SYSAUDITPOLICIES):</div> <table><tr><td>DELETE</td><td>INSERT</td><td>UPDATE</td></tr></table> <div>Privileges on all plans:</div> <div>BIND</div> <div>Privileges on all packages:</div> <table><tr><td>BIND</td><td>COPY</td></tr></table> <div>Privileges on all collections:</div> <div>CREATEIN</div> <div>Privileges on all schemas:</div> <table><tr><td>ALTERIN</td><td>CREATEIN</td><td>DROPIN</td></tr></table> <div>Privileges on use:</div> <table><tr><td>BUFFERPOOLS</td><td>STOGROUP</td><td>TABSPACE</td></tr></table>	BINDADD	BINDAGENT	DBDS	CREATEALIAS	CREATEDBA	CREATEDBC	CREATESG	CREATEMTAB	MONITOR1	MONITOR2	STOSPACE		ALTER	INDEX	REFERENCES	TRIGGER	DELETE	INSERT	UPDATE	BIND	COPY	ALTERIN	CREATEIN	DROPIN	BUFFERPOOLS	STOGROUP	TABSPACE
BINDADD	BINDAGENT	DBDS																											
CREATEALIAS	CREATEDBA	CREATEDBC																											
CREATESG	CREATEMTAB	MONITOR1																											
MONITOR2	STOSPACE																												
ALTER	INDEX	REFERENCES	TRIGGER																										
DELETE	INSERT	UPDATE																											
BIND	COPY																												
ALTERIN	CREATEIN	DROPIN																											
BUFFERPOOLS	STOGROUP	TABSPACE																											
SYSOPR	None	<div>Privileges:</div> <table><tr><td>DISPLAY</td><td>RECOVER</td><td>STOPALL</td><td>TRACE</td></tr></table> <div>Privileges on routines:</div> <table><tr><td>DISPLAY</td><td>START</td><td>STOP</td></tr></table>	DISPLAY	RECOVER	STOPALL	TRACE	DISPLAY	START	STOP																				
DISPLAY	RECOVER	STOPALL	TRACE																										
DISPLAY	START	STOP																											



Table 16. Administrative authorities and grantable privileges (continued)

Authority	Included authorities	Additional grantable privileges																																				
System DBADM	SQLADM	<div>System privileges:</div> <table><tr><td>BINDADD</td><td>BINDAGENT</td><td>CREATEALIAS</td></tr><tr><td>CREATEDBA</td><td>CREATEDBC</td><td>CREATETMTAB</td></tr><tr><td>DISPLAY</td><td>EXPLAIN</td><td>MONITOR1</td></tr><tr><td>MONITOR2</td><td>SQLADM</td><td>STOPALL</td></tr><tr><td>TRACE</td><td></td><td></td></tr></table> <div>Privileges on all collections:</div> <div>CREATEIN</div> <div>Privileges on all user databases:</div> <table><tr><td>CREATETAB</td><td>CREATETS</td><td>DISPLAYDB</td></tr><tr><td>DROP</td><td>IMAGCOPY</td><td>RECOVERDB</td></tr><tr><td>STARTDB</td><td>STOPDB</td><td></td></tr></table> <div>Privileges on all user tables (except for those defined with row permissions or column masks):</div> <table><tr><td>ALTER</td><td>INDEX</td><td>REFERENCES</td><td>TRIGGER</td></tr></table> <div>Privileges on all packages:</div> <table><tr><td>BIND</td><td>COPY</td></tr></table> <div>Privileges on all plans:</div> <div>BIND</div> <div>Privileges on all schemas:</div> <table><tr><td>ALTERIN</td><td>CREATEIN</td><td>DROPIN</td></tr></table> <div>Privileges on all sequences:</div> <div>ALTER</div> <div>Privileges on all distinct types:</div> <div>USAGE</div> <div>Privileges on use:</div> <div>TABLESPACE</div> <div>Privileges on all catalog tables:</div> <div>SELECT</div> <div>Privileges on updatable catalog tables (except SYSIBM.SYSAUDITPOLICIES):</div> <table><tr><td>DELETE</td><td>INSERT</td><td>UPDATE</td></tr></table>	BINDADD	BINDAGENT	CREATEALIAS	CREATEDBA	CREATEDBC	CREATETMTAB	DISPLAY	EXPLAIN	MONITOR1	MONITOR2	SQLADM	STOPALL	TRACE			CREATETAB	CREATETS	DISPLAYDB	DROP	IMAGCOPY	RECOVERDB	STARTDB	STOPDB		ALTER	INDEX	REFERENCES	TRIGGER	BIND	COPY	ALTERIN	CREATEIN	DROPIN	DELETE	INSERT	UPDATE
BINDADD	BINDAGENT	CREATEALIAS																																				
CREATEDBA	CREATEDBC	CREATETMTAB																																				
DISPLAY	EXPLAIN	MONITOR1																																				
MONITOR2	SQLADM	STOPALL																																				
TRACE																																						
CREATETAB	CREATETS	DISPLAYDB																																				
DROP	IMAGCOPY	RECOVERDB																																				
STARTDB	STOPDB																																					
ALTER	INDEX	REFERENCES	TRIGGER																																			
BIND	COPY																																					
ALTERIN	CREATEIN	DROPIN																																				
DELETE	INSERT	UPDATE																																				



### Related concepts

“Explicit privileges” on page 24

“Privileges by authorization ID and authority” on page 45

### Related reference

“Implicit privileges through object ownership” on page 30

“Utility authorities for DB2 catalog and directory” on page 44

## Installation SYSADM

Installation SYSADM authority is assigned to one or two IDs when DB2 is installed; it cannot be assigned to a role. These IDs have all the privileges of the SYSADM authority.

**GUPI** No other IDs can revoke the installation SYSADM authority; you can remove the authority only by changing the module that contains the subsystem initialization parameters (typically DSNZPARM).

In addition, DB2 does not record the installation SYSADM authority in the catalog. Therefore, the catalog does not need to be available to check installation SYSADM authority. The authority outside of the catalog is crucial. For example, if the directory table space DBD01 is stopped, DB2 might not be able to check the authority to start it again. In this case, only an installation SYSADM can start it.

IDs with the installation SYSADM authority can also perform the following actions:

- Run the CATMAINT utility
- Access DB2 when the subsystem is started with ACCESS(MAINT)
- Start databases DSNDB01 and DSNDB06 when they are stopped or in restricted status
- Run the DIAGNOSE utility with the WAIT statement
- Start and stop the database that contains the application registration table (ART) and the object registration table (ORT).
- Grant, revoke, and manage security-related objects regardless of the setting of the SEPARATE\_SECURITY system parameter. **GUPI**

## SYSADM

The SYSADM authority includes all the privileges, including system privileges, for creating objects and accessing all data. Depending on the setting of the SEPARATE\_SECURITY system parameter, the SYSADM authority can also create security objects and grant and revoke privileges.

**GUPI** Regardless of the SEPARATE\_SECURITY setting, an authorization ID or role with the SYSADM authority can perform the following actions. If SEPARATE\_SECURITY is set to NO, it can also grant other IDs the required privileges to perform the same actions.

- Use all the privileges of DBADM over any database
- Use EXECUTE privileges on all packages
- Use EXECUTE privileges on all routines
- Use USAGE privilege on distinct types, JARs, and sequences
- Use BIND on any plan and COPY on any package
- Use privileges over views that are owned by others

- Create and drop synonyms and views for other IDs on any table
- Drop database DSNDB07

An authorization ID or role with the SYSADM authority can also perform the following actions but cannot grant other IDs the privileges to perform them:

- Drop or alter any DB2 object, except system databases
- Issue a COMMENT ON statement for any table, view, index, column, package, plan
- Issue a LABEL ON statement for any table or view
- Terminate any utility job
- Create roles and trusted contexts (if SEPARATE\_SECURITY is set to NO)
- Set the current SQL ID to any valid value (if SEPARATE\_SECURITY is set to NO)
- Use any valid value for OWNER in BIND or REBIND (if SEPARATE\_SECURITY is set to NO)

Although an authorization ID or role with the SYSADM authority cannot grant the preceding privileges explicitly, it can accomplish this goal by granting to other IDs the SYSADM authority.

Regardless of the SEPARATE\_SECURITY setting, an authorization ID or role with the SYSADM authority can revoke any privileges that were granted by itself. When SEPARATE\_SECURITY is set to NO, the same ID or role can also revoke privileges that were granted by others. However, when SEPARATE\_SECURITY is set to YES,

the same ID or role cannot revoke privileges that were granted by others. 

## **SYSCTRL**

The SYSCTRL authority is designed for administering a system that contains sensitive data. With the SYSCTRL authority, you have nearly complete control of the DB2 subsystem. However, you cannot access user data directly unless you are explicitly granted the privileges to do so.


 Regardless of the SEPARATE\_SECURITY setting, an authorization ID or role with the SYSCTRL authority can perform the following actions:

- Act as installation SYSOPR (when the catalog is available) or DBCTRL over any database
- Run any allowable utility on any database
- Issue a COMMENT ON, LABEL ON, or LOCK TABLE statement for any table
- Create a view on any catalog table for itself or for other IDs
- Create tables and aliases for itself or for others IDs
- Bind a new plan or package and name any ID as the owner of the plan or package
- Create roles (only if SEPARATE\_SECURITY is set to NO)
- Use any valid value for OWNER in BIND or REBIND (only if SEPARATE\_SECURITY is set to NO)
- Has implicit ACCESSCTRL authority to grant most privileges (only if SEPARATE\_SECURITY is set to NO)

However, you cannot perform the following actions without the required additional privileges:


- Execute SQL statements that change data in any user tables or views
- Run plans or packages

- Set the current SQL ID to a value that is not one of its primary or secondary IDs
- Start or stop the database that contains the application registration table (ART) and the object registration table (ORT)
- Act fully as SYSADM or as DBADM over any database
- Access DB2 when the subsystem is started with ACCESS(MAINT)

The SYSCTRL authority is intended to separate system control functions from administrative functions. However, SYSCTRL is not a complete solution for a high-security system. If any plans have their EXECUTE privilege granted to PUBLIC, an ID or role with the SYSCTRL authority can grant itself the SYSADM authority. The only control over such actions is to audit the activity of IDs with high levels of authority. 


## Installation SYSOPR

Installation SYSOPR authority is assigned to one or two IDs when DB2 is installed; it cannot be assigned to a role. These IDs have all the privileges of the SYSOPR authority.

 No IDs can revoke the installation SYSOPR authority; you can remove it only by changing the module that contains the subsystem initialization parameters (typically DSNZPARM).


In addition, the installation SYSOPR authority is not recorded in the DB2 catalog. Therefore, the catalog does not need to be available to check the installation SYSOPR authority.

IDs with the installation SYSOPR authority can perform the following actions:

- Access DB2 when the subsystem is started with ACCESS(MAINT).
- Run all allowable utilities on the directory and catalog databases (DSNDB01 and DSNDB06).
- Run the REPAIR utility with the DBD statement.
- Start and stop the database that contains the application registration table (ART) and the object registration table (ORT).
- Issue dynamic SQL statements that are not controlled by the DB2 governor.
- Issue a START DATABASE command to recover objects that have LPL entries or group buffer pool RECOVERY-pending status. These IDs cannot change the access mode. 

## SYSOPR

A user with the SYSOPR authority can issue all DB2 commands except ARCHIVE LOG, START DATABASE, STOP DATABASE, and RECOVER BSDS.

 In addition, that user can run the DSN1SDMP utility and terminate any utility job. With the GRANT option, that user can grant these privileges to others.



## DBADM

The DBADM authority includes the DBCTRL privileges over a specific database. A user with the DBADM authority can access any tables in a specific database by using SQL statements.

**GUIP** With the DBADM authority, you can also perform the following actions:

- Drop or alter any table space, table, or index in the database
- Issue a COMMENT, LABEL, or LOCK TABLE statement for any table in the database
- Issue a COMMENT statement for any index in the database

If the value of the DBADM CREATE AUTH field on the DSNTIPP installation panel is set to YES during the DB2 installation, an ID with the DBADM authority can create the following objects:

- A view for another ID. The view must be based on at least one table, and that table must be in the database under DBADM authority.
- An alias for another ID on any table in the database.

An ID with DBADM authority on one database can create a view on tables and views in that database and other databases only if the ID has all the privileges that are required to create the view. For example, an ID with DBADM authority cannot create a view on a view that is owned by another ID.

If a user has the DBADM authority with the GRANT option, that user can grant these privileges to others. **GUIP**

## DBCTRL

The DBCTRL authority includes the DBMAINT privileges on a specific database. A user with the DBCTRL authority can run utilities that can change the data.

**GUIP** If the value of the DBADM CREATE AUTH field on the DSNTIPP installation panel is set to YES during the DB2 installation, an ID with DBCTRL authority can create an alias for another user ID on any table in the database.

If a user has the DBCTRL authority with the GRANT option, that user can grant those privileges to others. **GUIP**

## DBMAINT

A user with the DBMAINT authority can grant the privileges on a specific database to an ID.

**GUIP** With the DBMAINT authority, that user can perform the following actions within that database:

- Create objects
- Run utilities that don't change data
- Issue commands
- Terminate all utilities on the database except DIAGNOSE, REPORT, and STOSPACE

If a user has the DBMAINT authority with the GRANT option, that user can grant those privileges to others. **GUIP**

## PACKADM

The PACKADM authority has the package privileges on all packages in specific collections and the CREATE IN privilege on these collections.

**GUPI** If the BIND NEW PACKAGE installation option is set to BIND, the PACKADM authority also has the privilege to add new packages or new versions of existing packages.

If a user has the PACKADM authority with the GRANT option, that user can grant those privileges to others. **GUPI**

## System DBADM

The system DBADM authority allows an administrator, an authorization ID or a role, to manage databases across a DB2 subsystem, while having no access to the data in the databases. In other words, the system DBADM authority enables you to create, alter, and drop DB2 objects and issue commands for a DB2 subsystem, but does not give you the authority to access the data or the ability to grant or revoke privileges.

**GUPI** With the system DBADM authority, you can issue SQL statements to perform the following tasks:


- Create and drop aliases, auxiliary tables, and distinct types
- Create, alter, and drop databases, tables, global temporary tables, table spaces, and sequences
- Create triggers, functions, indexes, procedures, and views with additional required privileges
- Comment on all but security-related objects (i.e., roles, trusted contexts)
- Issue other SQL statements, such as the EXPLAIN, LABEL, PREPARE, and RENAME statements

You can also issue DB2 commands to perform the following tasks:

- Display status, configuration, and resource information
- Start and stop procedures and profiles
- Start, stop, and access databases
- Start, stop, and modify traces
- Bind, rebind, and free packages and plans
- Set the OWNER in BIND or REBIND to any ID (if SEPARATE\_SECURITY is set to NO)
- Alter and terminate the execution of DB2 utility job steps
- Recover threads that are left in an indoubt state or complete backout processing for units of recovery that are left incomplete during an earlier restart


With the system DBADM authority, you can also run certain DB2 utilities. The utilities include CHECK INDEX, CHECK LOB, COPY, COPYTOCOPY, DIAGNOSE, MODIFY RECOVERY, MODIFY STATISTICS, QUIESCE, REBUILD INDEX, RECOVER, REPORT, and RUNSTATS. In addition, you have implicit SELECT access on all catalog tables and implicit INSERT, DELETE, and UPDATE privileges on updatable catalog tables (except SYSIBM.SYSAUDITPOLICIES).

The system DBADM authority allows you to execute system-defined routines (recorded in the SYSIBM.SYSROUTINES catalog table), including stored procedures or functions, and any packages executed within the routines. It also allows you to drop non-security objects without requiring the ownership or other privileges to drop.


Only an authorization ID or a role with the SECADM authority can grant or revoke the system DBADM authority. By default, the system DBADM has all the privileges of the DATAACCESS and ACCESSCTRL authorities. If you do not want a user (an authorization ID or role) with the system DBADM authority to grant any explicit privileges, you can specify the WITHOUT ACCESSCTRL clause in the GRANT statement when you grant the authority. If you do not want a user with the system DBADM authority to access any user data in the databases, you can specify the WITHOUT DATAACCESS clause in the GRANT statement when you grant the authority. If necessary, you can still grant explicit privileges (i.e., SELECT) to the system DBADM user to access data or perform grants. 

## SECADM


The SECADM authority enables you to manage security-related objects in DB2 and control access to all database resources. It does not have any inherent privilege to access data stored in the objects, such as tables.

 With the SECADM authority, you can perform the following tasks:

- Create, alter, drop, and comment on row permissions
- Create, alter, drop, and comment on column masks
- Activate and deactivate row access control
- Activate and deactivate column access control
- Create, drop, and comment on roles
- Create, alter, drop, and comment on trusted contexts
- Create and comment on secure triggers and user-defined functions
- Alter the SECURED or NOT SECURED clause on triggers and user-defined functions
- Create audit policies by inserting rows into the SYSIBM.SYSAUDITPOLICIES catalog table
- Access and update the SYSIBM.SYSAUDITPOLICIES catalog table which records audit policy definitions
- Has implicit SELECT access on all catalog tables and implicit INSERT, DELETE, and UPDATE privileges on updatable catalog tables
- Grant and revoke all grantable privileges and authorities
- Issue the TRACE command to start, stop, and display a trace

If the SEPARATE\_SECURITY system parameter is set to YES, no other authority can grant the ACCESSCTRL, System DBADM, and DATAACCESS authorities or the CREATE\_SECURE\_OBJECT privilege, not even SYSADM. For example, only SECADM, not SYSADM or DBADM, can activate or deactivate row or column access control for a table. 

### Related reference

 Protection panel 2: DSNTIPP1 (DB2 Installation and Migration)

## ACCESSCTRL

The ACCESSCTRL authority allows you to grant explicit privileges to authorization IDs or roles by issuing SQL GRANT statements. It enables you to grant privileges on all objects and resources, except the CREATE\_SECURE\_OBJECT privilege and the system DBADM, DATAACCESS, and ACCESSCTRL authorities.



**GUPI** With the ACCESSCTRL authority, you can use the BY clause to revoke explicitly granted privileges from authorization IDs or roles, except the CREATE\_SECURE\_OBJECT privilege and the system DBADM, DATAACCESS, and ACCESSCTRL authorities. In addition, you have implicit SELECT access on all catalog tables and implicit INSERT, DELETE, and UPDATE privileges on updatable catalog tables (except SYSIBM.SYSAUDITPOLICIES).

Only an authorization ID or a role with the SECADM authority can grant or revoke the ACCESSCTRL authority. Revoking the ACCESSCTRL authority does not revoke the privileges that it has already granted. **GUPI**

## DATAACCESS

The DATAACCESS authority allows you to access and update data in user tables, views, and materialized query tables in a DB2 subsystem. It also allows you to execute plans, packages, functions, and procedures.

Only an authorization ID or a role with the SECADM authority can grant or revoke the DATAACCESS authority. With the DATAACCESS authority, you have implicit SELECT access on all catalog tables and implicit INSERT, DELETE, and UPDATE privileges on updatable catalog tables (except SYSIBM.SYSAUDITPOLICIES).

## SQLADM

The SQLADM authority allows you to issue the SQL EXPLAIN statements, execute the PROFILE commands, run the RUNSTATS and MODIFY STATISTICS utilities on all user databases, and execute system-defined routines, such as stored procedures or functions, and any packages that are executed within the routines.

Only an authorization ID or a role with the SECADM or ACCESSCTRL authority can grant or revoke the SQLADM authority. With the SQLADM authority, you have implicit SELECT access on all the catalog tables and implicit INSERT, DELETE, and UPDATE privileges on updatable catalog tables (except SYSIBM.SYSAUDITPOLICIES).

## Common DB2 administrative authorities

Several DB2 administrative authorities provide the same functionality in DB2 for z/OS and DB2 for Linux, UNIX, and Windows. With these authorities, administrators who manage DB2 on multiple operating systems can manage their database environments in a consistent approach.

**GUPI** The following authorities provide the same administrative functionality in DB2 for z/OS and DB2 for Linux, UNIX, and Windows:

Table 17. Common DB2 administrative authorities


Administrative authority	Capabilities
System DBADM	<ul style="list-style-type: none"><li>• Manages resources in all databases</li><li>• Does not have access to data or the ability to grant and revoke privileges</li><li>• Executes system-defined routines (i.e., stored procedures or functions) and any package within the routines</li><li>• Has implicit SELECT access on all catalog tables</li></ul>



Table 17. Common DB2 administrative authorities (continued)

Administrative authority	Capabilities
SECADM	<ul style="list-style-type: none"> <li>Controls access to all database resources</li> <li>Manages security-related objects (i.e., roles, trusted contexts, row permissions, and column masks)</li> <li>Grants and revokes explicit privileges that are granted by itself and others</li> <li>Has implicit SELECT access on all catalog tables</li> </ul>
ACCESSCTRL	<ul style="list-style-type: none"> <li>Grants privileges on all but security-related objects and resources</li> <li>Revokes privileges on all but security-related objects and resources that are granted by itself or others</li> <li>Does not grant the system DBADM, DATAACCESS, or ACCESSCTRL authority</li> <li>Has implicit SELECT access on all catalog tables</li> </ul>
DATAACCESS	<ul style="list-style-type: none"> <li>Has the ability to access data in all user tables, views, and materialized query tables</li> <li>Has the ability to execute all plans, packages, functions, and procedures</li> <li>Has implicit SELECT access on all catalog tables</li> </ul>
SQLADM	<ul style="list-style-type: none"> <li>Issues EXPLAIN SQL statements and PROFILE commands</li> <li>Executes RUNSTATS and MODIFY STATISTICS utilities on all user databases</li> <li>Performs tasks that require EXPLAIN and MONITOR2 privileges</li> <li>Executes system defined routines (i.e., stored procedures or functions) and any package executed within the routines</li> <li>Has implicit SELECT access on all the catalog tables</li> </ul>

DB2 for z/OS provides both the system DBADM authority and the DBADM authority, with each having a set of privileges. The system DBADM authority allows you to manage objects in all databases across a DB2 subsystem, but doesn't give you access to the data in the databases. In addition, with the system DBADM authority, you can perform administrative tasks and issue commands for a DB2 subsystem, but you don't have the authority to execute objects or the ability to grant or revoke privileges.

Unlike the system DBADM authority, the DBADM authority allows you to manage objects in a specific database and gives you access to the data in that database. You also get the privileges of the DBCTRL and DBMAINT authorities over the same database. 

### Related reference

“System DBADM” on page 40

“SECADM” on page 41

“ACCESSCTRL” on page 41

“DATAACCESS” on page 42

“SQLADM” on page 42

## Utility authorities for DB2 catalog and directory

The DB2 catalog is in the DSNDB06 database. Authorities that are granted on DSNDB06 also cover database DSNDB01, which contains the DB2 directory.

An ID with the ACCESSCTRL or SECADM authority can control access to the catalog in the following ways:

- By granting privileges or authorities on that database or on its tables or views
- By binding plans or packages that access the catalog

An ID with the ACCESSCTRL or SECADM authority can control access to the directory by granting privileges to run utilities on DSNDB06, but that ID cannot grant privileges on DSNDB01 directly.

The following table shows the utilities IDs with different authorities that can run on the DSNDB01 and DSNDB06 databases. Do not run REPAIR DBD against DSNDB01 and DSNDB06 because they are system databases; you will receive a system restriction violation message if you do. Also, you can use the LOAD utility to add lines to SYSIBM.SYSSTRINGS, but you cannot run it on other DSNDB01 or DSNDB06 tables.

Table 18. Utility privileges on the DB2 catalog and directory

Utilities	Installation SYSOPR, SYSCTRL, SYSADM, Installation SYSADM	DBCTRL, DBADM on DSNDB06	DBMAINT on DSNDB06	System DBADM	DATAACCESS	SQLADM
LOAD	No	No	No	No	No	No
REPAIR DBD	No	No	No	No	No	No
CHECK DATA	Yes	No	No	No	Yes	No
CHECK LOB	Yes	No	No	Yes	No	No
REORG TABLESPACE	Yes	No	No	No	Yes	No
STOSPACE	Yes	No	No	No	No	No
REBUILD INDEX	Yes	Yes	No	Yes	Yes	No
RECOVER	Yes	Yes	No	Yes	Yes	No
REORG INDEX	Yes	Yes	No	No	Yes	No
REPAIR	Yes	Yes	No	No	Yes	No
REPORT	Yes	Yes	No	Yes	Yes	No
CHECK INDEX	Yes	Yes	Yes	Yes	No	No
COPY	Yes	Yes	Yes	Yes	No	No

Table 18. Utility privileges on the DB2 catalog and directory (continued)

Utilities	Installation SYSOPR, SYSCTRL, SYSADM, Installation SYSADM	DBCTRL, DBADM on DSNDB06	DBMAINT on DSNDB06	System DBADM	DATAACCESS	SQLADM
MERGECOPY	Yes	Yes	Yes	Yes	No	No
MODIFY	Yes	Yes	Yes	Yes	No	No
QUIESCE	Yes	Yes	Yes	Yes	No	No
RUNSTATS	Yes	Yes	Yes	Yes	No	Yes

#### Related concepts

“Explicit privileges” on page 24

“Privileges by authorization ID and authority”

#### Related reference

“Implicit privileges through object ownership” on page 30

“Administrative authorities” on page 31

## Privileges by authorization ID and authority

When a process gains access to DB2, it has a primary authorization ID, one or more secondary authorization IDs, an SQL ID, and perhaps a specific role if it runs in a trusted context. To be able to perform certain actions, an authorization ID or role must hold the required privileges. To perform other actions, a set of IDs or roles must hold the required privileges.

For better performance, consider limiting the number of secondary IDs in your catalog table. A process can have up to 1012 secondary IDs. The more secondary IDs that must be checked, the longer the check takes. Also, make sure that the role and the current SQL ID have the necessary privileges for dynamic SQL statements. Because the role and the current SQL ID are checked first, the operation is fastest if they have all the necessary privileges.

#### Related concepts

“Explicit privileges” on page 24

#### Related reference

“Implicit privileges through object ownership” on page 30

“Administrative authorities” on page 31

“Utility authorities for DB2 catalog and directory” on page 44

## Privileges required for common job roles and tasks

The labels of the administrative authorities often suggest the job roles and responsibilities of the users who are empowered with the authorities.

**GUPI** For example, you might expect a system administrator to have the SYSADM authority. However, some organizations do not divide job responsibilities in the same way. The following table lists some of common job roles, the tasks that usually accompany them, and the DB2 authorities or privileges that are needed to perform those tasks.

Table 19. Required privileges for common jobs and tasks

Job title	Tasks	Required privileges
System operator	Issues commands to: <ul style="list-style-type: none"> <li>Start and stop DB2</li> <li>Control traces</li> <li>Display databases and threads</li> <li>Recover indoubt threads</li> <li>Start, stop, and display routines</li> </ul>	SYSOPR authority
System administrator	Performs emergency backup, with access to all data.	SYSADM authority
Security administrator	Authorizes other users, for some or all levels below.	<ul style="list-style-type: none"> <li>SYSCTRL authority (if SEPARATE_SECURITY is set to NO)</li> <li>SECADM authority</li> <li>ACCESSCTRL authority</li> </ul>
Database administrator	Designs, creates, loads, reorganizes, and monitors databases, tables, and other objects in the database.	<ul style="list-style-type: none"> <li>DBADM authority on a database. The DBADM authority on DSNDB04 allows you access to objects in all implicitly created databases.</li> <li>Use of storage groups and buffer pools</li> </ul>
Database administrator	<ul style="list-style-type: none"> <li>Designs and creates databases, tables, and other objects</li> <li>Administers all databases in the subsystem</li> </ul>	System DBADM authority
Database administrator	Manages data and executes plans and packages in a DB2 subsystem	DATAACCESS authority
Database administrator	Manages access to data in a DB2 subsystem	ACCESSCTRL authority
System programmer	<ul style="list-style-type: none"> <li>Installs a DB2 subsystem.</li> <li>Recovers the DB2 catalog.</li> <li>Repairs data.</li> </ul>	Installation SYSADM, which is assigned when DB2 is installed. (Consider securing the password for an ID with this authority so that the authority is available only when needed.)
Application programmer	<ul style="list-style-type: none"> <li>Develops and tests DB2 application programs.</li> <li>Creates tables of test data.</li> </ul>	<ul style="list-style-type: none"> <li>BIND on existing plans or packages, or BINDADD</li> <li>CREATE IN on some collections</li> <li>Privileges on some objects</li> <li>CREATETAB on some database, with a default table space provided</li> <li>CREATETAB on DSNDB04. It enables you to create tables in DSNDB04 and all implicitly created databases</li> <li>Privileges on some objects with the SQLADM authority</li> </ul>
Production binder	Binds, rebinds, and frees application packages and plans	A ROLE, secondary ID, or RACF group of which the binder has BINDADD, CREATE IN on collections privileges required by application packages and plans
Package administrator	Manages collections and the packages in them, and delegates the responsibilities.	PACKADM authority
User analyst	Defines the data requirements for an application program, by examining the DB2 catalog.	<ul style="list-style-type: none"> <li>SELECT on the SYSTABLES, SYSCOLUMNS, and SYSVIEWS catalog tables</li> <li>CREATETMTAB system privilege to create temporary tables</li> </ul>
Program end user	Executes an application program.	EXECUTE for the application plan

Table 19. Required privileges for common jobs and tasks (continued)

Job title	Tasks	Required privileges
Information center consultant	<ul style="list-style-type: none"> <li>Defines the data requirements for a query user.</li> <li>Provides the data by creating tables and views, loading tables, and granting access.</li> </ul>	<ul style="list-style-type: none"> <li>DBADM authority over some databases</li> <li>SELECT on the SYSTABLES, SYSCOLUMNS, and SYSVIEWS catalog tables</li> </ul>
Query user	<ul style="list-style-type: none"> <li>Issues SQL statements to retrieve, add, or change data.</li> <li>Saves results as tables or in global temporary tables.</li> </ul>	<ul style="list-style-type: none"> <li>EXPLAIN privilege on some tables and views</li> <li>SELECT, INSERT, UPDATE, DELETE on some tables and views</li> <li>CREATETAB, to create tables in other than the default database</li> <li>CREATETAB, to create tables in the implicitly created database</li> <li>CREATETMTAB system privilege to create temporary tables</li> <li>SELECT on SYSTABLES, SYSCOLUMNS, or views thereof. QMF™ provides the views.</li> </ul>

#### GUPI

### Checking access authorization for data definition statements

DB2 checks for the necessary authorization privileges and authorities when you use data definition statements on certain DB2 objects.

At both bind and run time, DB2 determines whether the authorization ID that you are using has the necessary privileges to access the following objects:

- Alias
- Table
- Explicitly created auxiliary table
- Explicitly created table space
- Explicitly created index
- Storage group
- Database

At run time, DB2 determines whether the authorization ID that you are using has the necessary privileges to access the following objects:

- Buffer pool that is involved with an implicitly created table space
- Buffer pool and storage group that are involved with an implicitly created auxiliary index and LOB table space
- Buffer pool and storage group that are involved with implicitly created XML indexes and XML table space
- Trigger
- Function
- Procedure
- Sequence
- View
- Trusted context

- JAR
- Role
- Distinct type
- Table, buffer pool, and storage group for an implicitly created unique key index, primary key index, or ROWID index.

## Privileges required for handling plans and packages

An ID, or a role that runs in a trusted context, needs specific privileges to perform actions on plans and packages.

### GUPI

The following table lists the IDs and describes the privileges that they need for performing each type of plan or package operation. A user-defined function, stored procedure, or trigger package does not need to be included in a package list. A trigger package cannot be deleted by FREE PACKAGE or DROP PACKAGE. The DROP TRIGGER statement must be used to delete the trigger package.

Table 20. Required privileges for basic operations on plans and packages

Operation	ID or role	Required privileges
Execute a plan	Primary ID, any secondary ID, or role	Any of the following privileges: <ul style="list-style-type: none"> <li>• Ownership of the plan</li> <li>• EXECUTE privilege for the plan</li> <li>• DATAACCESS authority</li> <li>• SYSADM authority</li> </ul>
Bind embedded SQL statements, for any bind operation	Package owner	Any of the following privileges: <ul style="list-style-type: none"> <li>• Applicable privileges required by the statements</li> <li>• Authorities that include the privileges</li> <li>• Ownership that implicitly includes the privileges</li> </ul> <p>Object names include the value of QUALIFIER, where it applies.</p>
BIND EXPLAIN without generating a package	Plan or package owner	Any of the following privileges: <ul style="list-style-type: none"> <li>• Ownership of the plan or package</li> <li>• BIND</li> <li>• BINDAGENT</li> <li>• EXPLAIN privilege</li> <li>• PACKADM</li> <li>• SQLADM</li> <li>• System DBADM authority</li> <li>• SYSCTRL</li> <li>• SYSADM</li> </ul>
Include package in PKLIST <sup>1</sup>	Plan owner	Any of the following privileges: <ul style="list-style-type: none"> <li>• Ownership of the package</li> <li>• EXECUTE privilege for the package</li> <li>• PACKADM authority over the package collection</li> <li>• SYSADM authority</li> </ul>
BIND a new plan using the default owner or primary authorization ID	Primary ID or role	Any of the following privileges: <ul style="list-style-type: none"> <li>• BINDADD privilege</li> <li>• System DBADM authority</li> <li>• SYSCTRL authority</li> <li>• SYSADM authority</li> </ul>

Table 20. Required privileges for basic operations on plans and packages (continued)

Operation	ID or role	Required privileges
BIND a new package using the default owner or primary authorization ID	Primary ID or role	<p>If the value of the field BIND NEW PACKAGE on installation panel DSNTIPP is BIND, any of the following privileges:</p> <ul style="list-style-type: none"> <li>• BIND privilege and CREATE IN privilege for the collection</li> <li>• PACKADM authority for the collection</li> <li>• System DBADM authority</li> <li>• SYSADM or SYSCTRL authority</li> </ul> <p>If BIND NEW PACKAGE is BINDADD, any of the following privileges:</p> <ul style="list-style-type: none"> <li>• BINDADD privilege and either the CREATE IN or PACKADM privilege for the collection</li> <li>• System DBADM authority</li> <li>• SYSADM or SYSCTRL authority</li> </ul>
BIND REPLACE or REBIND for a plan or package using the default owner or primary authorization ID	Primary ID, any secondary ID, or role	<p>Any of the following privileges:</p> <ul style="list-style-type: none"> <li>• Ownership of the plan or package</li> <li>• BIND privilege for the plan or package</li> <li>• BINDAGENT from the plan or package owner</li> <li>• PACKADM authority for the collection (for a package only)</li> <li>• System DBADM authority</li> <li>• SYSADM or SYSCTRL authority.</li> </ul>
BIND a new version of a package, with default owner	Primary ID or role	<p>If BIND NEW PACKAGE is BIND, any of the following privileges:</p> <ul style="list-style-type: none"> <li>• BIND privilege on the package or collection</li> <li>• BINDADD privilege and CREATE IN privilege for the collection</li> <li>• PACKADM authority for the collection</li> <li>• System DBADM authority</li> <li>• SYSADM or SYSCTRL authority</li> </ul> <p>If BIND NEW PACKAGE is BINDADD, any of the following:</p> <ul style="list-style-type: none"> <li>• BINDADD privilege and either the CREATE IN or PACKADM privilege for the collection</li> <li>• System DBADM authority</li> <li>• SYSADM or SYSCTRL authority</li> </ul>
FREE or DROP a package <sup>2</sup>	Primary ID, any secondary ID, or role	<p>Any of the following privileges:</p> <ul style="list-style-type: none"> <li>• Ownership of the package</li> <li>• BINDAGENT from the package owner</li> <li>• PACKADM authority for the collection</li> <li>• System DBADM authority</li> <li>• SYSADM or SYSCTRL authority</li> </ul>
COPY a package	Primary ID, any secondary ID, or role	<p>Any of the following:</p> <ul style="list-style-type: none"> <li>• Ownership of the package</li> <li>• COPY privilege for the package</li> <li>• BINDAGENT from the package owner</li> <li>• PACKADM authority for the collection</li> <li>• System DBADM authority</li> <li>• SYSADM or SYSCTRL authority</li> </ul>

Table 20. Required privileges for basic operations on plans and packages (continued)

Operation	ID or role	Required privileges
FREE a plan	Primary ID, any secondary ID, or role	Any of the following privileges: <ul style="list-style-type: none"> <li>• Ownership of the plan</li> <li>• BIND privilege for the plan</li> <li>• BINDAGENT from the plan owner</li> <li>• System DBADM authority</li> <li>• SYSADM or SYSCTRL authority</li> </ul>
Name a new OWNER other than the primary authorization ID for any bind operation	Primary ID, any secondary ID, or role	Any of the following privileges: <ul style="list-style-type: none"> <li>• New owner is the primary or any secondary ID</li> <li>• BINDAGENT from the new owner</li> <li>• System DBADM authority (if SEPARATE_SECURITY is set to NO)</li> <li>• SYSADM or SYSCTRL authority (if SEPARATE_SECURITY is set to NO)</li> </ul>

#### GUPI

### Privileges required for using dynamic SQL statements

An ID needs specific privileges to issue dynamic SQL statements.

#### GUPI

The following table lists the IDs and describes the privileges that they need for issuing each type of SQL statement:

Table 21. Required privileges for basic operations on dynamic SQL statements

Operation	ID or role	Required privileges
GRANT	Current SQL ID or role	Any of the following privileges: <ul style="list-style-type: none"> <li>• The applicable privilege with the grant option</li> <li>• An authority that includes the privilege, with the grant option (not needed for SYSADM or SYSCTRL)</li> <li>• Ownership that implicitly includes the privilege</li> </ul>
REVOKE	Current SQL ID or role	Must either have granted the privilege that is being revoked, or hold SYSCTRL or SYSADM authority.
CREATE, for unqualified object name	Current SQL ID or role	Applicable table, database, or schema privilege
Qualify name of object created	ID or role named as owner	Applicable table or database privilege. The qualifier can be any ID at all and does not need to have any privilege if the current SQL ID or the role (if in a trusted context with the ROLE AS OBJECT OWNER AND QUALIFIER clause specified) has the SYSADM, system DBADM, or SYSCTRL authority (wherever applicable) or the DBADM or DBCTRL authority for the database (wherever applicable).



Table 21. Required privileges for basic operations on dynamic SQL statements (continued)

Operation	ID or role	Required privileges
Other dynamic SQL if DYNAMICRULES uses run behavior	All primary IDs, role, secondary IDs, and the current SQL ID together	As required by the statement. Unqualified object names are qualified by the value of the special register CURRENT SQLID.
Other dynamic SQL if DYNAMICRULES uses bind behavior	Plan or package owner	As required by the statement. DYNAMICRULES behavior determines how unqualified object names are qualified.
Other dynamic SQL if DYNAMICRULES uses define behavior	Function or procedure owner	As required by the statement. DYNAMICRULES behavior determines how unqualified object names are qualified.
Other dynamic SQL if DYNAMICRULES uses invoke behavior	ID of the SQL statement that invoked the function or procedure or role	As required by the statement. DYNAMICRULES behavior determines how unqualified object names are qualified.



## Managing administrative authorities

DB2 provides a range of auditable administrative authorities that help you control access to sensitive business data. The granularity and flexibility in DB2 administrative authority help you achieve adequate separation of duties and responsibilities and prevent a single user from possessing unlimited privileges.

Depending on the setting of the SEPARATE\_SECURITY system parameter, you can separate DB2 security administration from system administration. You can set the parameter by using the SEPARATE SECURITY field on panel DSNTIPP1 during installation or migration.

If you set SEPARATE\_SECURITY to YES, the SYSADM authority can no longer manage security-related objects (i.e., roles, trusted contexts, row permissions, and column masks) or have the ability to grant or revoke privileges that are granted by others. The SYSCTRL authority can no longer manage roles or grant or revoke privileges that are granted by others, either. Instead, the SECADM authority will manage all security-related objects. The SECADM and ACCESSCTRL authorities control access to all databases even though they cannot access any user data in the databases.

In addition, the SYSADM authority can only set CURRENT SQLID to its primary or one of its secondary authorization IDs. The SYSADM, SYSCTRL, and system DBADM authorities can only set BIND OWNER to the primary or one of the secondary authorization IDs of the binder. Finally, the SYSADM authority will not have implicit insert, update, delete access to the SYSIBM.SYSAUDITPOLICIES table.

If you set SEPARATE\_SECURITY to NO (which is the default), the SYSADM authority retains all the existing privileges and responsibilities and gets implicit privileges of the SECADM authority. In other words, the SYSADM authority

continues to be the security administrator and manage all security-related objects, perform grants, and revoke privileges that are granted by others. In addition, it gets implicit insert, update, delete access on the SYSIBM. SYSAUDITPOLICIES table and is able to set CURRENT SQLID and BIND OWNER to any value.

Setting SEPARATE\_SECURITY to NO also allows the SYSCTRL authority to get most of the implicit privileges of the ACCESSCTRL authority. SYSCTRL can manage roles, perform certain grants, revoke privileges that are granted by others, and set BIND OWNER to any value.

The installation SYSADM authority is not affected by the setting of the SEPARATE\_SECURITY parameter. Installation SYSADM can manage security-related objects, grant and revoke authorities or privileges, and set CURRENT SQLID and BIND OWNER to any value regardless of the setting of the SEPARATE\_SECURITY parameter.

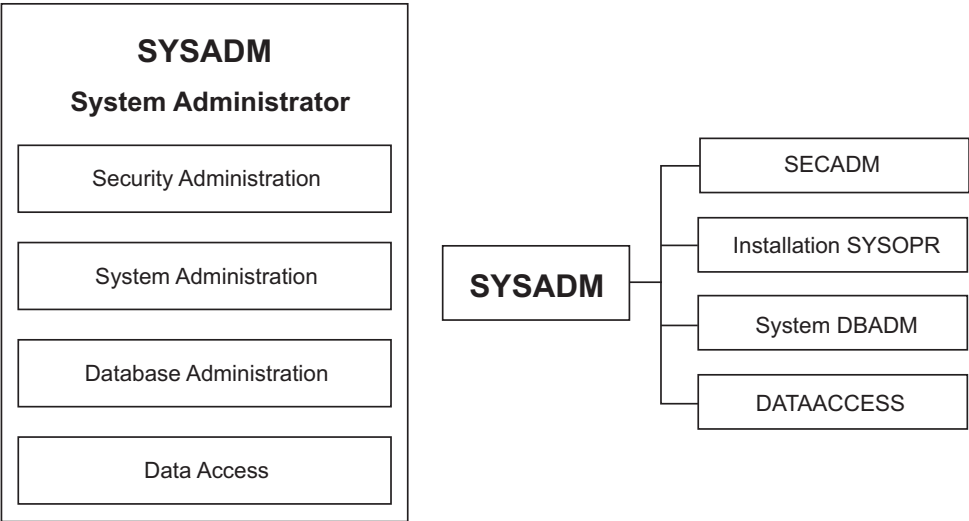
### Separating the SYSADM authority

Granularity and flexibility in DB2 administrative authority allows you to separate security administration, database administration, and data access control from system administration. Separating the SYSADM authority (a combination of security and system administration) can help you simplify your system administration and strengthen the security administration of your business data.

**GUIP** Choose the system and security administration model that best meets the security needs of your business:

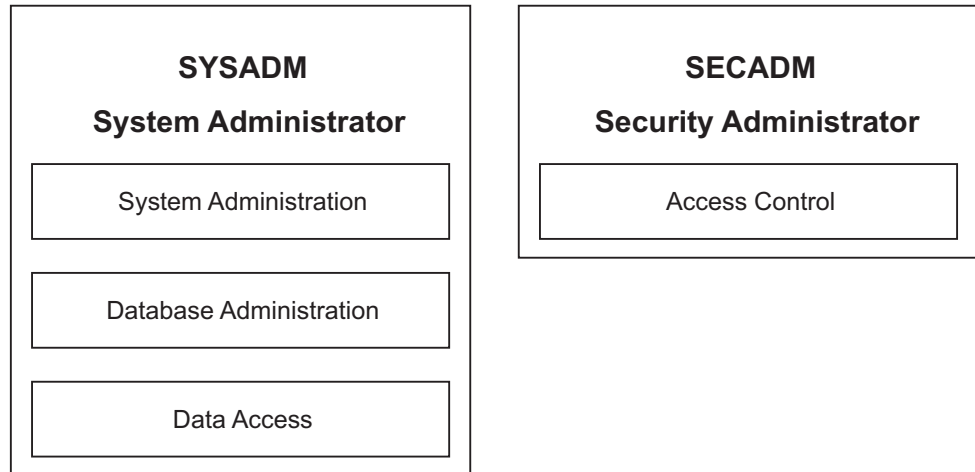
- Maintain the existing system administration model in which the SYSADM authority continues to be able to perform security administration

You must first set the SEPARATE\_SECURITY system parameter on panel DSNTIPP1 to NO (which is the default) during installation or migration. As shown below, this setting allows the system administrator to continue to be the security administrator and the SYSADM authority to get implicit privileges of the SECADM authority. A system administrator can therefore manage all security-related objects, perform grants, and revoke privileges that are granted by others.



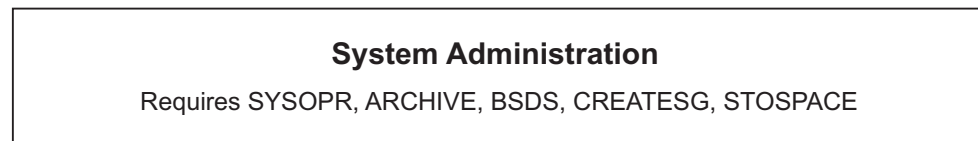
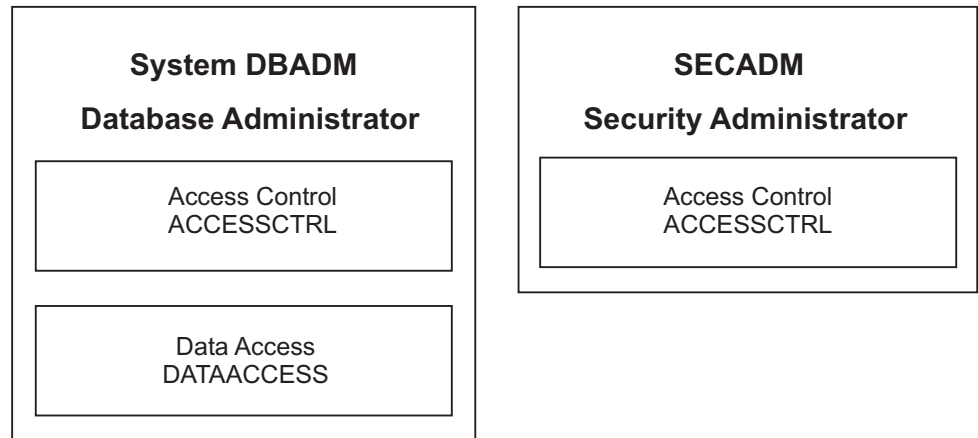
Setting SEPARATE\_SECURITY to NO also allows the SYSCTRL authority to get implicit privileges of the ACCESSCTRL authority. SYSCTRL can manage roles, perform grants, and revoke privileges that are granted by others.

- Separate security administration from system administration (SYSADM)  
You must first set the SEPARATE\_SECURITY system parameter on panel DSNTIPP1 to YES during installation or migration. As shown below, this setting separates the security administration from the SYSADM authority. A system administrator can no longer manage access control, audit policies, or security-related objects, including roles and trusted contexts. The SYSCTRL authority can no longer manage roles. Neither the SYSADM authority nor the SYSCTRL authority can grant or revoke privileges that are granted by others.

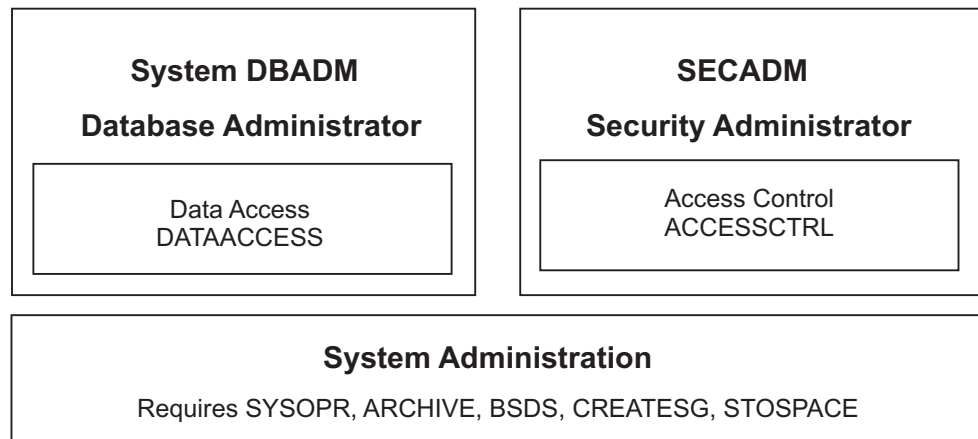


In addition to setting the SEPARATE\_SECURITY system parameter, you also need to set one of the system SECADM parameters to an authorization ID or a role during installation that will perform security administration. To ensure complete separation of system and security administration, do not set the SECADM system parameter to a SYSADM ID. Instead, set SECADM to a SECADM ID and installation SYSADM to an installation SYSADM ID.

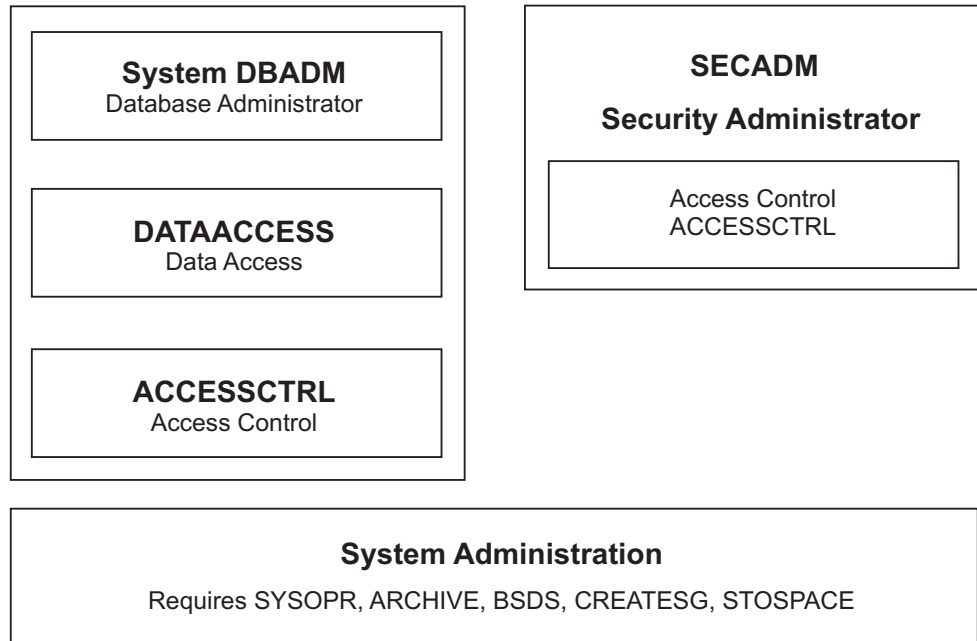
- Separate system database administration with the data access authority and the access control authority from system and security administration.  
DB2 provides both the system DBADM authority and the DBADM authority, with each having a different set of privileges. The system DBADM authority allows you to manage objects in all databases across a DB2 subsystem, but doesn't give you access to the data in the databases. In addition, with the system DBADM authority, you can perform administrative tasks and issue commands for a DB2 subsystem, but you don't have the authority to execute objects or the ability to grant or revoke privileges.  
Unlike the system DBADM authority, the DBADM authority allows you to manage objects in a specific database and gives you access to the data in that database. You also get the privileges of the DBCTRL and DBMAINT authorities over the same database.  
If you want the system database administrators to have access to data and the ability to grant and revoke privileges, you can grant them the system DBADM, DATAACCESS, and ACCESSCTRL authorities, as shown below. By default, the DATAACCESS and ACCESSCTRL authorities are granted when the system DBADM authority is granted.



If you want the system database administrators to have access to data, but not the ability to grant or revoke privileges, you can grant them the system DBADM and DATAACCESS authorities, but not the ACCESSCTRL authority, as shown below. You can also grant system database administrators the SYSOPR authority and the privileges to perform ARCHIVE, BSDS, CREATESG, STOSPACE, or other system-related tasks.



- Separate system database administration from the data access authority, the access control authority, security administration, and system administration.



If you want the system database administrators to manage database objects, but have no access to data or the ability to grant and revoke privileges, you can grant them the system DBADM authority, but not the SYSADM, DATAACCESS, or ACCESSCTRL authority. **GUPI**

#### Related reference

“System DBADM” on page 40  
 “SECADM” on page 41  
 “ACCESSCTRL” on page 41  
 “DATAACCESS” on page 42  
 “SQLADM” on page 42

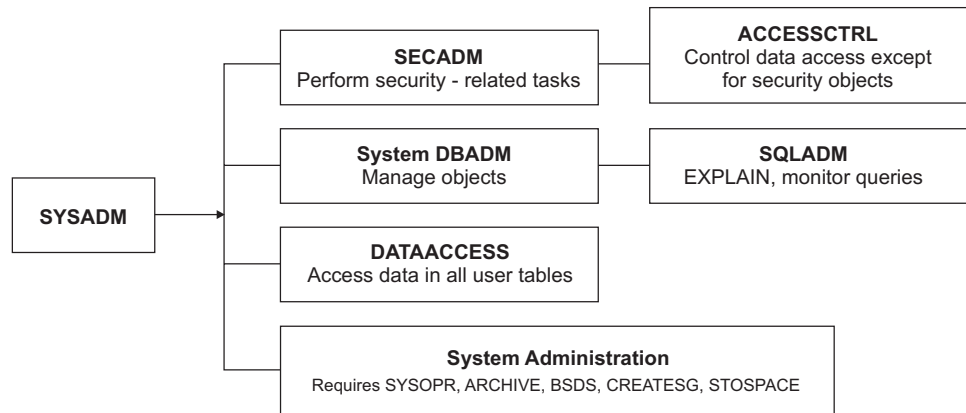
## Migrating the SYSADM authority

To take advantage of the granularity of DB2 administrative authority and simplify your system database administration, you can separate the privileges of the SYSADM authority and migrate them to other administrative authorities based on the security needs of your business. This will allow you to eliminate or minimize the need for granting the SYSADM authority.

If you decide to separate the SYSADM authority into the SECADM and other administrative authorities, consider setting the SEPARATE\_SECURITY system parameter on panel DSNTIPP1 to YES during installation or migration. This setting enables you to achieve complete separation of administrative duties.

To migrate the SYSADM authority that is currently assigned to authorization IDs or roles:

1. In your security policies, identify the administration model that you will use for separating the SYSADM authority and define the criteria for assigning specific administrative authorities to specific authorization IDs or roles.  
 Suppose that you choose the following model to separate the current SYSADM authority into the system DBADM, SECADM, DATAACCESS, ACCESSCTRL, and SQLADM authorities:



You can define the following set of criteria for granting administrative authorities:

- The system DBADM authority can be granted to database administrators who need to manage objects
- The DATAACCESS authority can be granted to database administrators who need to access data
- The ACCESSCTRL authority can be granted to database administrators who need to control access to DB2 subsystems
- The SECADM authority can be assigned (during installation) to security administrators who perform security administration and manage access control
- The SQLADM authority can be assigned to performance analysts who are responsible for analyzing the performance of DB2 subsystems
- The EXPLAIN privilege can be granted to application architects who need to explain SQL statements or collect metadata information about the statements
- The SYSOPR authority and the ARCHIVE, BSDS, CREATESG, and STOSPACE privileges can be granted to system administrators for performing system administrative tasks.

2. Perform a query to list all the users and roles that are currently granted the SYSADM authority.

The SYSADM authority can be granted to authorization IDs or roles. You can query the catalog and find out the users and roles who are currently granted the SYSADM authority.

Suppose that your query returns a list of the following six users, user groups, or roles that are assigned the SYSADM authority:

- John (Security administrator)
- Sally (Application Architect)
- Bob (Performance Analyst)
- ApplProgrammer\_role (Application Programmer role)
- SysAdmin\_Role (System administrator role)
- DBAdmGrp (database administrator group).

3. Divide the responsibilities of the SYSADM authority and grant to different IDs or roles based on your security policies, as shown below:

- John is granted the SECADM authority to perform security-related administration tasks and control access to DB2.

- Sally is granted the DATAACCESS authority because she requires DML privileges on tables during application development, but she does not need access control or database administration.
  - Bob is granted the SQLADM authority who analyzes the performance of DB2 subsystems, but does not need access to data.
  - ApplProg\_role is granted the EXPLAIN privilege because all application programmers need to explain SQL statements and collect metadata information in trusted context definitions.
  - DBAdmGrp is granted the system DBADM authority for managing and maintaining objects. Since database administrators belong to the DBAdmGrp RACF group, they should not be able to access data or grant and revoke privileges.
  - SysAdmin\_role is granted the SYSOPR authority and the ARCHIVE, BSDS, CREATESG, and STOSPACE privileges to perform system administrative tasks.
4. Revoke the SYSADM authority from all current IDs or roles.  
Once the authorities are granted, you can revoke the SYSADM authority from John, Sally, Bob, ApplProgrammer\_role and DBAdmGrp. Revoking the SYSADM authority causes the revoking of dependent privileges, by default. If you want to leave the grants that they had made, you can issue the REVOKE statement with the NOT INCLUDING DEPENDENT PRIVILEGES clause, assuming the REVOKE\_DEP\_PRIVILEGES system parameter is set to SQLSTMT.
  5. Once the SYSADM authority is revoked, set the SEPARATE\_SECURITY system parameter to YES on panel DSNTIPP1. With the installation SYSADM authority, you can perform an online change of the SEPARATE\_SECURITY system parameter and set it to YES. This further ensures that SYSADM is separated into SECADM and other authorities.

#### **Related reference**

“System DBADM” on page 40  
 “SECADM” on page 41  
 “ACCESSCTRL” on page 41  
 “DATAACCESS” on page 42  
 “SQLADM” on page 42

## **Creating roles or trusted contexts with the SECADM authority**

If you separate security administration from system and database administration, you need to have the SECADM authority to manage security-related objects in DB2 and control access to all database objects and resources in a subsystem.

To separate security administration from system administration, you must set the SEPARATE\_SECURITY system parameter on panel DSNTIPP1 to YES during installation or migration.

With the separation of security administration from system administration, the SYSADM authority can no longer define roles or trusted contexts or manage any other security-related objects; the SECADM authority is, instead, responsible for performing security administrative tasks, including creating roles and trusted contexts, activating row and column access control, and granting security-related authorities and privileges on objects.

To create roles or trusted contexts with the SECADM authority:



- Issue the following CREATE ROLE statement to create CTXROLE by using an authorization ID or role that is given the SECADM authority.

If SEPARATE\_SECURITY is set to YES, the SECADM authority is required to create roles and trusted contexts. **GUIP**

```
CREATE ROLE CTXROLE;
```

#### **GUIP**

DB2 checks to make sure that you have the required privilege to create roles and, upon successful verification, allows the creation of role CTXROLE.

- Issue the following CREATE TRUSTED CONTEXT statement to create CTX1 and associate CTXROLE with CTX1:

#### **GUIP**

```
CREATE TRUSTED CONTEXT CTX1
BASED UPON CONNECTION USING SYSTEM AUTHID USER1
DEFAULT ROLE CTXROLE
ATTRIBUTES (ADDRESS '9.67.40.219')
WITH USE FOR USER2, USER3
ENABLE;
```

#### **GUIP**

DB2 checks to make sure that you have the required privilege to create trusted contexts and, upon successful verification, allows the creation of trusted context CTX1.

#### **Related reference**

“SECADM” on page 41

## **Altering tables with the system DBADM authority**

The system DBADM authority separates object management from data access and access control. It allows object management without requiring the ownership of the object in a DB2 subsystem.

Suppose that you are a database administrator DB2ADMIN1 and need to alter TABLE1, but do not have any table privileges on the table. You must first be granted the system DBADM authority before you can alter the table.

To alter tables with the system DBADM authority:

1. Obtain the system DBADM authority from a security administrator

An authorization ID or role with the SECADM authority can grant you the system DBADM authority by issuing the following statement: **GUIP**

```
GRANT DBADM WITHOUT DATAACCESS WITHOUT ACCESSCTRL ON SYSTEM TO DB2ADMIN1;
```

#### **GUIP**

DB2 inserts a row in SYSIBM.SYSUSERAUTH with the column SDBADMAUTH set to 'Y', where column GRANTEE is set to DB2ADMIN1.

2. With the system DBADM authority, issue the ALTER TABLE statement to alter table TABLE1.

DB2 checks to make sure that you have the required privilege set, including the ALTER TABLE privilege that is allowed by the system DBADM authority. The table is altered successfully.



## Related reference

“System DBADM” on page 40

## Accessing data with the DATAACCESS authority

A database administrator must have the DATAACCESS authority to access data in all user tables, views, and materialized query tables in a DB2 subsystem.

Suppose that you are a database administrator DB2ADMIN1 and need access to data in TABLE1. You must first be granted the DATAACCESS authority.

To access data with the DATAACCESS authority:

1. Obtain the DATAACCESS authority from a security administrator. The SECADM (an authorization ID or role) can grant you the DATAACCESS authority by issuing the following statement:

**GUI**

```
GRANT DATAACCESS ON SYSTEM TO DB2ADMIN1;
```

**GUI**

DB2 inserts a row in SYSIBM.SYSUSERAUTH with the new column DATAACCESSAUTH set to 'Y', where column GRANTEE is set DB2ADMIN1.

2. After obtaining the DATAACCESS authority, issue an SQL SELECT statement to select from table TABLE1. DB2 checks to make sure that you have the required privilege set, including the SELECT privilege that is granted by the DATAACCESS authority. The SELECT statement completes successfully.

## Related reference

“DATAACCESS” on page 42

## Granting and revoking privileges with the ACCESSCTRL authority

If you separate database administration from system and security administration, a database administrator must have the ACCESSCTRL or SECADM authority to grant or revoke user privileges in a DB2 subsystem.

**GUI**

The ACCESSCTRL authority allows you to grant and revoke (BY clause) privileges on all resources in a DB2 subsystem. However, it cannot grant the CREATE\_SECURE\_OBJECT privilege or the system DBADM, DATAACCESS, and ACCESSCTRL authorities.

If you are a database administrator DB2ADMIN1 and need to grant application developer APPDEV1 load privileges on DBTEMP1, you must first have the ACCESSCTRL authority for yourself.

To grant or revoke privileges with the ACCESSCTRL authority:

1. Obtain the ACCESSCTRL authority from a security administrator. The SECADM (an authorization ID or role) can grant you the ACCESSCTRL authority by issuing the following statement:


```
GRANT ACCESSCTRL ON SYSTEM TO DB2ADMIN1;
```

DB2 inserts a row in SYSIBM.SYSUSERAUTH with the new column ACCESSCTRLAUTH set to 'Y', where column GRANTEE is set to DB2ADMIN1.

You can specify WITH GRANT OPTION when you issue the GRANT statement, but the option is ignored when the authority is ACCESSCTRL, DBADM, or DATAACCESS.

2. After obtaining the ACCESSCTRL authority, grant APPDEV1 load privileges on DBTEMP1 by issuing the following GRANT statement:

```
GRANT LOAD ON DATABASE DBTEMP1 TO APPDEV1;
```

DB2 checks to make sure that you have the required privilege set, including the GRANT privilege that is allowed by the ACCESSCTRL authority. The GRANT statement completes successfully. 

#### Related reference

“ACCESSCTRL” on page 41

---

## Managing explicit privileges

You can use the SQL GRANT and REVOKE statements to grant and remove privileges if you enable authorization checking during DB2 installation. You can grant to or revoke privileges from authorization IDs or roles if they run in a trusted context. You can revoke only privileges that are explicitly granted.

You can grant privileges in the following ways:

- Grant a specific privilege on one object in a single statement
- Grant a list of privileges
- Grant privileges on a list of objects
- Grant ALL, for all the privileges of accessing a single table, or for all privileges that are associated with a specific package

If you grant privileges on a procedure or a package, all versions of that procedure or package have those privileges. DB2 ignores duplicate grants and keeps only one record of a grant in the catalog. The suppression of duplicate records applies not only to explicit grants, but also to the implicit grants of privileges that are made when a package is created.

For example, suppose that Susan grants the SELECT privilege on the EMP table to Ray. Then suppose that Susan grants the same privilege to Ray again, without revoking the first grant. When Susan issues the second grant, DB2 ignores it and maintains the record of the first grant in the catalog.

Database privileges that are granted on DSNDB04 apply to all implicitly created databases. For example, if you have the DBADM authority on DSNDB04, you can select data from any table in any implicitly created database. If you have the STOPDB privilege on DSNDB04, you can stop any implicitly created database. However, you cannot grant the same authorities or privileges to others on any implicitly created database.

#### Related tasks

“Managing implicit privileges” on page 77

## Granting privileges to a role

You can grant privileges to a role by using the GRANT statement. You can associate primary authorization IDs with a role in the definition of the trusted context and then use the GRANT statement with the ROLE option to grant privileges to the role.

You can improve access control by granting privileges to roles. When you grant certain privileges to a role, you make those privileges available to all users that are associated with the role in the specific trusted context.

You can also simplify the administration of granting privileges by using roles rather than individual authorization IDs. To make a role a grantor, you need to specify the `ROLE AS OBJECT OWNER` clause when you define the trusted context. For a static `GRANT` statement, the grantor is the role that owns the plan or package. For a dynamic `GRANT` statement, the role for the primary authorization ID that executes the `GRANT` statement becomes the grantor.

## Granting privileges to the PUBLIC ID

You can grant to the `PUBLIC` ID privileges or authorities other than `CREATE_SECURE_OBJECT`, `system DBADM`, `DATAACCESS`, or `ACCESSCTRL`.

When you grant privileges to `PUBLIC`, the privileges become available to all IDs at the local DB2® site, including the owner IDs of packages that are bound from a remote location. Public access is generally not a good practice when it comes to the protection of sensitive business data and critical system resources. Many compliance requirements prohibit public access to any system components. For example, the Payment Card Industry (PCI) Data Security Standard Requirements and Security Assessment Procedures restrict the use of `PUBLIC`.

When you grant any privilege to `PUBLIC`, DB2 catalog tables record the grantee of the privilege as `PUBLIC`. DB2 also grants the following implicit table privileges to `PUBLIC` for declared temporary tables:

- All table privileges on the tables and the authority to drop the tables
- The `CREATETAB` privilege to define temporary tables in the work file database
- The `USE` privilege to use the table spaces in the work file database

You do not need any additional privileges to access the work file database and the temporary tables that are in it. You cannot grant or revoke table privileges for temporary tables. The DB2 catalog does not record these implicit privileges for declared temporary tables.

Because `PUBLIC` is a special identifier that is used by DB2 internally, you should not use `PUBLIC` as a primary ID or secondary ID. When a privilege is revoked from `PUBLIC`, authorization IDs to which the privilege was specifically granted retain the privilege.

However, when an ID uses `PUBLIC` privileges to perform actions, the actions and the resulting objects depend on the privileges that are currently in effect for `PUBLIC`. If `PUBLIC` loses a privilege, objects that are created with that privilege can be dropped or invalidated. The following examples demonstrate how certain objects depend on `PUBLIC` not losing its privileges.

**Example:** Suppose that Juan has the ID `USER1` and that Meg has the ID `USER2`. Juan creates a table `TAB1` and grants `ALL PRIVILEGES` on it to `PUBLIC`. Juan does not explicitly grant any privileges on the table to Meg's ID, `USER2`. Using the `PUBLIC` privileges, Meg creates a view on `TAB1`. Because the ID `USER2` requires the `SELECT` privilege on `TAB1` to create the view, the view is dropped if `PUBLIC` loses the privilege.

### Related tasks

“Granting privileges to remote users”

## Granting privileges to remote users

A query that arrives at your local DB2 subsystem through the distributed data facility (DDF) is accompanied by an authorization ID. After connection processing, the ID can be translated to another value and be associated with secondary authorization IDs.

DB2 also uses the ID to determine if the connection is associated with a trusted context. As the end result of these processes, the remote query is associated with a set of IDs that is known to your local DB2 subsystem. You assign privileges to these IDs in the same way that you assign privileges to IDs that are associated with local queries.



### Related tasks

“Granting privileges to the PUBLIC ID” on page 61

## Granting privileges through views

You can grant most table privileges (except ALTER, REFERENCES, TRIGGER, and INDEX) on a view. By creating a view and granting privileges through it, you can give an ID access to only a specific combination of data.

The ability to grant privileges through views is sometimes called *field-level access control* or *field-level sensitivity*.

Suppose that you want the ID MATH110 to be able to extract the following column data from the sample employee table for statistical investigation: HIREDATE, JOB, EDLEVEL, SEX, SALARY, BONUS, and COMM for DSN8910.EMP. However, you want to impose the following restrictions:

- No access to employee names or identification numbers
- No access to data for employees hired before 1996
- No access to data for employees with an education level less than 13
- No access to data for employees whose job is MANAGER or PRES

You can create and name a view that shows exactly that combination of data.

To grant privileges to the view that you create:

1. Issue the following CREATE statement to create the desired view:



```
CREATE VIEW SALARIES AS
  SELECT HIREDATE, JOB, EDLEVEL, SEX, SALARY, BONUS, COMM
    FROM DSN81010.EMP
   WHERE HIREDATE > '1995-12-31' AND
         EDLEVEL >= 13 AND
         JOB <> 'MANAGER' AND
         JOB <> 'PRES';
```



2. Issue the following statement to grant the SELECT privilege on the SALARIES view to MATH110:

**GUI**

```
GRANT SELECT ON SALARIES TO MATH110;
```

**GUI**

After you grant the privilege, MATH110 can execute SELECT statements on the restricted set of data only. Alternatively, you can give an ID access to only a specific combination of data by using multilevel security with row-level granularity.

**Related tasks**

“Granting privileges with the GRANT statement”

“Revoking privileges with the REVOKE statement” on page 68

## Granting privileges with the GRANT statement

You can assign privileges to an ID or a role by issuing the GRANT statement.

Suppose that the Spiffy Computer Company wants to create a database to hold information that is usually posted on hallway bulletin boards. For example, the database might hold notices of upcoming holidays and bowling scores.

To create and maintain the tables and programs that are needed for this application, the Spiffy Computer Company develops the security plan shown in the following diagram.

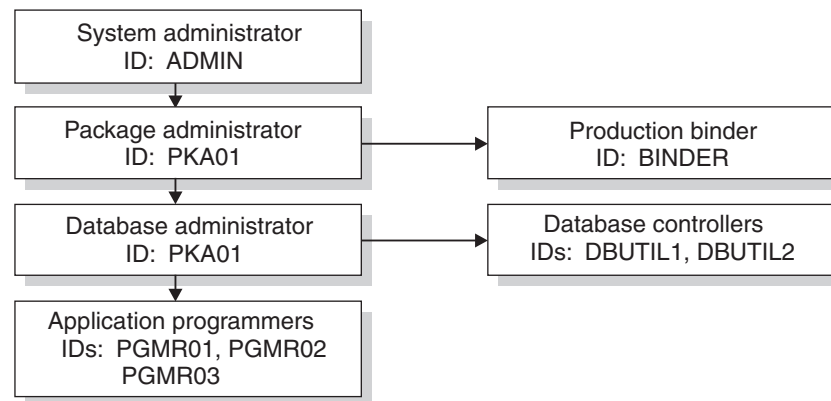
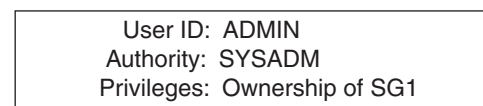


Figure 3. Security plan for the Spiffy Computer Company

The Spiffy Computer Company's system of privileges and authorities associates each role with an authorization ID. For example, the System Administrator role has the ADMIN authorization ID.

**GUI**

The system administrator uses the ADMIN authorization ID, which has the SYSADM authority, to create a storage group (SG1) and to issue the following statements:

1. GRANT PACKADM ON COLLECTION BOWLS TO PKA01 WITH GRANT OPTION;

This statement grants to PKA01 the CREATE IN privilege on the collection BOWLS and BIND, EXECUTE, and COPY privileges on all packages in the collection. Because ADMIN used the WITH GRANT OPTION clause, PKA01 can grant those privileges to others.

2. GRANT CREATEDBA TO DBA01;

This statement grants to DBA01 the privilege to create a database and to have DBADM authority over that database.


3. GRANT USE OF STOGROUP SG1 TO DBA01 WITH GRANT OPTION;

This statement allows DBA01 to use storage group SG1 and to grant that privilege to others.

4. GRANT USE OF BUFFERPOOL BP0, BP1 TO DBA01 WITH GRANT OPTION;

This statement allows DBA01 to use buffer pools BP0 and BP1 and to grant that privilege to others.

5. GRANT CREATE IN COLLECTION DSN8CC91 TO ROLE ROLE1;

This statement grants to ROLE1 the privilege to create new packages in collections DSN8CC91. 

User ID: PKA01  
Authority: PACKADM over the collection BOWLS


The package administrator, PKA01, controls the binding of packages into collections. PKA01 can use the CREATE IN privilege on the collection BOWLS and the BIND, EXECUTE, and COPY privileges on all packages in the collection. PKA01 also has the authority to grant these privileges to others.

User ID: DBA01  
Authority: DBADM over DB1  
Privileges: CREATEDBA  
Use of SG1 with GRANT  
Use of BP0 and BP1 with GRANT  
Ownership of DB1

The database administrator, DBA01, using the CREATEDBA privilege, creates the database DB1. When DBA01 creates DB1, DBA01 automatically has DBADM authority over the database.

User ID: DBUTIL1, DBUTIL2  
Authority: DBCTRL over DB1

The database administrator at Spiffy Computer Company wants help with running the COPY and RECOVER utilities. Therefore DBA01 grants DBCTRL authority over database DB1 to DBUTIL1 and DBUTIL2.

 To grant DBCTRL authority, the database administrator issues the following statement:

GRANT DBCTRL ON DATABASE DB1 TO DBUTIL1, DBUTIL2;

## Related tasks

“Granting privileges through views” on page 62

## Granting privileges to secondary IDs

The Spiffy Computer Company uses RACF to manage external access to DB2. Therefore, Spiffy can use secondary authorization IDs to define user groups and associate primary authorization IDs with those user groups.

The primary authorization IDs are the RACF user IDs. The secondary authorization IDs are the names of the groups with which the primary IDs are associated.

Spiffy can grant DB2 privileges to primary IDs indirectly, by granting privileges to secondary IDs that are associated with the primary IDs. This approach associates privileges with a *functional ID* rather than an *individual ID*. Functional IDs, also called group IDs, are granted privileges based on the function that certain job roles serve in the system. Multiple primary IDs can be associated with a functional ID and receive the privileges that are granted to that functional ID. In contrast, individual IDs are connected to specific people. Their privileges need to be updated as people join the company, leave the company, or serve different roles within the company. Functional IDs have the following advantages:

- Functional IDs reduce system maintenance because they are more permanent than individual IDs. Individual IDs require frequent updates, but each functional ID can remain in place until Spiffy redesigns its procedures.

**Example:** Suppose that Joe retires from the Spiffy Computer Company. Joe is replaced by Mary. If Joe's privileges are associated with functional ID DEPT4, those privileges are maintained in the system even after Joe's individual ID is removed from the system. When Mary enters the system, she will have all of Joe's privileges after her ID is associated with the functional ID DEPT4.

- Functional IDs reduce the number of grants that are needed because functional IDs often represent groups of individuals.
- Functional IDs reduce the need to revoke privileges and re-create objects when they change ownership.

**Example:** Suppose that Bob changes jobs within the Spiffy Computer Company. Bob's individual ID has privileges on many objects in the system and owns three databases. When Bob's job changes, he no longer needs privileges over these objects or ownership of these databases. Because Bob's privileges are associated with his individual ID, a system administrator needs to revoke all of Bob's privileges on objects and drop and re-create Bob's databases with a new owner. If Bob received privileges by association with a functional ID, the system administrator would only need to remove Bob's association with the functional ID.

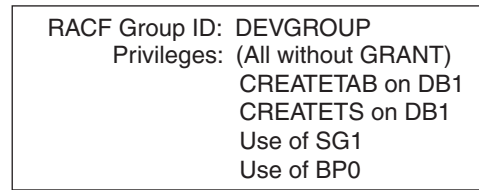
## Granting privileges to user groups

You can simplify the assignment and management of privileges by creating user groups and by granting privileges to the groups. In this way, you can efficiently assign the same set of privileges to all the users of a given group at the same time.

Suppose that the database administrator at Spiffy wants several employees in the Software Support department to create tables in the DB1 database. The database administrator creates DEVGROU as a RACF group ID for this purpose. To simplify the process, the database administrator decides that each CREATE TABLE statement should implicitly create a unique table space for the table. Hence,



DEVGROU needs the CREATETAB privilege, the CREATETS privilege, the privilege to use the SG1 storage group and, the privilege to use one of the buffer pools, BP0, for the implicitly created table spaces. The following diagram shows this group and their privileges:



**GUIP** The database administrator, DBA01, owns database DB1 and has the privileges to use storage group SG1 and buffer pool BP0. The database administrator holds both of these privileges with the GRANT option. The database administrator issues the following statements:

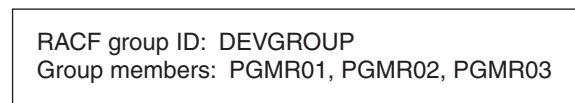
1. GRANT CREATETAB, CREATETS ON DATABASE DB1 TO DEVGROUP;
2. GRANT USE OF STOGROUP SG1 TO DEVGROUP;

3. GRANT USE OF BUFFERPOOL BP0 TO DEVGROUP; **GUIP**

Because the system and database administrators at Spiffy still need to control the use of those resources, the preceding statements are issued without the GRANT option.

Three programmers in the Software Support department write and test a new program, PROGRAM1. Their IDs are PGMR01, PGMR02, and PGMR03. Each programmer needs to create test tables, use the SG1 storage group, and use one of the buffer pools. All of those resources are controlled by DEVGROUP, which is a RACF group ID.

Therefore, granting privileges over those resources specifically to PGMR01, PGMR02, and PGMR03 is unnecessary. Each ID should be associated with the RACF group DEVGROUP and receive the privileges that are associated with that functional ID. The following diagram shows the DEVGROUP and its members:



The security administrator connects as many members as desired to the group DEVGROUP. Each member can exercise all the privileges that are granted to the group ID.

## Granting privileges for binding plans

Binding requires additional privileges. You must have the required privileges to bind a plan.

**GUIP** Suppose that three programmers can share the tasks that are done by the DEVGROUP ID. Someone creates a test table, DEVGROUP.T1, in database DB1 and loads it with test data. Someone writes a program, PROGRAM1, to display bowling scores that are contained in T1. Someone must bind the plan and packages that accompany the program.



Binding requires an additional privilege. ADMIN, who has the SYSADM authority, grants the required privilege by issuing the following statement:

```
GRANT BINDADD TO DEVGROUP;
```

#### GUIP

With that privilege, any member of the RACF group DEVGROUP can bind plans and packages that are to be owned by DEVGROUP. Any member of the group can rebind a plan or package that is owned by DEVGROUP. The following diagram shows the BINDADD privilege granted to the group:

RACF group ID: DEVGROUP Privilege: BINDADD
---

The Software Support department proceeds to create and test the program.

### Granting privileges for rebinding plans and packages

Spiffy has a different set of tables, which contain actual data that is owned by the ROLE PRODCN. PROGRAM1 is written with unqualified table names.

For example, table T1 was referred to as simply T1, not DEVGROUP.T1. The new packages and plan must refer to table PRODCN.T1. To move the completed program into production, someone must perform the following steps:

- Rebind the application plan with the owner PRODCN.
- Rebind the packages into the collection BOWLS, again with the owner PRODCN.

Spiffy gives that job to a production binder with the ID BINDER. BINDER needs privileges to bind a plan or package that DEVGROUP owns, to bind a plan or package with OWNER (PRODCN), and to add a package to the collection BOWLS. BINDER acquires these abilities through its RACF DEVGROUP group and ROLE PRODCN. ROLE PRODCN needs to have all the necessary privileges.

Suppose that ID BINDER has ROLE PRODCN when binding in a trusted context and that ROLE PRODCN has the following privileges:

DB2 Role: PRODCN Privileges: BINDADD CREATE IN collection BOWLS Privileges on SQL objects referenced in application
--

BINDER can bind plans and packages for owner ROLE PRODCN because it performs binds in a trusted context with ROLE PRODCN.

PACKADM, the package administrator for BOWLS, can grant the CREATE privilege with the following statement:

#### GUIP

```
GRANT CREATE ON COLLECTION BOWLS TO ROLE PRODCN;
```

#### GUIP

With the plan in place, the database administrator at Spiffy wants to make the PROGRAM1 plan available to all employees by issuing the following statement:

**GUIP**

```
GRANT EXECUTE ON PLAN PROGRAM1 TO PUBLIC;
```

**GUIP**

More than one ID has the authority or privileges that are necessary to issue this statement. For example, ADMIN has SYSADM authority and can grant the EXECUTE privilege. Also, any ID in a trusted context with ROLE PRODCNTN that owns PROGRAM1 can issue the statement. When EXECUTE is granted to PUBLIC, other IDs do not need any explicit authority on T1.

Finally, the plan to display bowling scores at Spiffy Computer Company is complete. The production plan, PROGRAM1, is created, and all IDs have the authority to execute the plan.

### Granting privileges for accessing distributed data

Some time after the system and database administrators at Spiffy Computer Company implement their security plan, the company president tells them that other applications on other systems must connect to the local DB2 subsystem. She wants people at every location to be able to access bowling scores through PROGRAM1 on the local subsystem.

**GUIP**

The administrators perform the following steps to enable access from all Spiffy locations:

1. Add a CONNECT statement to the program, naming the location at which table PRODCNTN.T1 resides. In this case, the table and the package reside at only the central location.
2. Issue the following statement so that PKA01, who has PACKADM authority, can grant the required privileges to DEVGROUP:  

```
GRANT CREATE IN COLLECTION BOWLS TO DEVGROUP;
```
3. Bind the SQL statements in PROGRAM1 as a package.
4. Bind the SQL statements in PROGRAM1 as a package by the package owner:  

```
GRANT EXECUTE ON PACKAGE PROGRAM1 TO PUBLIC;
```

Any system that is connected to the original DB2 location can run PROGRAM1 and execute the package by using DRDA<sup>®</sup> access. However, if the remote system is another DB2, a plan must be bound there that includes the package in its package list. **GUIP**

## Revoking privileges with the REVOKE statement

You can use the REVOKE statement to remove the privileges that you explicitly grant to an ID or a role.

**GUIP**

For example, you can revoke the privilege that you grant to an ID by issuing the following statement:

```
REVOKE authorization-specification FROM auth-id
```

Generally, you can revoke only the privileges that you grant. If you revoke privileges on a procedure or package, the privileges are revoked from all versions of that procedure or package.

However, an ID with the SECADM or ACCESSCTRL authority can revoke a privilege that has been granted by another ID with the following statement:

```
REVOKE authorization-specification FROM auth-id BY auth-id
```

If the SEPARATE SECURITY system parameter on panel DSNTIPP1 is set to NO (the default) during installation, an ID with the SYSADM or SYSCTRL authority can revoke a privilege that has been granted by another ID. In this case, the SYSADM authority implicitly has the privileges of the SECADM authority, and the SYSCTRL authority implicitly has the privileges of the ACCESSCTRL authority.

The BY clause specifies the authorization ID that originally granted the privilege. If two or more grantors grant the same privilege to an ID, executing a single REVOKE statement does not remove the privilege. To remove it, each grant of the privilege must be revoked.

The WITH GRANT OPTION clause of the GRANT statement allows an ID to pass the granted privilege to others. If the privilege is removed from the ID, its revocation can cascade to others depending on the setting of the REVOKE DEPRIV system parameter. For example, when a privilege is removed from authorization ID X, it is also removed from any ID to which X granted it, unless that ID also has the privilege from some other source.

**Example:** Suppose that DBA01 grants DBCTRL authority with the GRANT option on database DB1 to DBUTIL1. Then DBUTIL1 grants the CREATETAB privilege on DB1 to PGMR01. If DBA01 revokes DBCTRL from DBUTIL1, PGMR01 loses the CREATETAB privilege. If PGMR01 also granted the CREATETAB privilege to OPER1 and OPER2, they also lose the privilege.

**Example:** Suppose that PGMR01 from the preceding example created table T1 while holding the CREATETAB privilege. If PGMR01 loses the CREATETAB privilege, table T1 is not dropped, and the privileges that PGMR01 has as owner of the table are not deleted. Furthermore, the privileges that PGMR01 grants on T1 are not deleted. For example, PGMR01 can grant SELECT on T1 to OPER1 as long as PGMR01 owns the table. Even when the privilege to create the table is revoked, the table remains, the privilege remains, and OPER1 can still access T1.


**Example:** Consider the following REVOKE scenario:

1. Grant #1: SYSADM, SA01, grants SELECT on TABLE1 to USER01 with the GRANT option.
2. Grant #2: USER01 grants SELECT on TABLE1 to USER02 with the GRANT option.
3. Grant #3: USER02 grants SELECT on TABLE1 back to SA01.
4. USER02 then revokes SELECT on TABLE1 from SA01.

The cascade REVOKE process of Grant #3 determines if SA01 granted SELECT to anyone else. It locates Grant #1. Because SA01 did not have SELECT from any other source, this grant is revoked. The cascade REVOKE process then locates Grant #2 and revokes it for the same reason. In this scenario, the single REVOKE

---

1. DB2 does not cascade a revoke of the SYSADM authority from the installation SYSADM authorization IDs.

action by USER02 triggers and results in the cascade removal of all the grants even though SA01 has the SYSADM authority. The SYSADM authority is not considered. 

#### Related tasks

“Granting privileges with the GRANT statement” on page 63

“Revoking dependent privileges”

### Revoking dependent privileges

Revoking a privilege or authority, such as the SYSADM authority, from one user (an authorization ID or role) can result in the automatic removal of that privilege from other users and the privileges that it has granted. To prevent this, you can assign the REVOKE DEP PRIV parameter different values to control whether or not dependent privileges or authorities should be removed.

To specify the REVOKE DEP PRIV parameter, use one of the following approaches:

- Set REVOKE DEP PRIV to SQLSTMT (the default) if you want to use the dependent privileges clause on the REVOKE statement to control the revocation of dependent privileges.
  - Specify the NOT INCLUDING DEPENDENT PRIVILEGES clause on the REVOKE statement when you need to revoke a privilege or authority from a user but retain all the grants that are already made by that user. However, if the same privilege is later granted to that user again and subsequently revoked with the INCLUDING DEPENDENT PRIVILEGES clause specified, all dependent privileges including the grants made by the user earlier are removed.
  - Specify the INCLUDING DEPENDENT PRIVILEGES clause (the default) when you need to revoke a privilege or authority (other than ACCESSCTRL, DATAACCESS, and system DBADM) from a user and remove all the privileges or authorities that are already granted by that privilege or authority.
- Set REVOKE DEP PRIV to YES if you want to remove all dependent privileges or authorities whenever you revoke a privilege or authority other than ACCESSCTRL, DATAACCESS, and system DBADM.

You will receive an error if you specify the NOT INCLUDING DEPENDENT PRIVILEGES clause on the REVOKE statement when you revoke a privilege or authority other than ACCESSCTRL, DATAACCESS, and system DBADM.
- Set REVOKE DEP PRIV to NO if you want to retain all dependent privileges or authorities whenever you revoke a privilege or authority.

You will receive an error if you specify the INCLUDING DEPENDENT PRIVILEGES clause on the REVOKE statement.

If REVOKE DEP PRIV is set to NO or SQLSTMT or if the NOT INCLUDING DEPENDENT PRIVILEGES clause is specified on the REVOKE statement, dependent privileges or authorities are not revoked when a privilege or authority is revoked from a user. However, any packages, views, or MQTs that are owned by that user are invalidated, inoperative, or dropped.

Revoking dependent privileges does not occur in any of the following conditions:

- If the ACCESSCTRL authority is revoked from a user, grants made by the user are not revoked. However, if the user has already revoked its own grants prior to the removal of the ACCESSCTRL authority, that revocation of dependent privileges continues to take effect unless otherwise instructed through the REVOKE\_DEP\_PRIV parameter or the REVOKE statement.

- If the SECADM authority is removed from a user, grants made by the user are not revoked. However, if the user has already revoked its own grants prior to the removal of the SECADM authority, that revocation of dependent privileges continues to take effect unless otherwise instructed through the REVOKE\_DEP\_PRIV parameter or the REVOKE statement.

## Revoking privileges granted by multiple IDs

A user can be granted the same privilege by multiple IDs at different times, but that privilege and any dependent privileges can be simultaneously revoked.

**GUIP** Suppose that DBUTIL1 grants the CREATETAB privilege to PGMR01 and that DBUTIL2 also grants the CREATETAB privilege to PGMR01. The second grant is recorded in the catalog, with its date and time, but it has no other effect until the grant from DBUTIL1 to PGMR01 is revoked. After the first grant is revoked, DB2 must determine the authority that PGMR01 used to grant CREATETAB to OPER1. The following diagram illustrates the situation; the arrows represent the granting of the CREATETAB privilege.

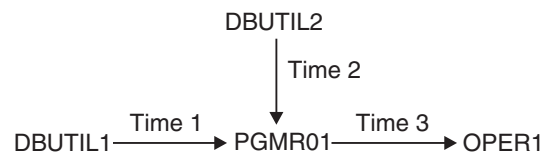


Figure 4. Authorization granted by two or more IDs

Suppose that DBUTIL1 issues the GRANT statement at Time 1 and that DBUTIL2 issues the GRANT statement at Time 2. DBUTIL1 and DBUTIL2 both use the following statement to issue the grant:

```
GRANT CREATETAB ON DATABASE DB1 TO PGMR01 WITH GRANT OPTION;
```

At Time 3, PGMR01 grants the privilege to OPER1 by using the following statement:

```
GRANT CREATETAB ON DATABASE DB1 TO OPER1;
```

After Time 3, DBUTIL1's authority is revoked, along with all of the privileges and authorities that DBUTIL1 granted. However, PGMR01 also has the CREATETAB privilege from DBUTIL2, so PGMR01 does not lose the privilege. The following criteria determine whether OPER1 loses the CREATETAB privilege when DBUTIL1's authority is revoked:

- If Time 3 *comes after* Time 2, OPER1 does not lose the privilege. The recorded dates and times show that, at Time 3, PGMR01 could have granted the privilege entirely on the basis of the privilege that was granted by DBUTIL2. That privilege was not revoked.
- If Time 3 *precedes* Time 2, OPER1 does lose the privilege. The recorded dates and times show that, at Time 3, PGMR01 could have granted the privilege only on the basis of the privilege that was granted by DBUTIL1. That privilege was revoked, so the privileges that are dependent on it are also revoked. **GUIP**

## Revoking privileges granted by all IDs

An ID with the SYSADM or SYSCTRL authority can revoke privileges that are granted by other IDs.

**GUIP** To revoke the CREATETAB privileges that are granted to PGMR01 on database DB1 by all IDs, use the following statement:

```
REVOKE CREATETAB ON DATABASE DB1 FROM PGMR01 BY ALL;
```

However, you might want to revoke only privileges that are granted by a specific ID. To revoke privileges that are granted by DBUTIL1 and to leave intact the same privileges if they were granted by any other ID, use the following statement:

```
REVOKE CREATETAB, CREATETS ON DATABASE DB1 FROM PGMR01 BY DBUTIL1;
```

**GUIP**

## Revoking privileges granted by a role

You can use the REVOKE statement to revoke privileges that are granted by a role in a trusted context.

To revoke privileges that are granted by a role, you can issue the REVOKE statement in the trusted context that was defined with the ROLE AS OBJECT OWNER clause. Also, make sure the role that revokes a privilege matches the one that grants the privilege. For a static REVOKE statement, the revoker is the role that owns the plan or package. For a dynamic REVOKE statement, the role for the primary authorization ID that executes the REVOKE statement becomes the revoker.

An authorization ID or role that has the SYSADM or SYSCTRL authority can use the BY (ROLE *role-name*) clause of the REVOKE statement to revoke privileges that are granted by a role.

## Revoking all privileges from a role

You can revoke all privileges that are assigned to a role by dropping the role itself or by using the REVOKE statement.

**GUIP** When you attempt to drop a role, make sure that the role does not own any objects. If the role owns objects, the DROP statement is terminated. If the role does not own any objects, the role is dropped. As a result, all privileges that are held by this role are revoked, and the revocation is cascaded. **GUIP**

## Revoking privileges for views

If a table privilege is revoked from the owner of a view on the table, the corresponding privilege on the view is revoked. The same privilege is also revoked from other IDs if it was granted by the view owner.

**GUIP** If the SELECT privilege on the base table is revoked from the owner of the view, the view is dropped. However, if another grantor granted the SELECT privilege to the view owner before the view was created, the view is not dropped.

**Example:** Suppose that OPER2 has the SELECT and INSERT privileges on table T1 and creates a view of the table. If the INSERT privilege on T1 is revoked from OPER2, all insert privileges on the view are revoked. If the SELECT privilege on T1 is revoked from OPER2, and if OPER2 did not have the SELECT privilege from another grantor before the view was created, the view is dropped.




If a view uses a user-defined function, the view owner must have the EXECUTE privilege on the function. If the EXECUTE privilege is revoked, the revoke fails because the view is using the privilege and the RESTRICT clause prevents the revoke.

An authorization ID with the SYSADM authority can create a view for another authorization ID. In this case, the view could have both a creator and an owner. The owner is automatically given the SELECT privilege on the view. However, the privilege on the base table determines whether the view is dropped.


**Example:** Suppose that IDADM, with SYSADM authority, creates a view on TABLX with OPER as the owner of the view. OPER now has the SELECT privilege on the view, but not necessarily any privileges on the base table. If SYSADM is revoked from IDADM, the SELECT privilege on TABLX is gone and the view is dropped.

If one ID creates a view for another ID, the catalog table SYSIBM.SYSTABAUTH needs either one or two rows to record the associated privileges. The number of rows that DB2 uses to record the privilege is determined by the following criteria:


- If IDADM creates a view for OPER when OPER has enough privileges to create the view by itself, only one row is inserted in SYSTABAUTH. The row shows only that OPER granted the required privileges.
- If IDADM creates a view for OPER when OPER does not have enough privileges to create the view by itself, two rows are inserted in SYSTABAUTH. One row shows IDADM as GRANTOR and OPER as GRANTEE of the SELECT privilege. The other row shows any other privileges that OPER might have on the view because of privileges that are held on the base table. 

### Revoking privileges for materialized query tables

If the SELECT privilege on a source table is revoked from the owner of a materialized query table, the corresponding privilege on the materialized query table is revoked. The same privilege is also revoked from other IDs if it was granted by the table owner.

 If the SELECT privilege on the source table is revoked from the owner of a materialized query table, the materialized query table is dropped. However, if another grantor granted the SELECT privilege to the materialized query table owner before the materialized query table was created, the materialized query table is not dropped.

**Example:** Suppose that OPER7 has the SELECT privilege on table T1 and creates a materialized query table T2 by selecting from T1. If the SELECT privilege on T1 is revoked from OPER7, and if OPER7 did not have the SELECT privilege from another grantor before T2 was created, T2 is dropped.

If a materialized query table uses a user-defined function, the owner of the materialized query table must have the EXECUTE privilege on the function. If the EXECUTE privilege is revoked, the revoke fails because the materialized query table is using the privilege and the RESTRICT clause prevents the revoke. 

### Revoking privileges for plans or packages

If the owner of an application plan or package loses a required privilege and does not have that privilege from another source, DB2 invalidates the package.

**GUPI** **Example:** Suppose that OPER2 has the SELECT and INSERT privileges on table T1 and creates a package that uses SELECT, but not INSERT. When privileges are revoked from OPER2, the plan is affected in the following ways:

- If the INSERT privilege is revoked, the plan is unaffected.
- If the revoked privilege was EXECUTE on a user-defined function, DB2 marks the package **inoperative** instead of invalid.

If authorization data is cached for a package and an ID loses EXECUTE authority on the package, that ID is removed from the cache. Similarly, if authorization data is cached for routines, a revoke or cascaded revoke of EXECUTE authority on a routine, or on all routines in a schema (schema.\*), from any ID causes the ID to be removed from the cache.

If authorization data is cached for plans, a revoke of EXECUTE authority on the plan from any ID causes the authorization cache to be invalidated.

If an application is caching dynamic SQL statements, and a privilege is revoked that was needed when the statement was originally prepared and cached, that statement is removed from the cache. Subsequent PREPARE requests for that statement do not find it in the cache and therefore execute a full PREPARE. If the plan or package is bound with KEEP\_DYNAMIC(YES), which means that the application does not need to explicitly re-prepare the statement after a commit operation, you might get an error on an OPEN, DESCRIBE, or EXECUTE of that statement following the next commit operation. The error can occur because a prepare operation is performed implicitly by DB2. If you no longer have sufficient authority for the prepare, the OPEN, DESCRIBE, or EXECUTE request fails. **GUPI**

## Revoking the SYSADM authority from users

You can revoke the SYSADM authority from users (IDs or roles) without revoking dependent privileges.

Revoking the SYSADM authority causes the revoking of dependent privileges, by default. If you want to leave the grants that they had made, you can issue the REVOKE statement with the NOT INCLUDING DEPENDENT PRIVILEGES clause, assuming the REVOKE\_DEP\_PRIVILEGES system parameter is set to SQLSTMT.

## Restrictions on privilege revocation

You can specify the RESTRICT clause of the REVOKE statement to impose limitations on privilege revocation.

**GUPI** Whether specified or not, the RESTRICT clause of the REVOKE statement always applies to the following objects:

- User-defined functions
- JARs (Java classes for a routine)
- Stored procedures
- Distinct types
- Sequences

When an attempt is made to revoke a privilege on one of these objects, DB2 determines whether the revokee owns an object that is dependent on the privilege. If such a dependency exists, the REVOKE statement proceeds only if the revokee also holds this privilege from another grantor or holds this privilege indirectly (such as if PUBLIC has this privilege, or if the revokee has SYSADM authority).



**Example:** Consider the following scenario:

1. UserA creates a user-defined function named UserA.UDFA.
2. UserA grants EXECUTE on UserA.UDFA to UserB.
3. User B then creates a user-defined function UserB.UDFB that is sourced on UserA.UDFA.

At this point, UserA attempts to revoke the EXECUTE privilege on UserA.UDFA from UserB. The revoke succeeds or fails based on the following criteria:

- If UserB has the EXECUTE privilege on UserA.UDFA only from UserA, the revoke fails with an accompanying message that indicates that a dependency on this privilege.
- If UserB has the EXECUTE privilege on UserA.UDFA from another source, directly or indirectly, the EXECUTE privilege that was granted by UserA is revoked successfully.

For distinct types, the following objects that are owned by the revokee can have dependencies:

- A table that has a column that is defined as a distinct type
- A user-defined function that has a parameter that is defined as a distinct type
- A stored procedure that has a parameter that is defined as a distinct type
- A sequence that has a parameter that is defined as a distinct type

For user-defined functions, the following objects that are owned by the revokee can have dependencies:

- Another user-defined function that is sourced on the user-defined function
- A view that uses the user-defined function
- A table that uses the user-defined function in a check constraint or user-defined default clause
- A trigger package that uses the user-defined function

For JARs (Java classes for a routine), the following objects that are owned by the revokee can have dependencies:

- A Java user-defined function that uses a JAR
- A Java stored procedure that uses a JAR

For stored procedures, a trigger package that refers to the stored procedure in a CALL statement can have dependencies.

For sequences, the following objects that are owned by the revokee can have dependencies:

- Triggers that contain NEXT VALUE or PREVIOUS VALUE expressions that specify a sequence
- Inline SQL routines that contain NEXT VALUE or PREVIOUS VALUE expressions that specify a sequence

One way to ensure that the REVOKE statement succeeds is to drop the object that has a dependency on the privilege. To determine which objects are dependent on which privileges before attempting the revoke, use the following SELECT statements.

For a distinct type:

- List all tables that are owned by the revokee USRT002 that contain columns that use the distinct type USRT001.UDT1:

```
SELECT * FROM SYSIBM.SYSCOLUMNS WHERE
  TBCREATOR = 'USRT002'      AND
  TYPESHEMA = 'USRT001'     AND
  TYPENAME = 'UDT1'         AND
  COLTYPE = 'DISTINCT';
```

- List the user-defined functions that are owned by the revokee USRT002 that contain a parameter that is defined as distinct type USRT001.UDT1:

```
SELECT * FROM SYSIBM.SYSPARMS WHERE
  OWNER = 'USRT002'          AND
  TYPESHEMA = 'USRT001'     AND
  TYPENAME = 'UDT1'         AND
  ROUTINETYPE = 'F';
```

- List the stored procedures that are owned by the revokee USRT002 that contain a parameter that is defined as distinct type USRT001.UDT1:

```
SELECT * FROM SYSIBM.SYSPARMS WHERE
  OWNER = 'USRT002'          AND
  TYPESHEMA = 'USRT001'     AND
  TYPENAME = 'UDT1'         AND
  ROUTINETYPE = 'P';
```

- List the sequences that are owned by the revokee USRT002 that contain a parameter that is defined as distinct type USRT001.UDT1:

```
SELECT SYSIBM.SYSEQUENCES.SCHEMA, SYSIBM.SYSEQUENCES.NAME
  FROM SYSIBM.SYSEQUENCES, SYSIBM.SYSDATATYPES WHERE
    SYSIBM.SYSEQUENCES.DATATYPEID = SYSIBM.SYSDATATYPES.DATATYPEID AND
    SYSIBM.SYSDATATYPES.SCHEMA = 'USRT001' AND
    SYSIBM.SYSDATATYPES.NAME = 'UDT1';
```

For a user-defined function:

- List the user-defined functions that are owned by the revokee USRT002 that are sourced on user-defined function USRT001.SPECUDF1:

```
SELECT * FROM SYSIBM.SYSROUTINES WHERE
  OWNER = 'USRT002'          AND
  SOURCESHEMA = 'USRT001'    AND
  SOURCESPECIFIC = 'SPECUDF1' AND
  ROUTINETYPE = 'F';
```

- List the views that are owned by the revokee USRT002 that use user-defined function USRT001.SPECUDF1:

```
SELECT * FROM SYSIBM.SYSVIEWDEP WHERE
  DCREATOR = 'USRT002'      AND
  BSHEMA = 'USRT001'       AND
  BNAME = 'SPECUDF1'       AND
  BTYPE = 'F';
```

- List the tables that are owned by the revokee USRT002 that use user-defined function USRT001.A\_INTEGER in a check constraint or user-defined default clause:

```
SELECT * FROM SYSIBM.SYSCONSTDEP WHERE
  DTBCREATOR = 'USRT002'    AND
  BSHEMA = 'USRT001'       AND
  BNAME = 'A_INTEGER'       AND
  BTYPE = 'F';
```

- List the trigger packages that are owned by the revokee USRT002 that use user-defined function USRT001.UDF4:

```
SELECT * FROM SYSIBM.SYSPACKDEP WHERE
  DOWNER = 'USRT002'        AND
  BQUALIFIER = 'USRT001'    AND
  BNAME = 'UDF4'            AND
  BTYPE = 'F';
```

For a JAR (Java class for a routine), list the routines that are owned by the revokee USRT002 and that use a JAR named USRT001.SPJAR:

```
SELECT * FROM SYSIBM.SYSROUTINES WHERE
  OWNER = 'USRT002'          AND
  JARCHEMA = 'USRT001'      AND
  JAR_ID = 'SPJAR';
```

For a stored procedure that is used in a trigger package, list the trigger packages that refer to the stored procedure USRT001.WLMLOCN2 that is owned by the revokee USRT002:

```
SELECT * FROM SYSIBM.SYSPACKDEP WHERE
  DOWNER = 'USRT002'        AND
  BQUALIFIER = 'USRT001'    AND
  BNAME = 'WLMLOCN2'        AND
  BTYPE = 'O';
```

For a sequence:

- List the sequences that are owned by the revokee USRT002 and that use a trigger named USRT001.SEQ1:

```
SELECT * FROM SYSIBM.SYSPACKDEP WHERE
  BNAME = 'SEQ1'
  BQUALIFIER = 'USRT001'
  BTYPE = 'Q'
  DOWNER = 'USRT002'
  DTYPE = 'T';
```

- List the sequences that are owned by the revokee USRT002 and that use a inline SQL routine named USRT001.SEQ1:

```
SELECT * FROM SYSIBM.SYSSEQUENCESDEP WHERE
  DCREATOR = 'USRT002'
  DTYPE = 'F'
  BNAME = 'SEQ1'
  BSCHEMA = 'USRT001';
```

 GUIP

---

## Managing implicit privileges

You acquire privileges implicitly through ownership of objects, including ownership of plans and packages. You can control access to data by managing those privileges through object ownership and stored procedures, which are also known as routines.

### Related tasks

“Managing explicit privileges” on page 60

## Managing implicit privileges through object ownership

Ownership of an object carries with it a set of related privileges on the object. DB2 provides separate controls for creation and ownership of objects.

In general, when you create an object, the owner of the object can be your primary authorization ID, one of your secondary authorization IDs, or the role that you are associated with in a trusted context.

### Ownership of objects with unqualified names

If an object name is unqualified, the object type and the way it is created determine its ownership.

**GUIP** If the name of a table, view, index, alias, or synonym is unqualified, you can establish the object's ownership in the following ways:

- If you issue the CREATE statement dynamically, perhaps using SPUFI, QMF, or some similar program, the owner of the created object is your current SQL ID. That ID must have the privileges that are needed to create the object.
- If you issue the CREATE statement statically, by running a plan or package that contains it, the ownership of the created object depends on the option that is used for the bind operation. You can bind the plan or package with either the QUALIFIER option, the OWNER option, or both.
  - If the plan or package is bound with the QUALIFIER option only, the authorization ID in the QUALIFIER option is the owner of the object. The QUALIFIER option allows the binder to name a qualifier to use for all unqualified names of tables, views, indexes, aliases, or synonyms that appear in the plan or package.
  - If the plan or package is bound with the OWNER option only, the authorization ID in the OWNER option is the owner of the object.
  - If the plan or package is bound with both the QUALIFIER option and the OWNER option, the authorization ID in the QUALIFIER option is the owner of the object.
  - If neither option is specified, the authorization ID of the binder of the plan or package is implicitly the object owner.

If the name of a user-defined function, stored procedure, distinct type, sequence, or trigger is unqualified, you can establish the ownership of one of these objects in these ways:

- If you issue the CREATE statement dynamically, the owner of the created object is your current SQL ID. That ID must have the privileges that are needed to create the object.
- If you issue the CREATE statement statically, by running a plan or package that contains it, the owner of the object is the plan or package owner. You can use the OWNER bind option to explicitly name the object owner. If you do not use the OWNER bind option, the binder of the package or plan is implicitly the object owner.

If the name of a user-defined function, stored procedure, distinct type, sequence, or trigger is unqualified, the implicit qualifier is determined based on the schema name in dynamic statements and the PATH bind option in static statements. The owner of a JAR (Java class for a routine) that is used by a stored procedure or a user-defined function is the current SQL ID of the process that performs the


INSTALL\_JAR function. **GUIP**

## Ownership of objects with qualified names

If an object name is qualified, the type of object indicates its ownership.


**GUIP** If you create a table, view, index, or alias with a qualified name, the owner of the object is the *schema name*. The schema name identifies the schema to which the object belongs. You can consider all of the objects that are qualified by the same schema name as a group of related objects.

If you create a distinct type, user-defined function, stored procedure, sequence, or trigger with a qualified name, the owner of the object is the authorization ID of the process. The owner of a JAR (Java class for a routine) that is used by a stored

procedure or a user-defined function is the current SQL ID of the process that performs the `INSTALL_JAR` function. 

## Ownership of objects within a trusted context

You can simplify the administration of authorization by having roles as object owners. In addition, object ownership carries with it a set of related privileges on the object; you can prevent users from obtaining implicit privileges from object ownership by making roles object owners.

 If the owner of an object is an authorization ID and you need to transfer the ownership to another ID, you need to drop the object first and re-create it with the new authorization ID as the owner. You don't need to take these steps if the owner is a role because all users that are associated with that role have the owner privilege.

The definition of a trusted context determines the ownership of objects that are created in the trusted context. Assume that you issue the `CREATE` statement dynamically and that the trusted context is defined with the `ROLE AS OBJECT OWNER` clause. In this case, the associated role is the owner of the objects, regardless of whether the objects are explicitly qualified.

In contrast, assume that you issue the `CREATE` statement statically and that the plan or package is bound in the trusted context with the `ROLE AS OBJECT OWNER` clause. In this case, the role that owns the plan or package also owns the objects that are created, regardless of whether the objects are explicitly qualified.



### Related concepts

“Trusted contexts” on page 205

### Related reference

“Establishing plan and package ownership in a trusted context” on page 81

## Changing object ownership

You can make a DB2 role, a primary authorization ID, or a secondary authorization ID the owner of an object.

Object ownership carries with it a set of related privileges on the object. The privileges that are implicit in ownership cannot be revoked; you cannot replace or change the owner of an object while the object exists.

If you a DB2 role the owner of an object, you don't need to change or replace the ownership. All users that are associated with that role have the same owner privileges. To make a role the owner of an object, you need to create the object in a trusted context that is defined with the `ROLE AS OBJECT OWNER AND QUALIFIER` clause.

You can change the owner of an object from an authorization ID to a role by using the `CATMAINT UPDATE` utility with the `OWNER` option. To do so, you must also have the installation `SYSADM` authority, define a trusted context with the `ROLE AS OBJECT OWNER AND QUALIFIER` clause, and run the process in the new function mode.

Alternately, you can make the object owning ID a secondary ID with which several primary IDs are associated. You can change the list of primary IDs that are associated with the secondary ID without dropping and re-creating the object.

If the owner of the object is a primary authorization ID and if you need to transfer the ownership to another ID, you must drop the object and then recreate it with a new authorization ID as the owner.

### Granting implicit privileges of object ownership

Certain implicit privileges of ownership correspond to the privileges that can be granted by a GRANT statement. For the privileges that do correspond, the owner of the object can grant them to other users.

**GUIP** **Example:** The owner of a table can grant the SELECT privilege on the table to any other user. To grant the SELECT privilege on TABLE3 to USER4, the owner of the table can issue the following statement:

```
GRANT SELECT ON TABLE3 TO USER4
```

**GUIP**

## Managing implicit privileges through plan or package ownership

If you are the owner of a plan or package, you must hold privileges to perform actions on the plan or package. You can grant privileges to execute the plan or package to any ID.

**GUIP** When the EXECUTE privilege on a plan or package is granted to an ID, the ID can execute a plan or package without holding the privileges for every action that the plan or package performs. However, the ID is restricted by the SQL statements in the original program.

**Example:** The program might contain the following statement:

```
EXEC SQL
  SELECT * INTO :EMPREC FROM DSN81010.EMP
  WHERE EMPNO='000010';
```

The statement puts the data for employee number 000010 into the host structure EMPREC. The data comes from table DSN81010.EMP, but the ID does not have unlimited access to DSN81010.EMP. Instead, the ID that has EXECUTE privilege for this plan can access rows in the DSN81010.EMP table only when EMPNO = '000010'.

If any of the privileges that are required by the package are revoked from the owner, the package is invalidated. The package must be rebound, and the new owner must have the required privileges. **GUIP**

### Establishing or changing plan or package ownership

You can use the BIND and REBIND subcommands to create or change an application plan or a package.

On either subcommand, you can use the OWNER option to name the owner of the resulting plan or package. Consider the following factors when naming an owner:

- Any user can name the primary ID or any secondary ID.
- An ID with the BINDAGENT privilege can name the grantor of that privilege.
- An ID with SYSCTRL or SYSADM authority can name any authorization ID on a BIND command, but not on a REBIND command.

If you omit the OWNER option, your primary ID becomes the owner on BIND, and the previous owner retains ownership on REBIND.

Some systems that can bind a package at a DB2 system do not support the OWNER option. When the OWNER option is not supported, the primary authorization ID is always the owner of the package because a secondary ID cannot be named as the owner.

#### **Related reference**

“Establishing plan and package ownership in a trusted context”

### **Establishing plan and package ownership in a trusted context**

You can issue the BIND and REBIND commands in a trusted context with the ROLE AS OBJECT OWNER clause to specify the ownership of a plan or package. In this trusted context, you can specify only a role, not an authorization ID, as the OWNER of a plan or package.

**GUPI** If you specify the OWNER option, the specified role becomes the owner of the plan or package. If you don't specify the OWNER option, the role that is associated with the binder becomes the owner. If the ROLE AS OBJECT OWNER clause is omitted for the trusted context, the current rules for plan and package ownership apply.

**Considerations:** If you want a role to own the package at the remote DB2, you need to define the role ownership in the trusted context at the remote server. Make sure to establish the connection to the remote DB2 as trusted when binding or re-binding the package at the remote server.

If you specify the OWNER option in a trusted connection during the remote BIND processing, the outbound authorization ID translation is not performed for the OWNER.

If the plan owner is a role and the application uses a package bound at a remote DB2 for z/OS server, the privilege of the plan owner to execute the package is not considered at the remote DB2 server. The privilege set of the authorization ID (either the package owner or the process runner determined by the DYNAMICRULES behavior) at the DB2 for z/OS server must have the EXECUTE privilege on the package at the DB2 server. **GUPI**

#### **Related concepts**

“Trusted contexts” on page 205

#### **Related tasks**

“Establishing or changing plan or package ownership” on page 80

#### **Related reference**

“Ownership of objects within a trusted context” on page 79

### **How DB2 resolves unqualified names**


A plan or package can contain SQL statements that use unqualified table and view names.

**GUPI** For static SQL, the default qualifier for those names is the owner of the plan or package. However, you can use the QUALIFIER option of the BIND command to specify a different qualifier. For static statements, the PATH bind option determines the path that DB2 searches to resolve unqualified distinct types, user-defined functions, stored procedures, sequences, and trigger names.




When you perform bind operations on packages or plans that contain static SQL, you should use group and ROLE authority rather than individual ID authority whenever possible. The combinations of OWNER, QUALIFIER, SCHEMA, and ROLE ownership provide you more flexibility.

For plans or packages that contain dynamic SQL, DYNAMICRULES behavior determines how DB2 qualifies unqualified object names. For unqualified distinct types, user-defined functions, stored procedures, sequences, and trigger names in dynamic SQL statements, DB2 uses the schema name as the qualifier. DB2 finds the schema name in the CURRENT PATH special register. For unqualified tables, views, aliases, and indexes, DB2 uses the CURRENT SCHEMA special register as the qualifier.

**Exception:** ALTER, CREATE, DROP, COMMENT ON, GRANT, and REVOKE statements follow different conventions for assigning qualifiers. For static SQL, you must specify the qualifier for these statements in the QUALIFIER bind option. For dynamic SQL, the qualifier for these statements is the value in the CURRENT SCHEMA special register. 


### Validating authorization for executing plans or packages

The owner of a plan or package must have authorization to execute all static SQL statements that are embedded in the plan or package. A bind operation always checks whether a local object exists and whether the owner has the required privileges on it.

 However, you do not need to have the authorization when the plan or package is bound. The objects to which the plan or package refers do not even need to exist at bind time. If the initial checking fails, an error message is returned. You can choose whether the failure prevents the bind operation from completion by using the VALIDATE option on the BIND PLAN and BIND PACKAGE commands.

The following values for the VALIDATE option determine how DB2 is to handle existence and authorization errors:

- RUN** If you choose RUN for the VALIDATE option, the bind succeeds even when existence or authorization errors exist. DB2 checks existence and authorization at run time.
- BIND** If you choose BIND for the VALIDATE option, which is recommended, the bind fails when existence or authorization errors exist. **Exception:** If you use the SQLERROR(CONTINUE) option on the BIND PACKAGE command, the bind succeeds, but the package's SQL statements that have errors cannot execute.

The corresponding existence and authorization checks for remote objects are always made at run time. Authorization to execute dynamic SQL statements is also checked at run time. Applications that use the Resource Recovery Services attachment facility (RRSAF) to connect to DB2 do not require a plan. 

### Checking authorization at a DB2 database server:

A remote requester, either a DB2 for z/OS server or other requesting system, runs a package at the DB2 intermediate server. DB2 checks for the privileges that are required for service requests.



**GUIP** As shown in the following diagram, a statement in the package uses an alias or a three-part name to request services from a DB2 database server.

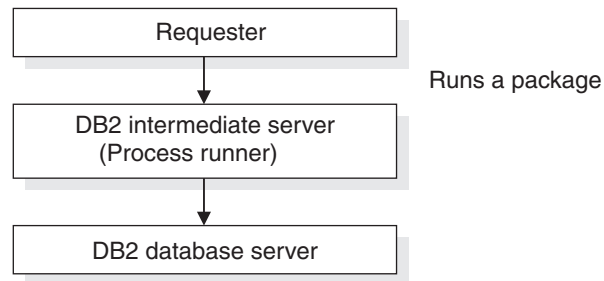


Figure 5. Execution at a second DB2 server

The ID that is checked for the required privileges to run at the DB2 database server can be:

- The owner of the plan, if not a role, that is running at the requester site (if the requester is DB2 for z/OS)  
If the owner of the plan is a role and the application uses a package bound at a remote DB2 for z/OS server, the authorization ID at the DB2 for z/OS server must have the EXECUTE privilege on the package at the DB2 server. The authorization ID can be the package owner or the process runner that is determined by the DYNAMICRULES behavior.
- The owner of the package that is running at the DB2 server

In addition, if a remote alias is used in the SQL statement, the alias must be defined at the requester site. The ID that is used depends on the following factors:

- Whether the requester is a DB2 for z/OS server or a different system
- The value of the DYNAMICRULES bind option
- Whether the SQL statement that is executed at the DB2 database server is static or dynamic

### Checking authorization for executing an RRSF application without a plan:

RRSAF provides the capability for an application to connect to DB2 and run without a DB2 plan.

If an RRSF application does not have a plan, the following authorization rules are true:

- For the following types of packages, the primary or secondary authorization ID and role of the process are used for checking authorization to execute the package:
  - A local package
  - A remote package that is on a DB2 for z/OS system and is accessed using DRDA
- At a DB2 for z/OS system, the authorization to execute the DESCRIBE TABLE statement includes checking the primary and secondary authorization IDs.
- For a double hop situation, the authorization ID that must hold the required privileges to execute SQL statements at the second server is determined as if the requester is not a DB2 for z/OS system.

## Caching authorization IDs for better performance

You can specify that DB2 is to cache authorization IDs for plans, packages, or routines (user-defined functions and stored procedures). Caching IDs can help improve performance, especially when IDs are frequently reused.

One cache exists for each plan, one global cache exists for packages, and a global cache exists for routines. The global cache for packages and routines are allocated at the DB2 startup. For a data sharing group, each member does its own authorization caching.

### Caching authorization IDs for plans:

Authorization checking is fastest when the plan is reused by an ID or role that already appears in the cache and when the EXECUTE privilege is granted to PUBLIC.

You can set the size of the plan authorization cache by using the BIND PLAN subcommand. The default cache size is specified by an installation option, with an initial default setting of 3072 bytes.

### Caching authorization IDs for packages:

DB2 authorization can cache roles or primary authorization IDs for handling packages. DB2 checks and caches a role if it is in effect and authorized. If a role is not in effect or authorized, DB2 checks and caches the primary authorization ID.

Caching roles or authorization IDs for packages can provide benefits for handling the following objects at run time:

- Stored procedures
- Remotely bound packages
- Local packages in a package list in which the plan owner does not have execute authority on the package at bind time, but does at run time
- Local packages that are not explicitly listed in a package list, but are implicitly listed by *collection-id.\**, *.\**, or *\*.package-id*

You can set the size of the package authorization cache by using the PACKAGE AUTH CACHE field on the DSNTIPP installation panel. The default value, 5 MB, is enough storage to support about 690 *collection-id.package-id* entries or *collection-id.\** entries.

You can cache more package authorization information by using any of the following strategies:

- Granting package execute authority to *collection.\**
- Increasing the size of the cache
- Granting package authority to a secondary ID or role when running in a trusted context
- Granting package execute authority to PUBLIC for some packages or collections



The QTPACAUT field in the package accounting trace indicates how often DB2 succeeds at reading package authorization information from the cache.



### Related reference

“Caching of EXECUTE on plans, packages, and routines” on page 247

### Caching authorization IDs for routines:

DB2 authorization can cache roles or primary authorization IDs for handling routines. DB2 checks and caches a role if it is in effect and authorized. If a role is not in effect or authorized, DB2 checks and caches the primary authorization ID.

The routine authorization cache stores roles or authorization IDs with the EXECUTE privilege on a specific routine. A routine is identified as *schema.routine-name.type*, where the routine name is one of the following names:

- The specific function name for user-defined functions
- The procedure name for stored procedures
- '\*' for all routines in the schema

You can set the size of the routine authorization cache by using the ROUTINE AUTH CACHE field on the DSNTIPP installation panel. The initial default size of 5 MB is enough storage to support about 690 *schema.routine.type* or *schema.\*.type* entries.

You can cache more authorization information about routines by using the following strategies:

- Granting EXECUTE on *schema.\**
- Increasing the size of the cache
- Granting package authority to a secondary ID or role when running in a trusted context
- Granting routine execute authority to PUBLIC for some or all routines in the schema.

#### Related reference

“Caching of EXECUTE on plans, packages, and routines” on page 247

### Authorizing plan or package access through applications

Because an ID executes a package or plan by running an application program, implementing control measures in an application program can be useful.

**Example:** Consider the following SQL statement:

**GUIP**

```
EXEC SQL
  SELECT * INTO :EMPREC FROM DSN81010.EMP
  WHERE EMPNO='000010';
```

**GUIP**

The statement permits access to the row of the employee table WHERE EMPNO='000010'. If you replace the value 000010 with a host variable, the program could supply the value of the variable and permit access to various employee numbers. Routines in the program could limit that access to certain IDs, certain times of the day, certain days of the week, or other special circumstances.

Stored procedures provide an alternative to controls in the application. By encapsulating several SQL statements into a single message to the DB2 server, a stored procedure can protect sensitive portions of the application program. Also, stored procedures can include access to non-DB2 resources, as well as DB2.

**Recommendation:** Do not use programs to extend security. Whenever possible, use other techniques, such as stored procedures or views, as a security mechanism. Using programs to extend security has the following drawbacks:

- Program controls are separate from other access controls, can be difficult to implement properly, are difficult to audit, and are relatively simple to bypass.
- Almost any debugging facility can be used to bypass security checks in a program.
- Other programs might use the plan without doing the needed checking.
- Errors in the program checks might allow unauthorized access.
- Because the routines that check security might be quite separate from the SQL statement, the security check could be entirely disabled without requiring a bind operation for a new plan.
- A BIND REPLACE operation for an existing plan does not necessarily revoke the existing EXECUTE privileges on the plan. (Revoking those privileges is the default, but the plan owner has the option to retain them. For packages, the EXECUTE privileges are always retained.)

For those reasons, if the program accesses any sensitive data, the EXECUTE privileges on the plan and on packages are also sensitive. They should be granted only to a carefully planned list of IDs.

### Restricting access of plans or packages to particular systems:

If you use controls in an application program, you can limit the access of a plan or package to the particular systems for which the application program is designed.

**GUIP** DB2 does not ensure that only specific programs are used with a plan, but program-to-plan control can be enforced in IMS and CICS. DB2 provides a consistency check to avoid accidental mismatches between program and plan, but the consistency check is not a security check.

You can use the the ENABLE and DISABLE options on the BIND and REBIND subcommands to restrict access of plans and packages to a particular system.

**Example:** The ENABLE IMS option allows the plan or package to run from any IMS connection. Unless other systems are also named, ENABLE IMS does not allow the plan or package to run from any other type of connection.

**Example:** DISABLE BATCH prevents a plan or package from running through a batch job, but it allows the plan or package to run from all other types of connection.

You can exercise even finer control with the ENABLE and DISABLE options. You can enable or disable particular IMS connection names, CICS application IDs, requesting locations, and so forth. **GUIP**

### Authorization checking for executing packages remotely:

The privileges that are required for a remote bind (BIND PACKAGE *location.collection*) must be granted at the server location.

**GUIP** The ID that owns the package must have all of the privileges that are required to run the package at the server, and BINDADD<sup>2</sup> and CREATE IN privileges at the server.


---

2. Or BIND, depending on the installation option BIND NEW PACKAGE.

### Exceptions:

- For a BIND COPY operation, the owner must have the COPY privilege at the local DB2 site or subsystem, where the package that is being copied resides.
- If the creator of the package is not the owner, the creator must have SYSCTRL authority or higher, or must have been granted the BINDAGENT privilege by the owner. That authority or privilege is granted at the local DB2.

Binding a plan with a package list (BIND PLAN PKLIST) is done at the local DB2, and bind privileges must be held there. Authorization to execute a package at a remote location is checked at execution time, as follows:

- If the server is a DB2 for z/OS subsystem:
  - If the subsystem parameter PRIVATE\_PROTOCOL is set to NO, the authorization ID of the process (primary ID or any secondary ID) must have the EXECUTE privilege for the package at the DB2 server.
  - If subsystem parameter PRIVATE\_PROTOCOL is set to AUTH, the owner of the plan at the DB2 requester must have the EXECUTE privilege on the package at the DB2 server.
- If the server is not DB2 for z/OS, the primary authorization ID must have whatever privileges are needed. Check that product's documentation. 

## Managing implicit privileges through routines

You can control authorization checking by using a DB2-supplied exit routine or an exit routine that you write. You can use the access control authorization routine to control authorization checking.

### Privileges required for executing routines

A number of steps are involved in implementing, defining, and invoking user-defined functions and stored procedures, which are also called *routines*.


 The following table summarizes the common tasks and the privileges that are required for executing routines.

Table 22. Common tasks and required privileges for routines

Role	Tasks	Required privileges
Implementer	If SQL is in the routine: codes, precompiles, and link-edits the program to use as the routine. Binds the program as the routine package.	If binding a package, BINDADD system privilege and CREATE IN on the collection.
	If no SQL is in the routine: codes, compiles, and link-edits the program.	
Definer	Issues a CREATE FUNCTION statement to define a user-defined function or CREATE PROCEDURE statement to define a stored procedure.	CREATEIN privilege on the schema. EXECUTE authority on the routine package when invoked.
Invoker	Invokes a routine from an SQL application.	EXECUTE authority on the routine.

The routine *implementer* typically codes the routine in a program and precompiles the program. If the program contains SQL statements, the implementer binds the DBRM. In general, the authorization ID that binds the DBRM into a package is the package owner. The implementer is the routine package owner. As package owner,

the implementer implicitly has EXECUTE authority on the package and has the authority to grant EXECUTE privileges to other users to execute the code within the package.

The implementer grants EXECUTE authority on the routine package to the definer. EXECUTE authority is necessary only if the package contains SQL. For user-defined functions, the definer requires EXECUTE authority on the package. For stored procedures, the EXECUTE privilege on the package is checked for the definer and other IDs.

The routine *definer* owns the routine. The definer issues a CREATE FUNCTION statement to define a user-defined function or a CREATE PROCEDURE statement to define a stored procedure. The definer of a routine is determined as follows:

- If the SQL statement is embedded in an application program, the definer is the authorization ID of the owner of the plan or package.
- If the SQL statement is dynamically prepared, the definer is the SQL authorization ID that is contained in the CURRENT SQLID special register. If the SQL statement is executed in a trusted context that is specified with the ROLE AS OBJECT OWNER clause, the definer is the role in effect.

The definer grants EXECUTE authority on the routine to the invoker, that is, any user that needs to invoke the routine.

The routine *invoker* invokes the routine from an SQL statement in the invoking plan or package. The invoker for a routine is determined as follows:

- For a static statement, the invoker is the plan or package owner.
- For a dynamic statement, the invoker depends on DYNAMICRULES behavior.

**GUPI**

## Granting privileges through routines

You can grant users the required privileges for implementing, defining, and using a user-defined function through exit routines.

### Implementing a user-defined function:

You can code an application program to implement a user-defined function.

**GUPI**

To implement a user-defined function:

1. The implementer codes a program that implements the user-defined function. Assume that the implementer codes the following external user-defined function in C and names the function C\_SALARY:

```
/******  
 * This routine accepts an employee serial number and a percent raise. *  
 * If the employee is a manager, the raise is not applied. Otherwise, *  
 * the new salary is computed, truncated if it exceeds the employee's *  
 * manager's salary, and then applied to the database. *  
*****/  
void C_SALARY                                /* main routine */  
( char *employeeSerial                      /* in: employee serial no. */  
  decimal *percentRaise                     /* in: percentage raise */  
  decimal *newSalary,                       /* out: employee's new salary */  
  short int *niEmployeeSerial               /* in: indic var, empl ser */  
  short int *niPercentRaise                 /* in: indic var, % raise */  
  short int *niNewSalary,                   /* out: indic var, new salary */  
  char *sqlstate,                           /* out: SQLSTATE */  
  char *fnName,                             /* in: family name of function*/
```

```

char      *specificName,      /* in: specific name of func */
char      *message            /* out: diagnostic message */
)
{
EXEC SQL BEGIN DECLARE SECTION;
char      hvEMPNO-7-;         /* host var for empl serial */
decimal   hvSALARY;           /* host var for empl salary */
char      hvWORKDEPT-3-;      /* host var for empl dept no. */
decimal   hvManagerSalary;    /* host var,emp's mgr's salary*/
EXEC SQL END DECLARE SECTION;

sqlstate = 0;
memset( message,0,70 );
/*****
 * Copy the employee's serial into a host variable
 *****/
strcpy( hvEMPNO,employeeSerial );
/*****
 * Get the employee's work department and current salary
 *****/
EXEC SQL SELECT  WORKDEPT, SALARY
                INTO :hvWORKDEPT, :hvSALARY
                FROM EMP
                WHERE EMPNO = :hvEMPNO;
/*****
 * See if the employee is a manager
 *****/
EXEC SQL SELECT  DEPTNO
                INTO :hvWORKDEPT
                FROM DEPT
                WHERE MGRNO = :hvEMPNO;
/*****
 * If the employee is a manager, do not apply the raise
 *****/
if( SQLCODE == 0 )
{
    newSalary = hvSALARY;
}
/*****
 * Otherwise, compute and apply the raise such that it does not
 * exceed the employee's manager's salary
 *****/
else
{
    /*****
     * Get the employee's manager's salary
     *****/
    EXEC SQL SELECT  SALARY
                    INTO :hvManagerSalary
                    FROM EMP
                    WHERE EMPNO = (SELECT MGRNO
                                   FROM DSN8610.DEPT
                                   WHERE DEPTNO = :hvWORKDEPT);
    /*****
     * Compute proposed raise for the employee
     *****/
    newSalary = hvSALARY * (1 + percentRaise/100);
    /*****
     * Don't let the proposed raise exceed the manager's salary
     *****/
    if( newSalary > hvManagerSalary
        newSalary = hvManagerSalary;
    /*****
     * Apply the raise
     *****/
    hvSALARY = newSalary;
    EXEC SQL UPDATE EMP

```

```

        SET SALARY = :hvSALARY
        WHERE EMPNO = :hvEMPNO;
    }

    return;
} /* end C_SALARY */

```

The implementer requires the UPDATE privilege on table EMP. Users with the EXECUTE privilege on function C\_SALARY do not need the UPDATE privilege on the table.

2. Because this program contains SQL, the implementer performs the following steps:
  - a. Precompile the program that implements the user-defined function.
  - b. Link-edit the user-defined function with DSNRLI (RRS attachment facility), and name the program's load module C\_SALARY.
  - c. Bind the DBRM into package MYCOLLID.C\_SALARY.

After performing these steps, the implementer is the *function package owner*.

3. The implementer then grants EXECUTE privilege on the user-defined function package to the definer.

```

GRANT EXECUTE ON PACKAGE MYCOLLID.C_SALARY
  TO definer

```

As package owner, the implementer can grant execute privileges to other users, which allows those users to execute code within the package. For example:

```

GRANT EXECUTE ON PACKAGE MYCOLLID.C_SALARY
  TO other_user

```

#### GUIP

### Defining a user-defined function:

You can define a user-defined function to perform specific operations by issuing the CREATE FUNCTION statement.

**GUIP** To define a user-defined function:

1. Issue the CREATE FUNCTION statement. For example, the following CREATE FUNCTION statement defines the user-defined function SALARY\_CHANGE to DB2:

```

CREATE FUNCTION
  SALARY_CHANGE(
    VARCHAR( 6 )
    DECIMAL( 5,2 ) )
  RETURNS
    DECIMAL( 9,2 )
  SPECIFIC schema.SALCHANGE
  LANGUAGE C
  DETERMINISTIC
  MODIFIES SQL DATA
  EXTERNAL NAME C_SALARY
  PARAMETER STYLE DB2SQL
  RETURNS NULL ON NULL CALL
  NO EXTERNAL ACTION
  NO SCRATCHPAD
  NO FINAL CALL
  ALLOW PARALLEL
  NO COLLID
  ASUTIME LIMIT 1
  STAY RESIDENT NO

```



```

PROGRAM TYPE SUB
WLM ENVIRONMENT WLMENV
SECURITY DB2
NO DBINFO;

```

After issuing the CREATE FUNCTION statement, the person who defined the function owns the user-defined function. This person (the definer) can execute the user-defined function package. In this case, the owner of the user-defined function package (the implementer) granted to the definer the EXECUTE privilege on the package that contains the user-defined function.

2. The definer grants the EXECUTE privilege on SALARY\_CHANGE to all function invokers.

```


GRANT EXECUTE ON FUNCTION SALARY_CHANGE
TO invoker1, invoker2, invoker3, invoker4

```

### 

#### Using a user-defined function:

The invoker of a user-defined function need to perform a sequence of tasks to use the user-defined function.


1.  The invoker codes an application program, named SALARY\_ADJ. The application program contains a static SQL statement that invokes the user-defined function SALARY\_CHANGE. SALARY\_CHANGE gives an employee a 10% raise if the employee is not a manager. The static SQL statement follows:

```

EXEC SQL SELECT  FIRSTNME,
                  LASTNAME
                  SALARY_CHANGE( :hvEMPNO, 10.0 )
INTO :hvFIRSTNME,
     :hvLASTNAME,
     :hvSALARY
FROM EMP
WHERE EMPNO = :hvEMPNO;

```

2. The invoker then precompiles, compiles, link-edits, and binds the invoking application's DBRM into the *invoking package*. An invoking package or invoking plan is the package or plan that contains the SQL that invokes the user-defined function. After performing these steps, the invoker is the owner of the invoking plan or package.

**Restriction:** The invoker must hold the SELECT privilege on the table EMP and the EXECUTE privilege on the function SALARY\_CHANGE. 

#### Authorization ID validation:

DB2 uses the rules for static SQL to determine the authorization ID (invoker) that executes the user-defined function package. For a static statement, the invoker is the authorization ID of the plan or package owner.

 The invoking package SALARY\_ADJ contains a static SQL SELECT statement that invokes the user-defined function SALARY\_CHANGE.

- While execution occurs in invoking package SALARY\_ADJ, DB2 uses the authorization ID of the invoker (the package owner).

The invoker requires the EXECUTE privilege on the user-defined function SALARY\_CHANGE, which the package SALARY\_ADJ invokes. Because the

user-defined function definer has the EXECUTE privilege on the user-defined function package C\_SALARY, the invoker does not require the explicit EXECUTE privilege.

- When execution changes to the user-defined function package C\_SALARY, DB2 uses the authorization ID of the implementer (the package owner). The package owner is the authorization ID with authority to execute all static SQL in the user-defined function package C\_SALARY. **GUPI**

## Authorization behaviors for dynamic SQL statements

The two key factors that influence authorization behaviors are the DYNAMICRULES value and the runtime environment of a package. The combination of the DYNAMICRULES value and the runtime environment determine the values for the dynamic SQL attributes. Those attribute values are called the *authorization behaviors*.

**GUPI** The DYNAMICRULES option on the BIND or REBIND command determines the values that apply at run time for the following dynamic SQL attributes:

- The authorization ID or role that is used to check authorization
- The qualifier that is used for unqualified objects
- The source for application programming options that DB2 uses to parse and semantically verify dynamic SQL statements

The DYNAMICRULES option also determines whether dynamic SQL statements can include GRANT, REVOKE, ALTER, CREATE, DROP, and RENAME statements.

In addition to the DYNAMICRULES value, the runtime environment of a package controls how dynamic SQL statements behave at run time. The two possible runtime environments are:

- The package runs as part of a stand-alone program.
- The package runs as a stored procedure or user-defined function package, or runs under a stored procedure or user-defined function.

A package that runs under a stored procedure or user-defined function is a package whose associated program meets one of the following conditions:

- The program is called by a stored procedure or user-defined function.
- The program is in a series of nested calls that start with a stored procedure or user-defined function. **GUPI**

### Run behavior:

DB2 processes dynamic SQL statements by using their standard attribute. These attributes are collectively called the *run behavior*.

The run behavior consists of the following attributes:

- **GUPI** DB2 uses the authorization IDs (primary, secondary and the current SQL ID) of the application process to check for authorization of dynamic SQL statements. It also checks the role in effect if running in a trusted context.
- Dynamic SQL statements use the values of application programming options that were specified during installation. The installation option USE FOR DYNAMICRULES has no effect.

- The GRANT, REVOKE, CREATE, ALTER, DROP, and RENAME statements can be executed dynamically. 

#### **Related concepts**

“Bind behavior”

“Define behavior”

“Invoke behavior” on page 94

#### **Related reference**

“Common attribute values for bind, define, and invoke behaviors” on page 94

#### **Bind behavior:**

DB2 uses the bind behavior to process dynamic SQL statements.

 The bind behavior consists of the following attributes:

- DB2 uses the authorization ID or role of the plan or package for authorization checking of dynamic SQL statements.
- Unqualified table, view, index, and alias names in dynamic SQL statements are implicitly qualified by the default schema, which is the value of the bind option QUALIFIER. If you do not specify the QUALIFIER bind option, DB2 uses the plan or package owner as the qualifier.

The values of the authorization ID or role and the qualifier for unqualified objects are the same as those that are used for embedded or static SQL statements.

- The bind behavior consists of the common attribute values for bind, define, and invoke behaviors. 

#### **Related concepts**

“Run behavior” on page 92

“Define behavior”

“Invoke behavior” on page 94

#### **Related reference**

“Common attribute values for bind, define, and invoke behaviors” on page 94

#### **Define behavior:**

When the package is run as or under a stored procedure or a user-defined function package, DB2 processes dynamic SQL statements by using the define behavior.

 The define behavior consists of the following attribute values:

- DB2 uses the authorization ID or role of the user-defined function or the stored procedure owner for authorization checking of dynamic SQL statements in the application package.
- The default qualifier for unqualified objects is the user-defined function or the stored procedure owner.
- Define behavior consists of the common attribute values for bind, define, and invoke behaviors.

When the package is run as a stand-alone program, DB2 processes dynamic SQL statements using bind behavior or run behavior, depending on the

DYNAMICRULES value specified. 

### Related concepts

“Run behavior” on page 92

“Bind behavior” on page 93

“Invoke behavior”

### Related reference

“Common attribute values for bind, define, and invoke behaviors”

### Invoke behavior:

When the package is run as, or runs under, a stored procedure or a user-defined function package, DB2 processes dynamic SQL statements by using the invoke behavior.

#### GUIP

The invoke behavior consists of the following attribute values:

- DB2 uses the authorization ID of the user-defined function or the stored procedure invoker to check the authorization for dynamic SQL statements in the application package. It uses the following rules:
  - The current SQL ID of the invoker is checked for the required authorization.
  - Secondary authorization IDs and roles that are associated with the primary authorization ID are also checked if they are needed for the required authorization.
- The default qualifier for unqualified objects is the user-defined function or the stored procedure invoker.
- Invoke behavior consists of the common attribute values for bind, define, and invoke behaviors.

When the package is run as a stand-alone program, DB2 processes dynamic SQL statements using bind behavior or run behavior, depending on the

DYNAMICRULES specified value.  GUIP

### Related concepts

“Run behavior” on page 92

“Bind behavior” on page 93

“Define behavior” on page 93


### Related reference

“Common attribute values for bind, define, and invoke behaviors”

### Common attribute values for bind, define, and invoke behaviors:

Certain attribute values apply to dynamic SQL statements in plans or packages that specify the bind, define, or invoke behavior.

The following attribute values apply:

-  GUIP You can execute the statement SET CURRENT SQLID in a package or plan that is bound with any DYNAMICRULES value. However, DB2 does not use the current SQL ID as the authorization ID for dynamic SQL statements. DB2 always uses the current SQL ID as the qualifier for the EXPLAIN output PLAN\_TABLE.
- If the value of installation option USE FOR DYNAMICRULES is YES, DB2 uses the application programming default values that were specified during installation to parse and semantically verify dynamic SQL statements. If the

value of USE for DYNAMICRULES is NO, DB2 uses the precompiler options to parse and semantically verify dynamic SQL statements.

- The GRANT, REVOKE, CREATE, ALTER, DROP, and RENAME statements cannot be executed dynamically.

The following table shows the DYNAMICRULES values and runtime environments, and the dynamic SQL behaviors that they yield.

Table 23. How DYNAMICRULES and the runtime environment determine dynamic SQL statement behavior

DYNAMICRULES value	Dynamic SQL statements in a stand-alone program environment	Dynamic SQL statements in a user-defined function or stored procedure environment
BIND	Bind behavior	Bind behavior
RUN	Run behavior	Run behavior
DEFINEBIND	Bind behavior	Define behavior
DEFINERUN	Run behavior	Define behavior
INVOKEBIND	Bind behavior	Invoke behavior
INVOKERUN	Run behavior	Invoke behavior

The following table shows the dynamic SQL attribute values for each type of dynamic SQL behavior.


Table 24. Definitions of dynamic SQL statement behaviors

Dynamic SQL attribute	Bind behavior	Run behavior	Define behavior	Invoke behavior
Authorization ID	Plan or package owner	Authorization IDs of the process and role, if applicable	User-defined function or stored procedure owner	Authorization ID of invoker <sup>1</sup>
Default qualifier for unqualified objects	Bind OWNER or QUALIFIER value	Current Schema register determines the qualifier	User-defined function or stored procedure owner	Authorization ID of invoker or role
CURRENT SQLID <sup>2</sup>	Not applicable	Applies	Not applicable	Not applicable
Source for application programming options	Determined by <i>dsnhdcp</i> <sup>3</sup> parameter DYNRULS <sup>4</sup>	Install panel DSNTIPF	Determined by <i>dsnhdcp</i> <sup>3</sup> parameter DYNRULS <sup>4</sup>	Determined by <i>dsnhdcp</i> <sup>3</sup> parameter DYNRULS <sup>4</sup>
Can execute GRANT, REVOKE, CREATE, ALTER, DROP, RENAME?	No	Yes	No	No

1. If the invoker is the primary authorization ID of the process or the current SQL ID, the following rules apply:
  - The ID or role of the invoker is checked for the required authorization.
  - Secondary authorization IDs are also checked if they are needed for the required authorization.
2. DB2 uses the current SQL ID as the authorization ID for dynamic SQL statements only for plans and packages that have DYNAMICRULES run behavior. For the other dynamic SQL behaviors, DB2 uses the authorization ID that is associated with each dynamic SQL behavior, as shown in this table.
 

The initial current SQL ID is independent of the dynamic SQL behavior. For stand-alone programs, the current SQL ID is initialized to the primary authorization ID. You can execute the SET CURRENT SQLID statement to

change the current SQL ID for packages with any dynamic SQL behavior, but DB2 uses the current SQL ID only for plans and packages with run behavior.

3. *dsnhdcp* is the application default load module. The default name is DSNHDECP.
4. The value of *dsnhdcp* parameter DYNRULS, which you specify in field USE FOR DYNAMICRULES in installation panel DSNTIPF, determines whether DB2 uses the precompiler options or the application programming defaults for dynamic SQL statements. 

#### **Related concepts**

“Run behavior” on page 92


“Bind behavior” on page 93

“Define behavior” on page 93

“Invoke behavior” on page 94

#### **Determining authorization IDs for dynamic SQL statements in routines:**

You can determine the authorization IDs under which dynamic SQL statements in routines run based on various factors. These factors include the ownership of the stored procedure or the stored procedure package.

 Suppose that A is a stored procedure and C is a program that is neither a user-defined function nor a stored procedure. Also suppose that subroutine B is called by both stored procedure A and program C. Subroutine B, which is invoked by a language call, is neither a user-defined function nor a stored procedure. AP is the package that is associated with stored procedure A, and BP is the package that is associated with subroutine B. A, B, and C execute as shown in the following diagram.

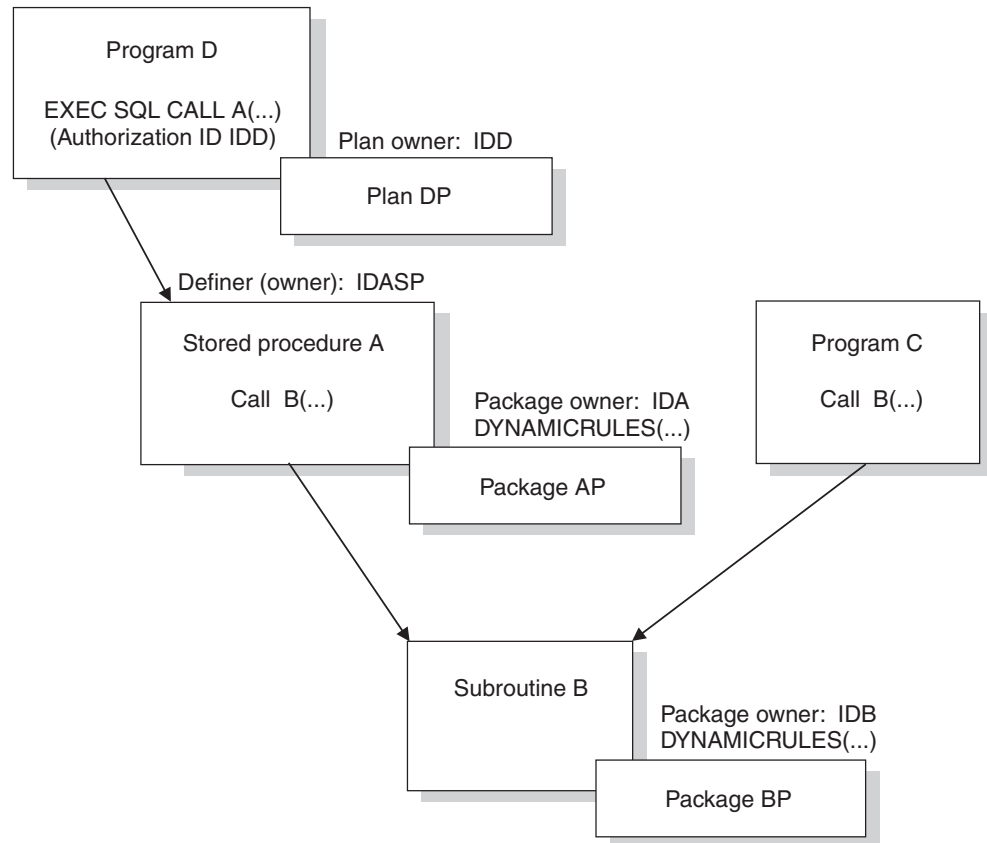


Figure 6. Authorization for dynamic SQL statements in programs and routines

Stored procedure A was defined by IDASP and is therefore owned by IDASP. The stored procedure package AP was bound by IDA and is therefore owned by IDA. Package BP was bound by IDB and is therefore owned by IDB. The authorization ID under which EXEC SQL CALL A runs is IDD, the owner of plan DP.

The authorization ID under which dynamic SQL statements in package AP run is determined in the following way:

- If package AP uses DYNAMICRULES bind behavior, the authorization ID for dynamic SQL statements in package AP is IDA, the owner of package AP.
- If package AP uses DYNAMICRULES run behavior, the authorization ID for dynamic SQL statements in package AP is the value of CURRENT SQLID when the statements execute.
- If package AP uses DYNAMICRULES define behavior, the authorization ID for dynamic SQL statements in package AP is IDASP, the definer (owner) of stored procedure A.
- If package AP uses DYNAMICRULES invoke behavior, the authorization ID for dynamic SQL statements in package AP is IDD, the invoker of stored procedure A.

The authorization ID under which dynamic SQL statements in package BP run is determined in the following way:

- If package BP uses DYNAMICRULES bind behavior, the authorization ID for dynamic SQL statements in package BP is IDB, the owner of package BP.

- If package BP uses DYNAMICRULES run behavior, the authorization ID for dynamic SQL statements in package BP is the value of CURRENT SQLID when the statements execute.
- If package BP uses DYNAMICRULES define behavior:
  - When subroutine B is called by stored procedure A, the authorization ID for dynamic SQL statements in package BP is IDASP, the definer of stored procedure A.
  - When subroutine B is called by program C:
    - If package BP uses the DYNAMICRULES option DEFINERUN, DB2 executes package BP using DYNAMICRULES run behavior, which means that the authorization ID for dynamic SQL statements in package BP is the value of CURRENT SQLID when the statements execute.
    - If package BP uses the DYNAMICRULES option DEFINEBIND, DB2 executes package BP using DYNAMICRULES bind behavior, which means that the authorization ID for dynamic SQL statements in package BP is IDB, the owner of package BP.
- If package BP uses DYNAMICRULES invoke behavior:
  - When subroutine B is called by stored procedure A, the authorization ID for dynamic SQL statements in package BP is IDD, the authorization ID under which EXEC SQL CALL A executed.
  - When subroutine B is called by program C:
    - If package BP uses the DYNAMICRULES option INVOKERUN, DB2 executes package BP using DYNAMICRULES run behavior, which means that the authorization ID for dynamic SQL statements in package BP is the value of CURRENT SQLID when the statements execute.
    - If package BP uses the DYNAMICRULES option INVOKEBIND, DB2 executes package BP using DYNAMICRULES bind behavior, which means that the authorization ID for dynamic SQL statements in package BP is IDB, the owner of package BP.

Now suppose that B is a user-defined function, as shown in the following diagram.



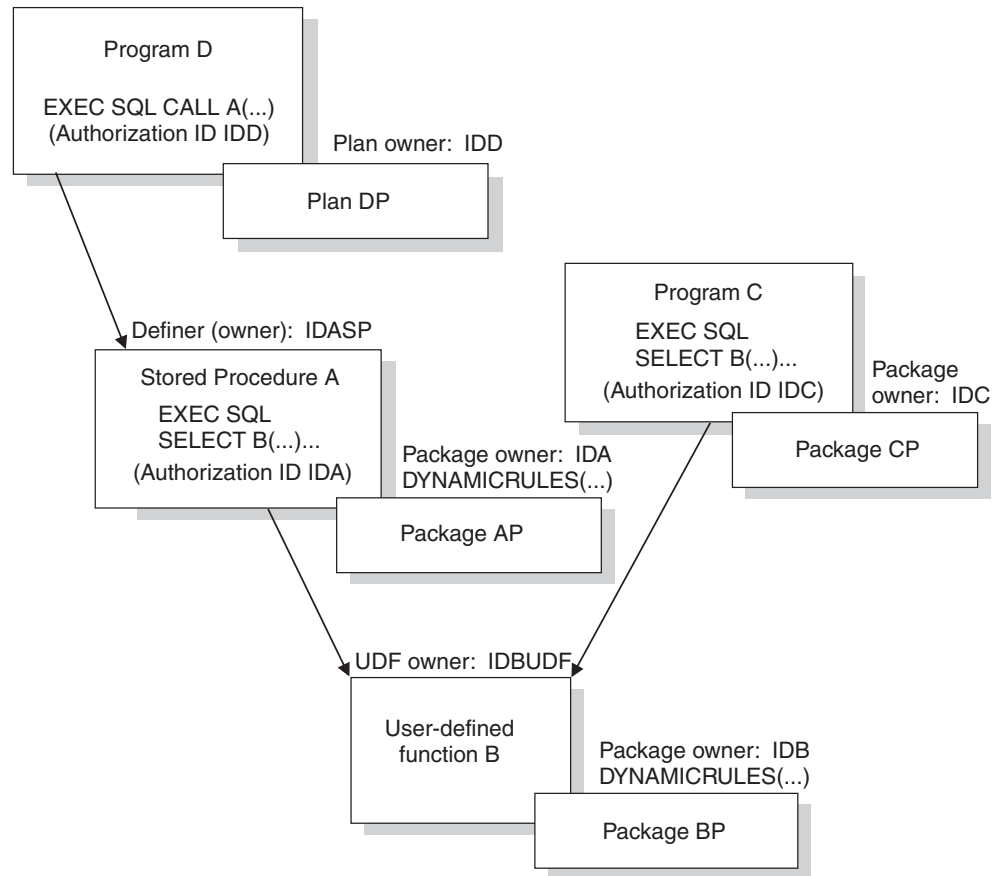


Figure 7. Authorization for dynamic SQL statements in programs and nested routines

User-defined function B was defined by IDBUDF and is therefore owned by ID IDBUDF. Stored procedure A invokes user-defined function B under authorization ID IDA. Program C invokes user-defined function B under authorization ID IDC. In both cases, the invoking SQL statement (EXEC SQL SELECT B) is static.

The authorization ID under which dynamic SQL statements in package BP run is determined in the following way:

- If package BP uses DYNAMICRULES bind behavior, the authorization ID for dynamic SQL statements in package BP is IDB, the owner of package BP.
- If package BP uses DYNAMICRULES run behavior, the authorization ID for dynamic SQL statements in package BP is the value of CURRENT SQLID when the statements execute.
- If package BP uses DYNAMICRULES define behavior, the authorization ID for dynamic SQL statements in package BP is IDBUDF, the definer of user-defined function B.
- If package BP uses DYNAMICRULES invoke behavior:
  - When user-defined function B is invoked by stored procedure A, the authorization ID for dynamic SQL statements in package BP is IDA, the authorization ID under which B is invoked in stored procedure A.
  - When user-defined function B is invoked by program C, the authorization ID for dynamic SQL statements in package BP is IDC, the owner of package CP, and is the authorization ID under which B is invoked in program C.

**GUPI**

## Related tasks

“Simplifying access authorization for routines”

“Using composite privileges”

“Performing multiple actions in one statement”

## Simplifying access authorization for routines:

You can simplify authorization for routines in several ways without violating any of the authorization standards at your installation.

**GUIP** Consider the following strategies to simplify authorization:

- Have the implementer bind the user-defined function package using DYNAMICRULES define behavior. With this behavior in effect, DB2 only needs to check the definer's ID to execute dynamic SQL statements in the routine. Otherwise, DB2 needs to check the many different IDs that invoke the user-defined function.
- If you have many different routines, group those routines into schemas. Then grant EXECUTE on the routines in the schema to the appropriate users. Users have execute authority on any functions that you add to that schema.

**Example:** To grant the EXECUTE privilege on a schema to PUBLIC, issue the following statement:

```
GRANT EXECUTE ON FUNCTION schemaname.* TO PUBLIC;
```

**GUIP**

## Related reference

“Determining authorization IDs for dynamic SQL statements in routines” on page 96

## Using composite privileges:

An SQL statement can name more than one object. A SELECT operation, for example, can join two or more tables, or an INSERT statement can use a subquery.

**GUIP** These operations require privileges on all of the tables that are included in the statement. However, you might be able to issue such a statement dynamically even though one of your IDs alone does not have all the required privileges.

If the DYNAMICRULES run behavior is in effect when the dynamic statement is prepared and your primary ID, any associated role, or any of your secondary IDs has all the needed privileges, the statement is validated. However, if you embed the same statement in a host program and try to bind it into a plan or package, the validation fails. The validation also fails for the dynamic statement if DYNAMICRULES bind, define, or invoke behavior is in effect when you issue the dynamic statement. In each case, all the required privileges must be held by the single authorization ID, determined by DYNAMICRULES behavior. **GUIP**

## Related reference

“Determining authorization IDs for dynamic SQL statements in routines” on page 96

## Performing multiple actions in one statement:

A REBIND or FREE subcommand can name more than one plan or package. If no owner is named, the set of privileges that is associated with the primary ID, the associated role, and the secondary IDs must include the BIND privilege for each object.

**GUPI** **Example:** Suppose that a user with a secondary ID of HQFINANCE has the BIND privilege on plan P1 and that another user with a secondary ID of HQHR has the BIND privilege on plan P2. Assume that someone with HQFINANCE and HQHR as secondary authorization IDs issues the following command:

```
REBIND PLAN(P1,P2)
```

P1 and P2 are successfully rebound, even though neither the HQFINANCE nor HQHR has the BIND privilege for both plans. **GUPI**

#### Related reference

“Determining authorization IDs for dynamic SQL statements in routines” on page 96

---

## Retrieving privilege records in the DB2 catalog

You can query the DB2 catalog tables by using the SQL SELECT statement. Executing the SQL statement requires appropriate privileges and authorities. You can control access to the catalog by granting and revoking these privileges and authorities.

### Catalog tables with privilege records

An authorization ID can hold different privileges. DB2 records information about the privileges of an ID in catalog tables.

**GUPI** The following catalog tables contain information about the privileges that IDs can hold:

*Table 25. Privileges information in DB2 catalog tables*

Table name	Records privileges held for or authorization related to
SYSIBM.SYSCOLAUTH	Updating columns
SYSIBM.SYSDBAUTH	Databases
SYSIBM.SYSPLANAUTH	Plans
SYSIBM.SYSPACKAUTH	Packages
SYSIBM.SYSRESAUTH	Buffer pools, storage groups, collections, table spaces, JARs, and distinct types
SYSIBM.SYSROUTINEAUTH	User-defined functions and stored procedures
SYSIBM.SYSSCHEMAAUTH	Schemas
SYSIBM.SYSTABAUTH	Tables and views
SYSIBM.SYSUSERAUTH	System authorities
SYSIBM.SYSSEQUENCEAUTH	Sequences
SYSIBM.SYSCONTEXT	Associating a role with a trusted context
SYSIBM.SYSCTXTTRUSTATTRS	Associating trust attributes with a trusted context
SYSIBM.SYSCONTEXTAUTHIDS	Associating users with a trusted context

## Retrieving all authorization IDs or roles with granted privileges

Catalog tables that contain authorization information include GRANTEE and GRANTEETYPE columns. Depending on the settings of these columns, you can modify the WHERE clause of the SELECT statement to retrieve all IDs or roles with the same privileges.

**GUPI** No single catalog table contains information about all privileges. If GRANTEETYPE is blank, the value of GRANTEE is the primary or secondary authorization ID that has been granted a privilege. If GRANTEETYPE is "L", the value of GRANTEE is a role.

To retrieve all IDs or roles with privileges, you can issue the SQL code as shown in the following example:

```
SELECT GRANTEE, 'PACKAGE ' FROM SYSIBM.SYSPACKAUTH
      WHERE GRANTEETYPE IN (' ', 'L')
UNION
SELECT GRANTEE, 'TABLE ' FROM SYSIBM.SYSTABAUTH
      WHERE GRANTEETYPE IN (' ', 'L')
UNION
SELECT GRANTEE, 'COLUMN ' FROM SYSIBM.SYSCOLAUTH
      WHERE GRANTEETYPE IN (' ', 'L')
UNION
SELECT GRANTEE, 'ROUTINE ' FROM SYSIBM.SYSROUTINEAUTH
      WHERE GRANTEETYPE IN (' ', 'L')
UNION
SELECT GRANTEE, 'PLAN ' FROM SYSIBM.SYSPLANAUTH
      WHERE GRANTEETYPE IN (' ', 'L')
UNION
SELECT GRANTEE, 'SYSTEM ' FROM SYSIBM.SYSUSERAUTH
      WHERE GRANTEETYPE IN (' ', 'L')
UNION
SELECT GRANTEE, 'DATABASE' FROM SYSIBM.SYSDBAUTH
      WHERE GRANTEETYPE IN (' ', 'L')
UNION
SELECT GRANTEE, 'SCHEMA ' FROM SYSIBM.SYSSCHEMAAUTH
      WHERE GRANTEETYPE IN (' ', 'L')
UNION
SELECT GRANTEE, 'USER ' FROM SYSIBM.SYSRESAUTH
      WHERE GRANTEETYPE IN (' ', 'L')
UNION
SELECT GRANTEE, 'SEQUENCE ' FROM SYSIBM.SYSSEQUENCEAUTH
      WHERE GRANTEETYPE IN (' ', 'L');
```

Periodically, you should compare the list of IDs or roles that is retrieved by this SQL code with the following lists:

- Lists of users from subsystems that connect to DB2 (such as IMS, CICS, and TSO)
- Lists of RACF users and groups
- Lists of users from other DBMSs that access your DB2 subsystem
- Lists of remote connections.

If DB2 lists IDs or roles that do not exist elsewhere, you should revoke their privileges. **GUPI**

## Retrieving multiple grants of the same privilege

You can query the catalog to find information about duplicate grants of the same privilege on objects. If multiple grant records clutter your catalog, consider revoking unnecessary grants, which removes duplicate grant data from the catalog.

To retrieve duplicate grants on plans:

Issue the following SQL statement:

**GUPI**

```
SELECT GRANTEE, NAME, COUNT(*)
  FROM SYSIBM.SYSPLANAUTH
 GROUP BY GRANTEE, NAME
  HAVING COUNT(*) > 2
 ORDER BY 3 DESC;
```

**GUPI**

This statement orders the duplicate grants by frequency, so that you can easily identify the most duplicated grants. Similar statements for other catalog tables can retrieve information about multiple grants on other types of objects.

If several grantors grant the same privilege to the same grantee, the DB2 catalog can become cluttered with similar data. This similar data is often unnecessary, and it might cause poor performance.

**Example 1:** Suppose that Judy, Kate, and Patti all grant the SELECT privilege on TABLE1 to Chris. If you care that Chris's ID has the privilege but not who granted the privilege, you might consider two of the SELECT grants to be redundant and unnecessary performance liabilities.

However, you might want to maintain information about authorities that are granted from several different IDs, especially when privileges are revoked.

**Example 2:** Suppose that the SELECT privilege from the previous example is revoked from Judy. If Chris has the SELECT privilege from only Judy, Chris loses the SELECT privilege. However, Chris retains the SELECT privilege because Kate and Patti also granted the SELECT privilege to Chris. In this case, the similar grants prove not to be redundant.

## Retrieving all authorization IDs or roles with the DBADM and system DBADM authorities

You can retrieve all authorization IDs or roles that have the DBADM and system DBADM authorities.

Issue the following statement to retrieve all authorization IDs or roles that have the DBADM authority:

**GUPI**

```
SELECT DISTINCT GRANTEE
  FROM SYSIBM.SYSDBAUTH
 WHERE DBADMAUTH <> ' ' AND GRANTEETYPE IN (' ','L');
```

Issue the following statement to retrieve all authorization IDs or roles that have the system DBADM authority on specific databases in the DB2 system:

```

SELECT DISTINCT GRANTEE
FROM SYSIBM.SYSUSERAUTH
WHERE SDBADMAUTH <> ' ' AND GRANTEETYPE IN (' ', 'L');

```

#### GUPI

## Retrieving all IDs or roles with access to the same table

You can retrieve all IDs or roles (GRANTEETYPE="L") that are explicitly authorized to access the same object.

#### GUPI

For example, to retrieve all IDs or roles (GRANTEETYPE="L") that are explicitly authorized to access the sample employee table (DSN81010.EMP in database DSN8D10A), issue the following statement:

```

SELECT DISTINCT GRANTEE FROM SYSIBM.SYSTABAUTH
WHERE TTNAME='EMP' AND TCREATOR='DSN8910' AND
      GRANTEETYPE IN (' ', 'L');

```

To retrieve all IDs or roles (GRANTEETYPE="L") that can change the sample employee table (IDs with administrative authorities and IDs to which authority is explicitly granted), issue the following statement:

```

SELECT DISTINCT GRANTEE FROM SYSIBM.SYSTABAUTH
WHERE TTNAME='EMP' AND
      TCREATOR='DSN8910' AND
      GRANTEETYPE IN (' ', 'L') AND
      (ALTERAUTH <> ' ' OR
       DELETEAUTH <> ' ' OR
       INSERTAUTH <> ' ' OR
       UPDATEAUTH <> ' ')
UNION
SELECT GRANTEE FROM SYSIBM.SYSUSERAUTH
WHERE SYSADMAUTH <> ' '
UNION
SELECT GRANTEE FROM SYSIBM.SYSDBAUTH
WHERE DBADMAUTH <> ' ' AND NAME='DSN8D91A';

```

To retrieve the columns of DSN81010.EMP for which update privileges have been granted on a specific set of columns, issue the following statement:

```

SELECT DISTINCT COLNAME, GRANTEE, GRANTEETYPE FROM SYSIBM.SYSCOLAUTH
WHERE CREATOR='DSN81010' AND TNAME='EMP'
ORDER BY COLNAME;

```

The character in the GRANTEETYPE column shows whether the privileges have been granted to a primary or secondary authorization ID (blank), a role (L), or are used by an application plan or package (P).

To retrieve the IDs that have been granted the privilege of updating one or more columns of DSN81010.EMP, issue the following statement:

```

SELECT DISTINCT GRANTEE FROM SYSIBM.SYSTABAUTH
WHERE TTNAME='EMP' AND
      TCREATOR='DSN8910' AND
      GRANTEETYPE IN (' ', 'L') AND
      UPDATEAUTH <> ' ';

```

The query returns only the IDs or roles (GRANTEETYPE="L") to which update privileges have been specifically granted. It does not return IDs or roles that have the privilege because of SYSADM or DBADM authority. You could include them by forming a union with additional queries, as shown in the following example:

```

SELECT DISTINCT GRANTEE GRANTEETYPE FROM SYSIBM.SYSTABAUTH
  WHERE TTNAME='EMP' AND
        TCREATOR='DSN8910' AND
        GRANTEETYPE IN (' ','L') AND
        UPDATEAUTH <> ' '
UNION
SELECT GRANTEE FROM SYSIBM.SYSUSERAUTH
  WHERE SYSADMAUTH <> ' '
UNION
SELECT GRANTEE FROM SYSIBM.SYSDBAUTH
  WHERE DBADMAUTH <> ' ' AND NAME='DSN8D91A';

```

#### GUPI

## Retrieving all IDs or roles with access to the same routine

You can retrieve the IDs or roles (GRANTEETYPE="L") that are authorized to access the same routines.

#### GUPI

For example, to retrieve the IDs or roles (GRANTEETYPE="L") that are authorized to access stored procedure PROCA in schema SCHEMA1, issue the following statement:

```

SELECT DISTINCT GRANTEE FROM SYSIBM.SYSROUTINEAUTH
  WHERE SPECIFICNAME='PROCA' AND
        SCHEMA='SCHEMA1' AND
        GRANTEETYPE IN (' ','L') AND
        ROUTINETYPE='P';

```

You can write a similar statement to retrieve the IDs or roles (GRANTEETYPE="L") that are authorized to access a user-defined function. To retrieve the IDs or roles that are authorized to access user-defined function UDFA in schema SCHEMA1, issue the following statement:

```

SELECT DISTINCT GRANTEE FROM SYSIBM.SYSROUTINEAUTH
  WHERE SPECIFICNAME='UDFA' AND
        SCHEMA='SCHEMA1' AND
        GRANTEETYPE IN (' ','L') AND
        ROUTINETYPE='F';

```

#### GUPI

## Retrieving plans or packages with access to the same table

You can retrieve all the plans or packages that are granted access to the same table.

#### GUPI

For example, to retrieve the names of application plans and packages that refer to table DSN81010.EMP directly, issue the following statement:

```

SELECT DISTINCT GRANTEE FROM SYSIBM.SYSTABAUTH
  WHERE GRANTEETYPE = 'P' AND
        TCREATOR = 'DSN81010' AND
        TTNAME = 'EMP';

```

The preceding query does not distinguish between plans and packages. To identify a package, use the COLLID column of table SYSTABAUTH, which names the collection in which a package resides and is blank for a plan.

A plan or package can refer to the table indirectly, through a view.

To find all views that refer to the table:

1. Issue the following query:

```
SELECT DISTINCT DNAME FROM SYSIBM.SYSVIEWDEP
WHERE BTYPE = 'T' AND
      BCREATOR = 'DSN81010' AND
      BNAME = 'EMP';
```

2. Write down the names of the views that satisfy the query. These values are instances of *DNAME\_list*.
3. Find all plans and packages that refer to those views by issuing a series of SQL statements. For each instance of *DNAME\_list*, issue the following statement:

```
SELECT DISTINCT GRANTEE FROM SYSIBM.SYSTABAUTH
WHERE GRANTEETYPE = 'P' AND
      TCREATOR = 'DSN81010' AND
      TTNAME = DNAME_list;
```

#### GUIP

## Retrieving privilege information through views

An ID with the SQLADM, system DBADM, DATAACCESS, ACCESSCTRL, SECADM, SYSADM, or SYSCTRL authority automatically has the privilege of retrieving data from catalog tables. If you do not want to grant the SELECT privilege on all catalog tables to PUBLIC, consider using views to let each ID retrieve information about its own privileges.

#### GUIP

For example, the following view includes the owner and the name of every table on which a user's primary authorization ID has the SELECT privilege:

```
CREATE VIEW MYSELECTS AS
SELECT TCREATOR, TTNAME FROM SYSIBM.SYSTABAUTH
WHERE SELECTAUTH <> ' ' AND
      GRANTEETYPE = ' ' AND
      GRANTEE IN (USER, 'PUBLIC', CURRENT SQLID);
```

The keyword USER in that statement is equal to the value of the primary authorization ID. To include tables that can be read by a secondary ID, set the current SQLID to that secondary ID before querying the view.

To make the view available to every ID, issue the following GRANT statement:

```
GRANT SELECT ON MYSELECTS TO PUBLIC;
```

Similar views can show other privileges. This view shows privileges over columns:

```
CREATE VIEW MYCOLS (OWNER, TNAME, CNAME, REMARKS, LABEL)
AS SELECT DISTINCT TBCREATOR, TBNAME, NAME, REMARKS, LABEL
FROM SYSIBM.SYSCOLUMNS, SYSIBM.SYSTABAUTH
WHERE TCREATOR = TBCREATOR AND
      TTNAME = TBNAME AND
      GRANTEETYPE = ' ' AND
      GRANTEE IN (USER, 'PUBLIC', CURRENT SQLID);
```

#### GUIP



---

## Implementing multilevel security with DB2

*Multilevel security* allows you to classify objects and users with security labels that are based on hierarchical security levels and non-hierarchical security categories. Multilevel security prevents unauthorized users from accessing information at a higher classification than their authorization. It also prevents users from declassifying information.

Using multilevel security with row-level granularity, you can define security for DB2 objects and perform security checks, including row-level security checks. Row-level security checks allow you to control which users have authorization to view, modify, or perform other actions on specific rows of data.

You can implement multilevel security with the following combinations:

### **DB2 authorization with multilevel security with row-level granularity**

In this combination, DB2 grants are used for authorization at the DB2 object level (database, table, and so forth). Multilevel security is implemented only at the row level within DB2.

### **External access control and multilevel security with row-level granularity**

In this combination, external access control (such as the RACF access control module) is used for authorization at the DB2 object level. External access control also uses security labels to perform mandatory access checking on DB2 objects as part of multilevel security. Multilevel security is also implemented on the row level within DB2.

**Important:** The following information about multilevel security is specific to DB2. It does not describe all aspects of multilevel security. However, this specific information assumes that you have general knowledge of multilevel security.

### **Related concepts**

“Multilevel security”

“Access control through multilevel security” on page 6

## Multilevel security

*Multilevel security* is a security policy that allows you to classify objects and users based on a system of hierarchical security levels and a system of non-hierarchical security categories.

Multilevel security provides the capability to prevent unauthorized users from accessing information at a higher classification than their authorization, and prevents users from declassifying information.

Multilevel security offers the following advantages:

- Multilevel security enforcement is mandatory and automatic.
- Multilevel security can use methods that are difficult to express through traditional SQL views or queries.
- Multilevel security does not rely on special views or database variables to provide row-level security control.
- Multilevel security controls are consistent and integrated across the system, so that you can avoid defining users and authorizations more than once.
- Multilevel security does not allow users to declassify information.

Using multilevel security, you can define security for DB2 objects and perform other checks, including row-level security checks. Row-level security checks allow you to control which users have authorization to view, modify, or perform other actions on specific rows of data.

Multilevel security and row access control are mutually exclusive. While you can activate column access control on a table that has a security label column and enforce it on a security label column, you cannot do the same with row access control. If a table has a security label column, you cannot enable it with row access control. Vice versa is true; if a table is activated with row access control, you cannot alter it to include a security label column.

#### **Related reference**

“Implementing multilevel security with DB2” on page 107

### **Security labels**

Multilevel security restricts access to an object or a row based on the security label of the object or row and the security label of the user.

For local connections, the security label of the user is the security label that the user specified during sign-on. This security label is associated with the DB2 primary authorization ID and accessed from the RACF ACEE control block. If no security label is specified during sign-on, the security label is the user's default security label.

For normal TCP/IP connections, the security label of the user can be defined by the security zone. IP addresses are grouped into security zones on the DB2 server. For trusted TCP/IP connections, the security label of the user is the security label established under the trusted context.

For SNA connections, the default security label for the user is used instead of the security label that the user signed on with.

Security labels can be assigned to a user by establishing a trusted connection within a trusted context. The trusted context definition specifies the security label that is associated with a user on the trusted connection. You can define trusted contexts if you have the SYSADM authority.

Security labels are based on security levels and security categories. You can use the Common Criteria (COMCRIT) environment's subsystem parameter to require that all tables in the subsystem are defined with security labels.

When defining security labels, do not include national characters, such as @, #, and \$. Use of these characters in security labels may cause CCSID conversion errors.

#### **Related concepts**

“Security levels” on page 109

“Security categories” on page 109

### **Determining the security label of a user**

DB2 provides several built-in session variables that contain information about the server and application process. You can obtain the value of a built-in session variable by invoking the GETVARIABLE command with the name of the built-in session variable.

One of the built-in session variables is the user's security label. You can issue the `GETVARIABLE('SYSIBM.SECLABEL')` command to obtain the security label of a user.

## Security levels

Along with security categories, hierarchical security levels are used as a basis for mandatory access-checking decisions.

When you define the security level of an object, you define the degree of sensitivity of that object. Security levels ensure that an object of a certain security level is protected from access by a user of a lower security level.

### Related concepts

“Security labels” on page 108

“Security categories”

## Security categories

Security categories are the non-hierarchical basis for mandatory access-checking decisions.

When making security decisions, mandatory access control checks whether one set of security categories includes the security categories that are defined in a second set of security categories.

### Related concepts

“Security labels” on page 108

“Security levels”

## Users and objects in multilevel security

In multilevel security, a *user* is any entity that requires access to system resources; the entity can be a human user, a stored procedure, or a batch job. An *object* is any system resource to which access must be controlled; the resource can be a data set, a table, a table row, or a command.

### Related concepts

“Global temporary tables with multilevel security”

“Materialized query tables with multilevel security” on page 110

“Constraints in a multilevel-secure environment” on page 110

“Field, edit, and validation procedures in a multilevel-secure environment” on page 110

“Triggers in a multilevel-secure environment” on page 111

## Global temporary tables with multilevel security

For a declared temporary table with a column definition, no syntax exists to specify a security label on a `DECLARE GLOBAL TEMPORARY TABLE` statement. An attempt to specify a security label results in an error.

If a `DECLARE GLOBAL TEMPORARY TABLE` statement uses a `fullselect` or a `LIKE` predicate or a `CREATE GLOBAL TEMPORARY TABLE` statement uses a `LIKE` predicate, the resulting temporary table can inherit the security label column from the referenced table or view. However, the temporary table does not inherit any security attributes on that column. That means that the inherited column in the temporary table is not defined `AS SECURITY LABEL`. The column in the temporary table is defined as `NOT NULL`, with no default. Therefore, any statements that insert data in the temporary table must provide a value for the inherited column.

### **Related concepts**

“Users and objects in multilevel security” on page 109

“Materialized query tables with multilevel security”

“Constraints in a multilevel-secure environment”

“Field, edit, and validation procedures in a multilevel-secure environment”

“Triggers in a multilevel-secure environment” on page 111

### **Materialized query tables with multilevel security**

Materialized query tables are tables that contain information that is derived and summarized from other tables.

If one or more of the source tables for a materialized query table has multilevel security with row-level granularity enabled, some additional rules apply to working with the materialized query table and the source tables.

### **Related concepts**

“Users and objects in multilevel security” on page 109

“Global temporary tables with multilevel security” on page 109

“Constraints in a multilevel-secure environment”

“Field, edit, and validation procedures in a multilevel-secure environment”

“Triggers in a multilevel-secure environment” on page 111

### **Constraints in a multilevel-secure environment**

Although a referential constraint is not allowed for the security label column, DB2 enforces referential constraints for other columns in the table that are not defined with a security label.

Constraints operate in an multilevel-secure environment in the following ways:

- A unique constraint is allowed on a security label column.
- A referential constraint is not allowed on a security label column.
- A check constraint is not allowed on a security label column.

Multilevel security with row-level checking is not enforced when DB2 checks a referential constraint.

### **Related concepts**

“Users and objects in multilevel security” on page 109

“Global temporary tables with multilevel security” on page 109

“Materialized query tables with multilevel security”

“Field, edit, and validation procedures in a multilevel-secure environment”

“Triggers in a multilevel-secure environment” on page 111

### **Field, edit, and validation procedures in a multilevel-secure environment**

In a multilevel-secure environment, field procedures, edit procedures, and validation procedures operate in certain ways.

- Field procedures are not allowed on a security label column. Edit procedures that are defined as WITH ROW ATTRIBUTES are not allowed on a table with a security label column.
- Validation procedures are allowed on a table that is defined with a security label column. When an authorized user with write-down privilege makes an INSERT or UPDATE request for a row, the validation procedure passes the new row with

the security label of the user. If the authorized user does not have write-down privilege, the security label of the row remains the same.

#### **Related concepts**

“Users and objects in multilevel security” on page 109

“Global temporary tables with multilevel security” on page 109

“Materialized query tables with multilevel security” on page 110

“Constraints in a multilevel-secure environment” on page 110

“Triggers in a multilevel-secure environment”

### **Triggers in a multilevel-secure environment**

When a transition table is generated as the result of a trigger, the security label of the table or row from the original table is not inherited by the transition table. Therefore, multilevel security with row-level checking is not enforced for transition tables and transition values.

If an ALTER TABLE statement is used to add a security label column to a table with a trigger on it, the same rules apply to the new security label column that would apply to any column that is added to the table with the trigger on it.

When a BEFORE trigger is activated, the value of the NEW transition variable that corresponds to the security label column is set to the security label of the user if either of the following criteria are met:

- Write-down control is in effect and the user does not have the write-down privilege
- The value of the security label column is not specified

#### **Related concepts**

“Users and objects in multilevel security” on page 109

“Global temporary tables with multilevel security” on page 109

“Materialized query tables with multilevel security” on page 110

“Constraints in a multilevel-secure environment” on page 110

“Field, edit, and validation procedures in a multilevel-secure environment” on page 110

## **Mandatory access checking**

Mandatory access checking evaluates dominance relationships between user security labels and object security labels and determines whether to allow certain actions based on certain rules.

- If the security label of the user dominates the security label of the object, the user can read from the object.
- If the security label of a user and the security label of the object are equivalent, the user can read from and write to the object.
- If the security label of the user dominates the security label of the object, the user cannot write to the object unless the user has the write-down RACF privilege.
- If the security label of the user is disjoint with the security label of the object, the user cannot read or write to that object.

**Exception:** IDs with the installation SYSADM authority bypass mandatory access checking at the DB2 object level because actions by Install SYSADM do not invoke the external access control exit routine (DSNX@XAC). However, multilevel security with row-level granularity is enforced for IDs with Install SYSADM authority.

After the user passes the mandatory access check, a discretionary check follows. The discretionary access check restricts access to objects based on the identity of a user, the user's role (if one exists), and the groups to which the user belongs. The discretionary access check ensures that the user is identified as having a "need to know" for the requested resource. The check is discretionary because a user with a certain access permission is capable of passing that permission to any other user.

### **Dominance relationships between security labels**

Mandatory access checking is based on the dominance relationships between user security labels and object security labels. One security label dominates another security label in certain conditions.

- The security level that defines the first security label is greater than or equal to the security level that defines the second security label.
- The set of security categories that defines one security label includes the set of security categories that defines the other security label.

Comparisons between user security labels and object security labels can result in four types of relationships:

#### **Dominant**

One security label dominates another security label when both of the following conditions are true:

- The security level that defines the first security label is greater than or equal to the security level that defines the second security label.
- The set of security categories that defines the first security label includes the set of security categories that defines the other security label.

Reading data requires that the user security label dominates the data security label.

#### **Reverse dominant**

One security label reverse dominates another security label when both of the following conditions are true:

- The security level that defines the first security label is less than or equal to the security level that defines the second security label.
- The set of security categories that defines the first security label is a subset of the security categories that defines the other security label.

#### **Equivalent**

One security label is equivalent to another security label when they are the same or have the same level and set of categories. If both dominance and reverse dominance are true for two security labels, they are equivalent. The user security label must be equivalent to the data security label to be able to read and write data without being able to write down.

#### **Disjoint**

A security label is disjoint or incompatible with another security label if incompatible security categories cause neither security label to dominate the other security label. Two security labels are disjoint when each of them has at least one category that the other does not have. Disjoint access is not allowed, even when a user is allowed to write down. If a user security label that is disjoint to the data security label issues an INSERT, UPDATE, or LOAD command, DB2 issues an error.

**Example:** Suppose that the security level "secret" for the security label HIGH is greater than the security level "sensitive" for the security label MEDIUM. Also, suppose that the security label HIGH includes the security categories Project\_A,

Project\_B, and Project\_C, and that the security label MEDIUM includes the security categories Project\_A and Project\_B. The security label HIGH dominates the security label MEDIUM because both conditions for dominance are true.

**Example:** Suppose that the security label HIGH includes the security categories Project\_A, Project\_B, and Project\_C, and that the security label MEDIUM includes the security categories Project\_A and Project\_Z. In this case, the security label HIGH does not dominate the security label MEDIUM because the set of security categories that define the security label HIGH does not contain the security category Project\_Z.

## Write-down control

*Mandatory access checking* prevents a user from declassifying information. It prevents a user from writing to an object unless the security label of the user is equivalent to or dominated by that of the object.

DB2 requires either the equivalence of the security labels or the *write-down privilege* of the user to write to DB2 objects.

**Example:** Suppose that user1 has a security label of HIGH and that row\_x has a security label of MEDIUM. Because the security label of the user and the security label of the row are not equivalent, user1 cannot write to row\_x. Therefore, write-down control prevents user1 from declassifying the information that is in row\_x.

**Example:** Suppose that user2 has a security label of MEDIUM and that row\_x has a security label of MEDIUM. Because the security label of the user and the security label of the row are equivalent, user2 can read from and write to row\_x. However, user2 cannot change the security label for row\_x unless user2 has write-down privilege. Therefore write-down control prevents user2 from declassifying the information that is in row\_x.

## Granting write-down privileges

To grant the write-down privilege, you need to define a profile and then allow users to access the profile.

To grant write-down privilege to users:

1. Issue the following RACF command to define an IRR.WRITEDOWN.BYUSER profile.

```
RDEFINE FACILITY IRR.WRITEDOWN.BYUSER UACC(NONE)
```

2. Issue the following RACF command to allow users to access the IRR.WRITEDOWN.BYUSER profile that you just created.

```
PERMIT IRR.WRITEDOWN.BYUSER ID(USRT051 USRT052 USRT054 USRT056 -  
USRT058 USRT060 USRT062 USRT064 USRT066 USRT068 USRT041) -  
ACCESS(UPDATE) CLASS(FACILITY)
```

## Implementing multilevel security at the object level

You can implement multilevel security with DB2 at the object level.

To implement multilevel security with DB2 at the object level:

1. Define security labels in RACF for all DB2 objects that require mandatory access checking by using the RDEFINE command.

Define security labels for the following RACF resource classes:

- DSNADM (administrative authorities)
- DSNR (access to DB2 subsystems)



- MDSNBP and GSNBP (buffer pools)
- MDSNCL and GDSNCL (collections)
- MDSNJR and MDSNJR (JAR)
- MDSNPN and GDSNPN (plans)
- MDSNSC and GDSNSC (schema)
- MDSNSG and GDSNSG (storage groups)
- MDSNSM and GDSNSM (system privileges)
- MDSNSP and GDSNSP (stored procedures)
- MDSNSQ and GDSNSQ (sequences)
- MDSNTB and GDSNTB (tables, views, indexes)
- MDSNTS and GDSNTS (table spaces)
- MDSNUF and GDSNUF (user-defined functions)

**Recommendation:** Define the security label SYSMULTI for DB2 subsystems that are accessed by users with different security labels and tables that require row-level granularity.

2. Specify a proper hierarchy of security labels.

In general, the security label of an object that is higher in the object hierarchy should dominate the security labels of objects that are lower in the hierarchy. RACF and DB2 do not enforce the hierarchy; they merely enforce the dominance rules that you establish.

You can use RACF to define security labels for the DB2 objects in the following object hierarchy:

- Subsystem or data sharing group
  - Database
    - Table space
      - Table
        - Column
        - Row
    - View
    - Storage group
    - Buffer pool
    - Plan
    - Collection
      - Package
    - Schema
      - Stored procedure or user-defined function
      - Java Archive (JAR)
      - Distinct type
      - Sequence

The following examples suggest dominance relationships among objects in the DB2 object hierarchy.

**Example:** A collection should dominate a package.

**Example:** A subsystem should dominate a database. That database should dominate a table space. That table space should dominate a table. That table should dominate a column.

**Example:** If a view is based on a single table, the table should dominate the view. However, if a view is based on multiple tables, the view should dominate the tables.

3. Define security labels and associate users with the security labels in RACF. If you are using a TCP/IP connection, you need to define security labels in RACF for the security zones into which IP addresses are grouped. These IP addresses represent remote users. Give users with SYSADM, SYSCTRL, and SYSOPR authority the security label of SYSHIGH.



4. Activate the SECLABEL class in RACF. If you want to enforce write-down control, turn on write-down control in RACF.
5. Install the external security access control authorization exit routine (DSNX@XAC), such as the RACF access control module.

#### Related tasks

“Implementing multilevel security with row-level granularity”

“Restricting access to the security label column” on page 117

## Implementing multilevel security with row-level granularity

Many applications need row-level security within the relational database so that access can be restricted to a specific set of rows. This security control often needs to be mandatory so that users are unable to bypass the row-level security mechanism. Using mandatory controls with z/OS and RACF provides consistency across the system.

**Requirement:** You must have z/OS Version 1 Release 5 or later to use DB2 authorization with multilevel security with row-level granularity.

You can implement multilevel security with row-level granularity with or without implementing multilevel security on the object level. If you implement multilevel security on the object level, you must define security labels in RACF for all DB2 objects and install the external security access control authorization exit routine. If you do not use the access control authorization exit routine or RACF access control, you can use DB2 native authorization control.

You can implement multilevel security with row-level granularity with or without implementing multilevel security on the object level.

**Recommendation:** Use multilevel security at the object level with multilevel security with row-level granularity. Using RACF with multilevel security provides an independent check at run time and always checks the authorization of a user to the data.

DB2 performs multilevel security with row-level granularity by comparing the security label of the user to the security label of the row that is accessed. Because security labels can be equivalent without being identical, DB2 uses the RACROUTE REQUEST=DIRAUTH macro to make this comparison when the two security labels are not the same. For read operations, such as SELECT, DB2 uses ACCESS=READ. For update operations, DB2 uses ACCESS=READWRITE.

The write-down privilege for multilevel security with row-level granularity has the following properties:

- A user with the write-down privilege can update the security label of a row to any valid value. The user can make this update independent of the user's dominance relationship with the row.
- DB2 requires that a user have the write-down privilege to perform certain utilities.
- If write-down control is not enabled, all users with valid security labels are equivalent to users with the write-down privilege.

## Related tasks

“Implementing multilevel security at the object level” on page 113

“Restricting access to the security label column” on page 117

## Creating tables with multilevel security

You can use multilevel security with row-level checking to control table access. You can do so by creating or altering a table that has a column with the AS SECURITY LABEL attribute.

**GUIP** Tables with multilevel security in effect can be dropped by using the DROP TABLE statement. Users must have a valid security label to execute CREATE TABLE, ALTER TABLE, and DROP TABLE statements on tables with multilevel security enabled.

The performance of tables that you create and alter can suffer if the security label is not included in indexes. The security label column is used whenever a table with multilevel security enabled is accessed. Therefore, the security label column should be included in indexes on the table. If you do not index the security label column, you cannot maintain index-only access.

When a user with a valid security label creates a table, the user can implement row-level security by including a security label column. The security label column can have any name, but it must be defined as CHAR(8) and NOT NULL WITH DEFAULT. It also must be defined with the AS SECURITY LABEL clause.

**Example:** To create a table that is named TABLEMLS1 and that has row-level security enabled, issue the following statement:

```
CREATE TABLE TABLEMLS1
  (EMPNO      CHAR(6)      NOT NULL,
   EMPNAME    VARCHAR(20)  NOT NULL,
   DEPTNO     VARCHAR(5)
   SECURITY    CHAR(8)      NOT NULL WITH DEFAULT AS SECURITY LABEL,
   PRIMARY KEY (EMPNO)
)
IN DSN8D71A.DSN8S71D;
```

After the user specifies the AS SECURITY LABEL clause on a column, users can indicate the security label for each row by entering values in that column. When a user creates a table and includes a security label column, SYSIBM.SYSTABLES indicates that the table has row-level security enabled. Once a user creates a table with a security label column, the security on the table cannot be disabled. The

table must be dropped and recreated to remove this protection. **GUIP**

## Adding multilevel security to existing tables

If you have a valid security label, you can implement row-level security on an existing table by adding a security label column to the table.

The security label column can have any name, but it must be defined as CHAR(8) and NOT NULL WITH DEFAULT. It also must be defined with the AS SECURITY LABEL clause.

**Example:** Suppose that the table EMP does not have row-level security enabled. To alter EMP so that it has row-level security enabled, issue the following statement:

**GUIP**

```
ALTER TABLE EMP
  ADD SECURITY CHAR(8) NOT NULL WITH DEFAULT AS SECURITY LABEL;
```

#### GUPI

After a user specifies the AS SECURITY LABEL clause on a column, row-level security is enabled on the table and cannot be disabled. The security label for existing rows in the table at the time of the alter is the same as the security label of the user that issued the ALTER TABLE statement.

**Important:** Packages and dynamic statements are invalidated when a table is altered to add a security label column.

### Removing tables with multilevel security

With valid privileges, you can drop a table that has row-level security in effect.

It is the required privilege that you have on the table, not the row-level security of the table, that determines whether or not a DROP statement succeeds. When you drop a table that has row-level security, DB2 generates an audit record.

### Caching security labels

DB2 caches security labels to improve performance when multilevel security with row-level granularity is used.

DB2 caches all security labels that are checked (successfully and unsuccessfully) during processing. At commit or rollback, the security labels are removed from the cache. If a security policy that employs multilevel security with row-level granularity requires an immediate change and long-running applications have not committed or rolled back, you might need to cancel the application.

## Restricting access to the security label column

If you do not want users to see a security label column, you can create views that do not include the column.

#### GUPI

**Example:** Suppose that the ORDER table has the following columns: ORDERNO, PRODNO, CUSTNO, SECURITY. Suppose that SECURITY is the security label column, and that you do not want users to see the SECURITY column. Use the following statement to create a view that hides the security label column from users:

```
CREATE VIEW V1 AS
  SELECT ORDERNO, PRODNO, CUSTNO FROM ORDER;
```

Alternatively, you can create views that give each user access only to the rows that include that user's security label column. To do that, retrieve the value of the SYSIBM.SECLABEL session variable, and create a view that includes only the rows that match the session variable value.

**Example:** To allow access only to the rows that match the user's security label, use the following CREATE statement:

```
CREATE VIEW V2 AS SELECT * FROM ORDER
  WHERE SECURITY=GETVARIABLE('SYSIBM.SECLABEL');
```

#### GUPI

### Related tasks

“Implementing multilevel security at the object level” on page 113

“Implementing multilevel security with row-level granularity” on page 115

## Managing data in a multilevel-secure environment

Multilevel security with row-level checking affects the results of the SELECT, INSERT, UPDATE, MERGE, DELETE, and TRUNCATE statements.

For example, row-level checking ensures that DB2 does not return rows that have a HIGH security label to a user that has a LOW security label. Users must have a valid security label to execute the SELECT, INSERT, UPDATE, MERGE, DELETE, and TRUNCATE statements.

This effect also applies to the results of the LOAD, UNLOAD, and REORG TABLESPACE utilities on tables that are enabled with multilevel security.

### Using the SELECT statement with multilevel security

When a user with a valid security label selects data from one or more tables with row-level security enabled, DB2 compares the security label of the user to the security label of each row.

#### GUIP

Results from the checking are returned according to the following rules:

- If the security label of the user dominates the security label of the row, DB2 returns the row.
- If the security label of the user does not dominate the security label of the row, DB2 does not return the data from that row, and DB2 does not generate an error report.

**Example:** Suppose that Alan has a security label of HIGH, Beth has a security label of MEDIUM, and Carlos has a security label of LOW. Suppose that DSN8910.EMP contains the data that is shown in the following table and that the SECURITY column has been declared with the AS SECURITY LABEL clause.

Table 26. Sample data from DSN8910.EMP

EMPNO	LASTNAME	WORKDEPT	SECURITY
000010	HAAS	A00	LOW
000190	BROWN	D11	HIGH
000200	JONES	D11	MEDIUM
000210	LUTZ	D11	LOW
000330	LEE	E21	MEDIUM

Now, suppose that Alan, Beth, and Carlos each submit the following SELECT statement:

```
SELECT LASTNAME
FROM EMP
ORDER BY LASTNAME;
```

Because Alan has the security label HIGH, he receives the following result:

BROWN  
HAAS  
JONES  
LEE  
LUTZ


Because Beth has the security label MEDIUM, she receives the following result:

HAAS  
JONES  
LEE  
LUTZ

Beth does not see BROWN in her result set because the row with that information has a security label of HIGH.


Because Carlos has the security label LOW, he receives the following result:

HAAS  
LUTZ

Carlos does not see BROWN, JONES, or LEE in his result set because the rows with that information have security labels that dominate Carlos's security label. Although Beth and Carlos do not receive the full result set for the query, DB2 does not return an error code to Beth or Carlos. 

### Using the INSERT statement with multilevel security

When a user with a valid security label inserts data into a table with row-level security, the security label of the row is determined according to a specific set of rules.

-  If the user has write-down privilege or write-down control is not enabled, the user can set the security label for the row to any valid security label. If the user does not specify a value for the security label, the security label of the row becomes the same as the security label of the user.
- If the user does not have write-down privilege and write-down control is enabled, the security label of the row becomes the same as the security label of the user.

**Example:** Suppose that Alan has a security label of HIGH, that Beth has a security label of MEDIUM and write-down privilege defined in RACF, and that Carlos has a security label of LOW. Write-down control is enabled.

Now, suppose that Alan, Beth, and Carlos each submit the following INSERT statement:

```
INSERT INTO DSN8910.EMP(EMPNO, LASTNAME, WORKDEPT, SECURITY)
VALUES('099990', 'SMITH', 'C01', 'MEDIUM');
```


Because Alan does not have write-down privilege, Alan cannot choose the security label of the row that he inserts. Therefore DB2 ignores the security label of MEDIUM that is specified in the statement. The security label of the row becomes HIGH because Alan's security label is HIGH.

Because Beth has write-down privilege on the table, she can specify the security label of the new row. In this case, the security label of the new row is MEDIUM. If Beth submits a similar INSERT statement that specifies a value of LOW for the security column, the security label for the row becomes LOW.

Because Carlos does not have write-down privilege, Carlos cannot choose the security label of the row that he inserts. Therefore DB2 ignores the security label of MEDIUM that is specified in the statement. The security label of the row becomes LOW because Carlos' security label is LOW.

**Considerations for INSERT from a fullselect:** For statements that insert the result of a fullselect, DB2 does not return an error code if the fullselect contains a table with a security label column. DB2 allows it if the target table does not contain a security label column while the source table contains one.

**Considerations for SELECT...FROM...INSERT statements:** If the user has write-down privilege or write-down control is not in effect, the security label of the user might not dominate the security label of the row. For statements that insert rows and select the inserted rows, the INSERT statement succeeds. However, the inserted row is not returned.

**Considerations for INSERT with subselect:** If you insert data into a table that does not have a security label column, but a subselect in the INSERT statement does include a table with a security label column, row-level checking is performed for the subselect. However, the inserted rows will not be stored with a security label column. 

## Using the UPDATE statement with multilevel security

When a user with a valid security label updates a table with row-level security enabled, DB2 compares the security label of the user to the security label of the row.

 The update proceeds according to the following rules:

- If the security label of the user and the security label of the row are equivalent, the row is updated and the value of the security label is determined by whether the user has write-down privilege:
  - If the user has write-down privilege or write-down control is not enabled, the user can set the security label of the row to any valid security label.
  - If the user does not have write-down privilege and write-down control is enabled, the security label of the row is set to the value of the security label of the user.
- If the security label of the user dominates the security label of the row, the result of the UPDATE statement is determined by whether the user has write-down privilege:
  - If the user has write-down privilege or write-down control is not enabled, the row is updated and the user can set the security label of the row to any valid security label.
  - If the user does not have write-down privilege and write-down control is enabled, the row is not updated.
- If the security label of the row dominates the security label of the user, the row is not updated.

**Example:** Suppose that Alan has a security label of HIGH, that Beth has a security label of MEDIUM and write-down privilege defined in RACF, and that Carlos has a security label of LOW. Write-down control is enabled.

Suppose that DSN8910.EMP contains the data that is shown in the following table and that the SECURITY column has been declared with the AS SECURITY LABEL clause.

*Table 27. Sample data from DSN8910.EMP*

EMPNO	LASTNAME	WORKDEPT	SECURITY
000190	BROWN	D11	HIGH
000200	JONES	D11	MEDIUM
000210	LUTZ	D11	LOW

Now, suppose that Alan, Beth, and Carlos each submit the following UPDATE statement:

```
UPDATE DSN8910.EMP
  SET DEPTNO='X55', SECURITY='MEDIUM'
  WHERE DEPTNO='D11';
```

Because Alan has a security label that dominates the rows with security labels of MEDIUM and LOW, his write-down privilege determines whether these rows are updated. Alan does not have write-down privilege, so the update fails for these rows. Because Alan has a security label that is equivalent to the security label of the row with HIGH security, the update on that row succeeds. However, the security label for that row remains HIGH because Alan does not have the write-down privilege that is required to set the security label to any value. The results of Alan's update are shown in the following table:

*Table 28. Sample data from DSN8910.EMP after Alan's update*

EMPNO	EMPNAME	DEPTNO	SECURITY
000190	BROWN	X55	HIGH
000200	JONES	D11	MEDIUM
000210	LUTZ	D11	LOW

Because Beth has a security label that dominates the row with a security label of LOW, her write-down privilege determines whether this row is updated. Beth has write-down privilege, so the update succeeds for this row and the security label for the row becomes MEDIUM. Because Beth has a security label that is equivalent to the security label of the row with MEDIUM security, the update succeeds for that row. Because the row with the security label of HIGH dominates Beth's security label, the update fails for that row. The results of Beth's update are shown in the following table:

*Table 29. Sample data from DSN8910.EMP after Beth's update*

EMPNO	EMPNAME	DEPTNO	SECURITY
000190	BROWN	D11	HIGH
000200	JONES	X55	MEDIUM
000210	LUTZ	X55	MEDIUM

Because Carlos's security label is LOW, the update fails for the rows with security labels of MEDIUM and HIGH. Because Carlos has a security label that is equivalent to the security label of the row with LOW security, the update on that row succeeds. However, the security label for that row remains LOW because



Carlos does not have the write-down privilege, which is required to set the security label to any value. The results of Carlos's update are shown in the following table:

Table 30. Sample data from DSN8910.EMP after Carlos's update

EMPNO	EMPNAME	DEPTNO	SECURITY
000190	BROWN	D11	HIGH
000200	JONES	D11	MEDIUM
000210	LUTZ	X55	LOW

**Recommendation:** To avoid failed updates, qualify the rows that you want to update with the following predicate, for the security label column SECLABEL:

WHERE SECLABEL=GETVARIABLE('SYSIBM.SECLABEL')

Using this predicate avoids failed updates because it ensures that the user's security label is equivalent to the security label of the rows that DB2 attempts to update.

**Considerations for SELECT...FROM...UPDATE statements:** If the user has write-down privilege or if the write-down control is not in effect, the security label of the user might not dominate the security label of the row. For statements that update rows and select the updated rows, the UPDATE statement succeeds.

However, the updated row is not returned. **GUPI**

## Using the MERGE statement with multilevel security

MERGE is an SQL statement that combines the conditional INSERT and UPDATE operations on a target table. Data that is not already present in the target table is inserted with the INSERT part of the MERGE statement. Data that is already present in the target table is updated with the UPDATE part of the MERGE statement.

**GUPI** Because the MERGE statement consists of the INSERT and UPDATE operations, the multilevel security rules for the INSERT operation apply to the INSERT part of the MERGE statement and the multilevel security rules for the UPDATE operation apply to the UPDATE part of the MERGE statement.


For the INSERT part of the MERGE statement, when a user with a valid security label inserts data into a table with row-level security enabled, the security label of the row is determined according to the following rules:

- If the user has write-down privilege or if the write-down control is not enabled, the user can set the security label for the row to any valid security label. If the user does not specify a value for the security label, the security label of the row becomes the same as the security label of the user.
- If the user does not have write-down privilege and if the write-down control is enabled, the security label of the row becomes the same as the security label of the user.

For the UPDATE part of the MERGE statement, when a user with a valid security label updates a table with row-level security enabled, DB2 compares the security label of the user to the security label of the row. The update proceeds according to the following rules:



- If the security label of the user and the security label of the row are equivalent, the row is updated and the value of the security label is determined by whether the user has write-down privilege:
  - If the user has write-down privilege or if the write-down control is not enabled, the user can set the security label of the row to any valid security label.
  - If the user does not have write-down privilege and if the write-down control is enabled, the security label of the row is set to the value of the security label of the user.
- If the security label of the user dominates the security label of the row, the result of the UPDATE operation is determined by whether the user has write-down privilege:
  - If the user has write-down privilege or if the write-down control is not enabled, the row is updated and the user can set the security label of the row to any valid security label.
  - If the user does not have write-down privilege and if the write-down control is enabled, the row is not updated.
- If the security label of the row dominates the security label of the user, the row is not updated.

*Considerations for SELECT...FROM...MERGE statements:* If the user has write-down privilege or if the write-down control is not in effect, the security label of the user might not dominate the security label of the row. For statements that merge rows and select the merged rows, the MERGE statement succeeds. However, the merged row is not returned. 

## Using the DELETE statement with multilevel security

When a user with a valid security label deletes data from a table with row-level security, DB2 compares the security label of the user to that of the row.

 The delete proceeds according to the following rules:

- If the security label of the user and the security label of the row are equivalent, the row is deleted.
- If the security label of the user dominates the security label of the row, the user's write-down privilege determines the result of the DELETE statement:
  - If the user has write-down privilege or write-down control is not enabled, the row is deleted.
  - If the user does not have write-down privilege and write-down control is enabled, the row is not deleted.
- If the security label of the row dominates the security label of the user, the row is not deleted.

**Example:** Suppose that Alan has a security label of HIGH, that Beth has a security label of MEDIUM and write-down privilege defined in RACF, and that Carlos has a security label of LOW. Write-down control is enabled.

Suppose that DSN8910.EMP contains the data that is shown in the following table and that the SECURITY column has been declared with the AS SECURITY LABEL clause.

Table 31. Sample data from DSN8910.EMP

EMPNO	LASTNAME	WORKDEPT	SECURITY
000190	BROWN	D11	HIGH
000200	JONES	D11	MEDIUM
000210	LUTZ	D11	LOW

Now, suppose that Alan, Beth, and Carlos each submit the following DELETE statement:

```
DELETE FROM DSN8910.EMP
WHERE DEPTNO='D11';
```

Because Alan has a security label that dominates the rows with security labels of MEDIUM and LOW, his write-down privilege determines whether these rows are deleted. Alan does not have write-down privilege, so the delete fails for these rows. Because Alan has a security label that is equivalent to the security label of the row with HIGH security, the delete on that row succeeds. The results of Alan's delete are shown in the following table:

Table 32. Sample data from DSN8910.EMP after Alan's delete

EMPNO	EMPNAME	DEPTNO	SECURITY
000200	JONES	D11	MEDIUM
000210	LUTZ	D11	LOW

Because Beth has a security label that dominates the row with a security label of LOW, her write-down privilege determines whether this row is deleted. Beth has write-down privilege, so the delete succeeds for this row. Because Beth has a security label that is equivalent to the security label of the row with MEDIUM security, the delete succeeds for that row. Because the row with the security label of HIGH dominates Beth's security label, the delete fails for that row. The results of Beth's delete are shown in the following table:

Table 33. Sample data from DSN8910.EMP after Beth's delete


EMPNO	EMPNAME	DEPTNO	SECURITY
000190	BROWN	D11	HIGH

Because Carlos's security label is LOW, the delete fails for the rows with security labels of MEDIUM and HIGH. Because Carlos has a security label that is equivalent to the security label of the row with LOW security, the delete on that row succeeds. The results of Carlos's delete are shown in the following table:

Table 34. Sample data from DSN8910.EMP after Carlos's delete

EMPNO	EMPNAME	DEPTNO	SECURITY
000190	BROWN	D11	HIGH
000200	JONES	D11	MEDIUM


**Important:** Do not omit the WHERE clause from DELETE statements. If you omit the WHERE clause from the DELETE statement, checking occurs for rows that have security labels. This checking behavior might have a negative impact on performance.

*Considerations for SELECT...FROM...DELETE statements:* If the user has write-down privilege or write-down control is not in effect, the security label of the user might not dominate the security label of the row. For statements that delete rows and select the deleted rows, the DELETE statement succeeds. However, the deleted row is not returned. 

## Using the TRUNCATE statement with multilevel security

When a user with a valid security label uses a TRUNCATE statement to delete all data from a table with row-level security enabled, DB2 compares the security label of the user to the security label of each row.

 The delete proceeds according to the following rules:

- If the security label of the user and the security label of the row are equivalent, the row is deleted.
- If the security label of the user dominates the security label of the row, the user's write-down privilege determines the result of the DELETE statement:
  - If the user has write-down privilege or write-down control is not enabled, the row is deleted.
  - If the user does not have write-down privilege and write-down control is enabled, the row is not deleted.
- If the security label of the row dominates the security label of the user, the row is not deleted.
- If the row cannot be deleted as a result of the security label verification, the TRUNCATE statement fails. 

## Using utilities with multilevel security

You need a valid security label and additional authorizations to run certain LOAD, UNLOAD, and REORG TABLESPACE jobs on tables that have multilevel security enabled. All other utilities check only for authorization to operate on the table space; they do not check for row-level authorization.

**LOAD:** You must have the write-down privilege to run LOAD REPLACE on a table space that contains a table with multilevel security enabled. In this case, you can specify the values for the security label column.

When you run LOAD RESUME, you must have the write-down privilege to specify values for the security label column. If you run a LOAD RESUME job and do not have the write-down privilege, DB2 assigns your security label as the value for each row in the security label column.

**UNLOAD:** Additional restrictions apply to UNLOAD jobs on tables that have multilevel security enabled. Each row is unloaded only if the security label of the user dominates the security label of the row. If security label of the user does not dominate the security label of the row, the row is not unloaded and DB2 does not issue an error message.

**REORG TABLESPACE:** REORG TABLESPACE jobs on tables that have multilevel security enabled have the following restrictions:

- For jobs with the UNLOAD EXTERNAL option, each row is unloaded only if the security label of the user dominates the security label of the row. If the security label of the user does not dominate the security label of the row, the row is not unloaded and DB2 does not issue an error message.

- For jobs with the DISCARD option, a qualifying row is discarded only if the user has the write-down privilege and the security label of the user dominates the security label of the row.

## Implementing multilevel security in a distributed environment

SQL statements that originate from remote requesters can participate in a multilevel secure environment if all information on the requester has the same security label and all users of the requester are permitted to that security label.

Management of multilevel security in a distributed environment requires physical control of the participating systems and careful management of the network. Managed systems must be prevented from communicating with other systems that do not have equivalent security labels.

### Configuring TCP/IP with multilevel security

A communications server IP stack that runs in a multilevel secure environment can be configured as either a restricted stack or an unrestricted stack.

**Recommendation:** Use an unrestricted stack for DB2. An unrestricted stack is configured with an ID that is defined with a security label of SYSMULTI. A single z/OS system can concurrently run a mix of restricted and unrestricted stacks. Unrestricted stacks allow DB2 to use any security label to open sockets.

All users on a TCP/IP connection have the security label that is associated with the IP address that is defined on the server. If a user requires a different security label, the user must enter through an IP address that has that security label associated with it. If you require multiple IP addresses on a remote z/OS server, a workstation, or a gateway, you can configure multiple virtual IP addresses. This strategy can increase the number of security labels that are available on a client.

Remote users that access DB2 by using a TCP/IP network connection use the security label that is associated with the RACF SERVAUTH class profile when the remote user is authenticated. Security labels are assigned to the database access thread when the DB2 server authenticates the remote server by using the RACROUTE REQUEST = VERIFY service.

If you use a trusted context for your TCP/IP connection, you can define a default security label for all users or specific security labels for individual users who use the trusted context. The security label that is defined in the trusted context overrides the one for the TCP/IP connection in RACF.

### Configuring SNA with multilevel security

Security labels are assigned to the database access thread when the DB2 server authenticates the remote server by using the RACROUTE REQUEST = VERIFY service. The service establishes a security label for the authorization ID that is associated with the database access thread.

For SNA connections, this security label is the default security label that is defined for the remote user.

---

## Chapter 3. Managing access through RACF

You can control whether a local or remote application can gain access to a specific DB2 subsystem from different environments. You can set different levels of security depending on whether the requesting application uses SNA or Transmission Control Protocol/Internet Protocol (TCP/IP) protocols to access DB2.

After the local system authenticates the incoming ID, it treats the ID like a local connection request or a local sign-on request. You can process the ID with your connection or sign-on exit routine and associate secondary authorization IDs with the ID. If you are sending a request to a remote DB2 subsystem, that subsystem can subject your request to various security checks.

You can use an external security system, such as RACF, IMS, or CICS, to authorize and authenticate a remote request before it reaches your DB2 subsystem. The discussion in the following topics assumes that you use RACF, or an equivalent system, for external access control.

---

### Establishing RACF protection for DB2

You can install and use RACF to protect your DB2 resources.

To establish RACF protection for DB2, complete the following steps in any order:

- Define DB2 resources to RACF for protection.
- Grant RACF access to the protected DB2 resources.

### Defining DB2 resources to RACF

To establish RACF protection for your DB2 subsystem, you must define your DB2 resources to RACF and authorize RACF for authentication checking.

To define your DB2 resources to RACF:

- Define the names of protected access profiles.
- Enable RACF checking for the DSNR and SERVER classes.

You can also perform the following tasks:

- Control whether two DBMSs that use VTAM® LU 6.2 can establish sessions with each other.
- Authorize IDs that are associated with stored procedures address spaces to run the appropriate attachment facility.
- Authorize the ID that is associated with the DDF address space to use z/OS UNIX System Services if you use TCP/IP.

## Related tasks

“Permitting RACF access” on page 129

“Managing authorization for stored procedures” on page 137

“Protecting connection requests that use the TCP/IP protocol” on page 146

“Establishing Kerberos authentication through RACF” on page 147

## Naming protected access profiles

The RACF resource class for DB2 is DSNR that is contained in the RACF class descriptor table. The profiles in that class help you control access to a DB2 subsystem from another environment. The environment can be IMS, CICS, the distributed data facility (DDF), TSO, CAF, or batch.

Each profile has a name of the form *subsystem.environment*, where:

- *subsystem* is the name of a DB2 subsystem, of one to four characters; for example, DSN or DB2T.
- *environment* denotes the environment, by one of the following terms:
  - MASS for IMS (including MPP, BMP, Fast Path, and DL/I batch).
  - SASS for CICS.
  - DIST for DDF.
  - RRSAP for Resource Recovery Services attachment facility. Stored procedures use RRSAP in WLM-established address spaces.
  - BATCH for all others, including TSO, CAF, batch, all utility jobs, and requests that come through the call attachment facility.

To control access, you need to define a profile, as a member of class DSNR, for every combination of subsystem and environment you want to use. For example, suppose that you want to access:

- Subsystem DSN from TSO and DDF
- Subsystem DB2P from TSO, DDF, IMS, and RRSAP
- Subsystem DB2T from TSO, DDF, CICS, and RRSAP

Then define the profiles with the following names:

```
DSN.BATCH  DSN.DIST
DB2P.BATCH DB2P.DIST DB2P.MASS DB2P.RRSAP
DB2T.BATCH DB2T.DIST DB2T.SASS DB2T.RRSAP
```

You can do that with a single RACF command, which also names an owner for the resources:

```
RDEFINE DSNR (DSN.BATCH DSN.DIST DB2P.BATCH DB2P.DIST DB2P.MASS DB2P.RRSAP
              DB2T.BATCH DB2T.DIST DB2T.SASS DB2T.RRSAP) OWNER(DB2OWNER)
```

In order to access a subsystem in a particular environment, a user must be on the access list of the corresponding profile. You add users to the access list by using the RACF **PERMIT** command. If you do not want to limit access to particular users or groups, you can give universal access to a profile with a command like this:

```
RDEFINE DSNR (DSN.BATCH) OWNER(DB2OWNER) UACC(READ)
```

## Enabling RACF checking for the DSNR and SERVER classes

You can allow RACF to check for the DSNR and SERVER classes.

You can issue the following command to enable RACF access control to check for resources in the DSNR resource class:

```
SETROPTS CLASSACT(DSNR)
```

If you are using stored procedures in a WLM-established address space, you might also need to enable RACF checking for the SERVER class.

### Enabling partner LU verification

With RACF and VTAM, you can control whether two logical units (LU) that use LU 6.2 can connect to each other.

Each member of a connecting pair must establish a profile for the other member. For example, if LUAAA and LUBBB are to connect and know each other by those LUNAMES, issue RACF commands similar to these:

```
At LUAAA: RDEFINE APPCLU netid.LUAAA.LUBBB UACC(NONE) ...
At LUBBB: RDEFINE APPCLU netid.LUBBB.LUAAA UACC(NONE) ...
```

Here, *netid* is the network ID, given by the VTAM start option NETID.

When you create those profiles with RACF, use the SESSION operand to supply:

- The VTAM password as a session key (SESSKEY suboperand)
- The maximum number of days between changes of the session key (INTERVAL suboperand)
- An indication of whether the LU pair is locked (LOCK suboperand)

Finally, to enable RACF checking for the new APPCLU resources, issue this RACF command at both LUAAA and LUBBB:

```
SETROPTS CLASSACT(APPCLU)
```

## Permitting RACF access

You must perform certain tasks in a required order to enable a process to use protected RACF resources.

To enable a process to use protected RACF resources:

1. Define RACF user IDs for DB2-started tasks
2. Add RACF groups
3. Grant users and groups access

### Related tasks

“Defining DB2 resources to RACF” on page 127

“Managing authorization for stored procedures” on page 137

“Protecting connection requests that use the TCP/IP protocol” on page 146

“Establishing Kerberos authentication through RACF” on page 147

## Defining RACF user IDs for DB2-started tasks

A DB2 subsystem provides started-task address spaces.

The following are DB2 started-task address spaces:

- *ssnm*DBM1 for database services
- *ssnm*MSTR for system services
- *ssnm*DIST for the distributed data facility
- Names for your WLM-established address spaces for stored procedures

You must associate each of these address spaces with a RACF user ID. You can also assign each of them to a RACF group name. The RACF user IDs and group names that are associated with DB2 address spaces are listed in the following table:



Table 35. DB2 address spaces and associated RACF user IDs and group names

Address Space	RACF User ID	RACF Group Name
DSNMSTR	SYSDSP	DB2SYS
DSNDBM1	SYSDSP	DB2SYS
DSNDIST	SYSDSP	DB2SYS
DSNWLM	SYSDSP	DB2SYS
DB2TMSTR	SYSDSPT	DB2TEST
DB2TDBM1	SYSDSPT	DB2TEST
DB2TDIST	SYSDSPT	DB2TEST
DB2TSPAS	SYSDSPT	DB2TEST
DB2PMSTR	SYSDSPD	DB2PROD
DB2PDBM1	SYSDSPD	DB2PROD
DB2PDIST	SYSDSPD	DB2PROD
CICSSYS	CICS	CICSGRP
IMSCNTL	IMS	IMSGRP

You can use one of the two ways that RACF provides to associate user IDs and groups with started tasks: the STARTED class and the started procedures table (ICHRIN03). If you use the STARTED class, the changes take effect without a subsequent IPL. If you use ICHRIN03, you must perform another IPL for the changes to take effect. You cannot start the DB2 address spaces with batch jobs.

If you have IMS or CICS applications issuing DB2 SQL requests, you must associate RACF user IDs, and can associate group names, with:

- The IMS control region
- The CICS address space
- The four DB2 address spaces

If the IMS and CICS address spaces are started as batch jobs, provide their RACF IDs and group names with the USER and GROUP parameters on the JOB statement. If they are started as started-tasks, assign the IDs and group names as you do for the DB2 address spaces, by changing the RACF STARTED class or the RACF started procedures table.

The RACF user ID and group name do not need to match those that are used for the DB2 address spaces, but they must be authorized to run the Resource Recovery Services attachment facility (for WLM-established stored procedures address spaces). Note that the WLM-established stored procedures started tasks IDs require an OMVS segment.

If your installation has implemented the RACF STARTED class, you can use it to associate RACF user IDs and group names with the DB2 started procedures address spaces. If you have not previously set up the STARTED class, you first need to enable generic profile checking for the class:

```
SETROPTS GENERIC(STARTED)
```

Then, you need to define the RACF identities for the DB2 started tasks:



```

RDEFINE STARTED DSNMSTR.** STDATA(USER(SYSDP) GROUP(DB2SYS) TRUSTED(NO))
RDEFINE STARTED DSNDM1.** STDATA(USER(SYSDP) GROUP(DB2SYS) TRUSTED(NO))
RDEFINE STARTED DSNDIST.** STDATA(USER(SYSDP) GROUP(DB2SYS) TRUSTED(NO))
RDEFINE STARTED DSNWLM.** STDATA(USER(SYSDP) GROUP(DB2SYS) TRUSTED(NO))
RDEFINE STARTED DB2TMSTR.** STDATA(USER(SYSDSPT) GROUP(DB2TEST) TRUSTED(NO))
...

```

Then, you need to activate the RACLIST processing to read the profiles into a data space:

```

SETROPTS CLASSACT(STARTED)
SETROPTS RACLIST(STARTED)

```

Lastly, you need to refresh the in-storage profiles:

```

SETROPTS RACLIST(STARTED) REFRESH

```

If you use the RACF-started procedures table (ICHRIN03) to associate RACF user IDs and group names with the DB2 started procedures address spaces, you need to change, reassemble, and link edit the resulting object code to z/OS. The following example shows a sample job that reassembles and link edits the RACF started-procedures table (ICHRIN03):

```

/**
/** REASSEMBLE AND LINKEDIT THE RACF STARTED-PROCEDURES
/** TABLE ICHRIN03 TO INCLUDE USERIDS AND GROUP NAMES
/** FOR EACH DB2 CATALOGED PROCEDURE. OPTIONALLY, ENTRIES
/** FOR AN IMS OR CICS SYSTEM MIGHT BE INCLUDED.
/**
/** AN IPL WITH A CLPA (OR AN MLPA SPECIFYING THE LOAD
/** MODULE) IS REQUIRED FOR THESE CHANGES TO TAKE EFFECT.
/**

ENTCOUNT DC    AL2(((ENDTABLE-BEGTABLE)/ENTLNTH)+32768)
*              NUMBER OF ENTRIES AND INDICATE RACF FORMAT
*
* PROVIDE FOUR ENTRIES FOR EACH DB2 SUBSYSTEM NAME.
*
BEGTABLE DS      0H
* ENTRIES FOR SUBSYSTEM NAME "DSN"
      DC CL8'DSNMSTR'      SYSTEM SERVICES PROCEDURE
      DC CL8'SYSDSP'      USERID
      DC CL8'DB2SYS'      GROUP NAME
      DC X'00'            NO PRIVILEGED ATTRIBUTE
      DC XL7'00'          RESERVED BYTES
ENTLNTH EQU *-BEGTABLE  CALCULATE LENGTH OF EACH ENTRY
      DC CL8'DSNDM1'      DATABASE SERVICES PROCEDURE
      DC CL8'SYSDSP'      USERID
      DC CL8'DB2SYS'      GROUP NAME
      DC X'00'            NO PRIVILEGED ATTRIBUTE
      DC XL7'00'          RESERVED BYTES
      DC CL8'DSNDIST'     DDF PROCEDURE
      DC CL8'SYSDSP'      USERID
      DC CL8'DB2SYS'      GROUP NAME
      DC X'00'            NO PRIVILEGED ATTRIBUTE
      DC XL7'00'          RESERVED BYTES
      DC CL8'SYSDSP'      USERID
      DC CL8'DB2SYS'      GROUP NAME
      DC X'00'            NO PRIVILEGED ATTRIBUTE
      DC XL7'00'          RESERVED BYTES
      DC CL8'DSNWLM'      WLM-ESTABLISHED S.P. ADDRESS SPACE
      DC CL8'SYSDSP'      USERID
      DC CL8'DB2SYS'      GROUP NAME
      DC X'00'            NO PRIVILEGED ATTRIBUTE
      DC XL7'00'          RESERVED BYTES
* ENTRIES FOR SUBSYSTEM NAME "DB2T"
      DC CL8'DB2TMSTR'    SYSTEM SERVICES PROCEDURE

```

```

DC      CL8'SYSDSPT'      USERID
DC      CL8'DB2TEST'      GROUP NAME
DC      X'00'             NO PRIVILEGED ATTRIBUTE
DC      XL7'00'           RESERVED BYTES
DC      CL8'DB2TDBM1'     DATABASE SERVICES PROCEDURE
DC      CL8'SYSDSPT'      USERID
DC      CL8'DB2TEST'      GROUP NAME
DC      X'00'             NO PRIVILEGED ATTRIBUTE
DC      XL7'00'           RESERVED BYTES
DC      CL8'DB2TDIST'     DDF PROCEDURE
DC      CL8'SYSDSPT'      USERID
DC      CL8'DB2TEST'      GROUP NAME
DC      X'00'             NO PRIVILEGED ATTRIBUTE
DC      XL7'00'           RESERVED BYTES
DC      CL8'SYSDSPT'      USERID
DC      CL8'DB2TEST'      GROUP NAME
DC      X'00'             NO PRIVILEGED ATTRIBUTE
DC      XL7'00'           RESERVED BYTES
*      ENTRIES FOR SUBSYSTEM NAME "DB2P"
DC      CL8'DB2PMSTR'     SYSTEM SERVICES PROCEDURE
DC      CL8'SYSDSPD'      USERID
DC      CL8'DB2PROD'      GROUP NAME
DC      X'00'             NO PRIVILEGED ATTRIBUTE
DC      XL7'00'           RESERVED BYTES
DC      CL8'DB2PDBM1'     DATABASE SERVICES PROCEDURE
DC      CL8'SYSDSPD'      USERID
DC      CL8'DB2PROD'      GROUP NAME
DC      X'00'             NO PRIVILEGED ATTRIBUTE
DC      XL7'00'           RESERVED BYTES
DC      CL8'DB2PDIST'     DDF PROCEDURE
DC      CL8'SYSDSPD'      USERID
DC      CL8'DB2PROD'      GROUP NAME
DC      X'00'             NO PRIVILEGED ATTRIBUTE
DC      XL7'00'           RESERVED BYTES
DC      CL8'SYSDSPD'      USERID
DC      CL8'DB2PROD'      GROUP NAME
DC      X'00'             NO PRIVILEGED ATTRIBUTE
DC      XL7'00'           RESERVED BYTES
*      OPTIONAL ENTRIES FOR CICS AND IMS CONTROL REGION
DC      CL8'CICSSYS'      CICS PROCEDURE NAME
DC      CL8'CICS'         USERID
DC      CL8'CICSGRP'      GROUP NAME
DC      X'00'             NO PRIVILEGED ATTRIBUTE
DC      XL7'00'           RESERVED BYTES
DC      CL8'IMSCNTL'      IMS CONTROL REGION PROCEDURE
DC      CL8'IMS'          USERID
DC      CL8'IMSGRP'       GROUP NAME
DC      X'00'             NO PRIVILEGED ATTRIBUTE
DC      XL7'00'           RESERVED BYTES
ENDTABLE DS      0D
END

```

The example shows the sample entries for three DB2 subsystems (DSN, DB2T, and DB2P), optional entries for CICS and IMS, and DB2 started tasks for the DB2 subsystems, CICS, the IMS control region.

### Related tasks

“Adding RACF groups”

“Granting users and groups access” on page 133

## Adding RACF groups

You can issue the **ADDGROUP** command to add a new RACF group.

You need first to issue the following **ADDUSER** command to add user DB2OWNER and give it class authorization for DSNR and USER.

```
ADDUSER DB2OWNER CLAUTH(DSNR USER) UACC(NONE)
```

DB2OWNER can now add users to RACF and issue the RDEFINE command to define resources in class DSNR. It also has control over and responsibility for the entire DB2 security plan in RACF.

To add group DB2 to the existing SYS1 group and make DB2OWNER the owner of the new group, issue the following RACF command:

```
ADDGROUP DB2 SUPGROUP(SYS1) OWNER(DB2OWNER)
```

To connect DB2OWNER to group DB2 with the authority to create new subgroups, add users, and manipulate profiles, issue the following RACF command:

```
CONNECT DB2OWNER GROUP(DB2) AUTHORITY(JOIN) UACC(NONE)
```

To make DB2 the default group for commands issued by DB2OWNER, issue the following RACF command:

```
ALTUSER DB2OWNER DFLTGRP(DB2)
```

To create the group DB2USER and add five users to it, issue the following RACF commands:

```
ADDGROUP DB2USER SUPGROUP(DB2)  
ADDUSER (USER1 USER2 USER3 USER4 USER5) DFLTGRP(DB2USER)
```

To define a user to RACF, use the RACF ADDUSER command. That invalidates the current password. You can then log on as a TSO user to change the password.

#### *DB2 considerations when using RACF groups:*

- When a user is newly connected to, or disconnected from, a RACF group, the change is not effective until the next logon. Therefore, before using a new group name as a secondary authorization ID, a TSO user must log off and log on, or a CICS or IMS user must sign on again.
- A user with the SPECIAL, JOIN, or GROUP-SPECIAL RACF attribute can define new groups with any name that RACF accepts and can connect any user to them. Because the group name can become a secondary authorization ID, you should control the use of those RACF attributes.
- Existing RACF group names can duplicate existing DB2 authorization IDs. That duplication is unlikely for the following reasons:
  - A group name cannot be the same as a user name.
  - Authorization IDs that are known to DB2 are usually known to RACF.

However, you can create a table with an owner name that is the same as a RACF group name and use the IBM-supplied sample connection exit routine. Then any TSO user with the group name as a secondary ID has ownership privileges on the table. You can prevent that situation by designing the connection exit routine to stop unwanted group names from being passed to DB2.

#### **Related tasks**

“Defining RACF user IDs for DB2-started tasks” on page 129

“Granting users and groups access”

### **Granting users and groups access**

You can use the PERMIT command to grant users or groups access to resources in class DSNR.

Suppose that the DB2OWNER group in the following example is authorized for class DSNR, owns the profiles, and has the right to change them. You can issue the following commands to authorize the DB2USER members, the system administrators, and operators to be TSO users.

These users can run batch jobs and DB2 utilities on the three systems: DSN, DB2P, and DB2T. The ACCESS(READ) operand allows use of DB2 without the ability to manipulate profiles.

```
PERMIT DSN.BATCH CLASS(DSNR) ID(DB2USER) ACCESS(READ)
PERMIT DB2P.BATCH CLASS(DSNR) ID(DB2USER) ACCESS(READ)
PERMIT DB2T.BATCH CLASS(DSNR) ID(DB2USER) ACCESS(READ)
```

**Defining profiles for IMS and CICS:** You want the IDs for attaching systems to use the appropriate access profile. For example, to let the IMS user ID use the access profile for IMS on system DB2P, issue the following RACF command:

```
PERMIT DB2P.MASS CLASS(DSNR) ID(IMS) ACCESS(READ)
```

To let the CICS group ID use the access profile for CICS on system DB2T, issue the following RACF command:

```
PERMIT DB2T.SASS CLASS(DSNR) ID(CICSGRP) ACCESS(READ)
```

**Providing installation authorities to default IDs:** When DB2 is installed, IDs are named to have special authorities—one or two IDs for SYSADM and one or two IDs for SYSOPR. Those IDs can be connected to the group DB2USER; if they are not, you need to give them access. The next command permits the default IDs for the SYSADM and SYSOPR authorities to use subsystem DSN through TSO:

```
PERMIT DSN.BATCH CLASS(DSNR) ID(SYSADM,SYSOPR) ACCESS(READ)
```

**Using secondary IDs:** You can use secondary authorization IDs to define a RACF group. After you define the RACF group, you can assign privileges to it that are shared by multiple primary IDs. For example, suppose that DB2OWNER wants to create a group GROUP1 and to give the ID USER1 administrative authority over the group. USER1 should be able to connect other existing users to the group. To create the group, DB2OWNER issues this RACF command:

```
ADDGROUP GROUP1 OWNER(USER1) DATA('GROUP FOR DEPT. G1')
```

To let the group connect to the DSN system through TSO, DB2OWNER issues this RACF command:

```
PERMIT DSN.BATCH CLASS(DSNR) ID(GROUP1) ACCESS(READ)
```

USER1 can now connect other existing IDs to the group GROUP1 by using the RACF CONNECT command:

```
CONNECT (USER2 EPSILON1 EPSILON2) GROUP(GROUP1)
```

If you add or update secondary IDs for CICS transactions, you must start and stop the CICS attachment facility to ensure that all threads sign on and get the correct security information.

**Allowing users to create data sets:** You can use RACF to protect the data sets that store DB2 data. If you use the approach and when you create a new group of DB2 users, you might want to connect it to a group that can create data sets. To allow USER1 to create and control data sets, DB2OWNER creates a generic profile and permits complete control to USER1 and to the four administrators. The SYSDSP parameter also gives control to DB2.

```

ADDSO  'DSNC100.DSNNBC.ST*' UACC(NONE)

PERMIT 'DSNC100.DSNNBC.ST*'
      ID(USER1 SYSOSP SYSAD1 SYSAD2 SYSOP1 SYSOP2) ACCESS(ALTER)

```

### Related tasks

“Defining RACF user IDs for DB2-started tasks” on page 129

“Adding RACF groups” on page 132

## Granting authorization on DB2 commands

IDs must be authorized to issue DB2 commands. If you authorize IDs by issuing DB2 GRANT statements, the GRANT statements must be made to a primary authorization ID, a secondary authorization ID, a role, or PUBLIC.

When RACF is used for access control, an ID must have appropriate RACF authorization on DB2 commands or must be granted authorization for DB2 commands to issue commands from a logged-on MVS console or from TSO SDSF.

You can ensure that an ID can issue DB2 commands from logged-on MVS consoles or TSO SDSF by using one of the following methods:

- Grant authorization for DB2 commands to the primary, secondary authorization ID, or role.
- Define RACF classes and permits for DB2 commands.
- Grant SYSOPR authority to appropriate IDs.

## Permitting access from remote requesters

You can use the DSNR RACF class to access the distributed data address space and to control access from remote requesters.

The following RACF commands let the users in the group DB2USER access DDF on the DSN subsystem. These DDF requests can originate from any partner in the network.

**Example:** To permit READ access on profile DSN.DIST in the DSNR class to DB2USER, issue the following RACF command:

```
PERMIT DSN.DIST CLASS(DSNR) ID(DB2USER) ACCESS(READ)
```

If you want to ensure that a specific user can access only when the request originates from a specific LU name, you can use WHEN(APPCPORT) on the PERMIT command.

**Example:** To permit access to DB2 distributed processing on subsystem DSN when the request comes from USER5 at LUNAME equal to NEWYORK, issue the following RACF command:

```
PERMIT DSN.DIST CLASS(DSNR) ID(USER5) ACCESS(READ) +
      WHEN(APPCPORT(NEWYORK))
```

For connections that come through TCP/IP, use the RACF APPCPORT class or the RACF SERVAUTH class with TCP/IP Network Access Control to protect unauthorized access to DB2.

**Example:** To use the RACF APPCPORT class, perform the following steps:

1. Activate the ACCPORT class by issuing the following RACF command:

```
SETOPTS CLASSACT(APPCPORT) REFRESH
```

2. Define the general resource profile and name it TCPIP. Specify NONE for universal access and APPCPORT for class. Issue the following RACF command:  
RDEFINE APPCPORT (TCPIP) UACC(NONE)
3. Permit READ access on profile TCPIP in the APPCPORT class. To permit READ access to USER5, issue the following RACF command:  
PERMIT TCPIP ACCESS(READ) CLASS(APPCPORT) ID(USER5)
4. Permit READ access on profile DSN.DIST in the DSNR class. To permit READ access to USER5, issue the following RACF command:  
PERMIT DSN.DIST CLASS(DSNR) ID(USER5) ACCESS(READ) +  
WHEN(APPCPORT(TCPIP))
5. Refresh the APPCPORT class by issuing the following RACF command:  
SETROPTS CLASSACT(APPCPORT) REFRESH RACLIST(APPCPORT)

If the RACF APPCPORT class is active on your system, and a resource profile for the requesting LU name already exists, you must permit READ access to the APPCPORT resource profile for the user IDs that DB2 uses. You must permit READ access even when you are using the DSNR resource class. Similarly, if you are using the RACF APPL class and that class restricts access to the local DB2 LU name or generic LU name, you must permit READ access to the APPL resource for the user IDs that DB2 uses.

**Recommendation:** Use z/OS Communications Server IP Network Access Control and z/OS Security Server RACF SERVAUTH class if you want to use the port of entry (POE) for remote TCP/IP connections.

**Requirement:** To use the RACF SERVAUTH class and TCP/IP Network Access Control, you must have z/OS V1.5 (or later) installed.

**Example:** To use the RACF SERVAUTH class and TCP/IP Network Access Control, perform the following steps:

1. Set up and configure TCP/IP Network Access Control by using the NETACCESS statement that is in your TCP/IP profile.

For example, suppose that you need to allow z/OS system access only to IP addresses from 9.0.0.0 to 9.255.255.255. You want to define these IP addresses as a security zone, and you want to name the security zone IBM. Suppose also that you need to deny access to all IP addressed outside of the IBM security zone, and that you want to define these IP addresses as a separate security zone. You want to name this second security zone WORLD. To establish these security zones, use the following NETACCESS clause:

```
NETACCESS INBOUND OUTBOUND
; NETWORK/MASK      SAF
  9.0.0.0/8          IBM
  DEFAULT            WORLD
ENDNETACCESS
```

Now, suppose that USER5 has an IP address of 9.1.2.3. TCP/IP Network Access Control would determine that USER5 has an IP address that belongs to the IBM security zone. USER5 would be granted access to the system. Alternatively, suppose that USER6 has an IP address of 1.1.1.1. TCP/IP Network Access Control would determine that USER6 has an IP address that belongs to the WORLD security zone. USER6 would not be granted access to the system.

2. Activate the SERVAUTH class by issuing the following TSO command:  
SETROPTS CLASSACT(SERVAUTH)
3. Activate RACLIST processing for the SERVAUTH class by issuing the following TSO command:

SETROPTS RACLIST(SERVAUTH)

4. Define the IBM and WORLD general resource profiles in RACF to protect the IBM and WORLD security zones by issuing the following commands:

```
RDEFINE SERVAUTH (EZB.NETACCESS.ZOSV1R5.TCPIP.IBM) UACC(NONE)
RDEFINE SERVAUTH (EZB.NETACCESS.ZOSV1R5.TCPIP.WORLD) UACC(NONE)
```

5. Permit USER5 and SYSDSP read access to the IBM profile by using the following commands.

```
PERMIT EZB.NETACCESS.ZOSV1R5.TCPIP.IBM ACCESS READ CLASS(SERVAUTH) ID(USER5)
PERMIT EZB.NETACCESS.ZOSV1R5.TCPIP.IBM ACCESS READ CLASS(SERVAUTH) ID(SYSDSP)
```

6. Permit SYSDSP read access to the WORLD profile by using the following command:

```
PERMIT EZB.NETACCESS.ZOSV1R5.TCPIP.WORLD ACCESS READ CLASS(SERVAUTH) ID(USER5)
```

7. For these permissions to take effect, refresh the RACF database by using the following command:

```
SETROPTS CLASSACT(SERVAUTH) REFRESH RACLIST(SERVAUTH)
```

## Enabling IMS transactions to use RACF authorization control of DB2 objects

You can enable IMS transactions to use RACF authorization control of DB2 objects and other resources.

To enable IMS transactions to exploit RACF authorization of DB2 objects and resources:

1. Configure IMS to use APPC/OTMA security FULL or create an IMS Build Security Environment exit routine (DFSBSEX0). Code DFSBSEX0 to return RC4 in register 15, which will instruct IMS to create the ACEE in the dependent region.
2. Install the default DB2 exit routine DSNX@XAC.
3. Define a RACF profile for each DB2 object and resource to be accessed by IMS transactions.
4. Issue the RACF PERMIT command to authorize IMS transaction authorization IDs that are allowed to access these DB2 objects and resources.

### Related concepts

 The default DB2 exit routine (RACF Access Control Module Guide)

### Related information

IMS build security environment exit routine

Administering APPC/IMS

## Managing authorization for stored procedures

DB2 for z/OS provides a variety of methods to help you ensure that users are properly authorized to create and execute stored procedures. DB2 also provides ways for you to keep stored procedures secure.

- “Authorizing IDs for using RRSAP” on page 138
- “Specifying WLM-established server address spaces for stored procedures” on page 138
- “Managing authorizations for creation of stored procedures in WLM environments” on page 139
- “Authorizing users to refresh WLM environments” on page 140
- “Controlling stored procedure access to non-DB2 resources by using RACF” on page 140



- “Granting the CREATEIN privilege on schemas for stored procedures” on page 141
- “Granting privileges for using distinct types” on page 142
- “Granting privileges for using JAR files” on page 143
- “Granting privileges for executing stored procedures and stored procedure packages” on page 143
- “Controlling remote execution of stored procedures by using trusted contexts” on page 144


## Authorizing IDs for using RRSF

When started, WLM-established address spaces use the Resource Recovery Services attachment facility (RRSAF) to attach to DB2. You must authorize the IDs that are associated with WLM-established stored procedures address spaces so that they can use RRSF.

To authorize user IDs that are associated with WLM-established stored procedures address spaces to use RRSF:

1. Create a *ssnm*.RRSAF profile in RACF. For example, you can define *ssnm*.RRSAF in the DSNR resource class with a universal access authority of NONE by issuing the following command:  
RDEFINE DSNR (DB2P.RRSAF DB2T.RRSAF) UACC(NONE)
2. Refresh the in-storage profiles with the profile that you just defined. For example, you can issue the following command:  
SETOPTS RACLIST(DSNR) REFRESH
3. Add user IDs that are associated with WLM-established stored procedures address spaces to the RACF-started procedures table. For example, you can issue the following command:  
RDEFINE STARTED DSNWLM.\*\* STDATA(USER(SYSDP) GROUP(DB2SYS) TRUSTED(NO))
4. Refresh the in-storage profiles. For example, you can issue the following command:  
SETOPTS RACLIST(STARTED) REFRESH
5. Grant read access to *ssnm*.RRSAF to the IDs that are associated with the stored procedures address spaces. For example, you can issue the following command:  
PERMIT DB2P.RRSAF CLASS(DSNR) ID(SYSDSP) ACCESS(READ)

## Related information

 Summary of RACF commands (CICS Transaction Server for z/OS)

## Specifying WLM-established server address spaces for stored procedures

You can manage access to WLM through the server resource class and specify address spaces as WLM-established server address spaces for running stored procedures.

To specify address spaces as WLM-established server address spaces that can run stored procedures:

1. Define a new SERVER class by using the server resource class.  
If you do not define a SERVER class, any address space that connects to WLM as a server address space can run stored procedures.
2. Authorize a RACF profile to associate with the SERVER class. For example:  
RDEFINE SERVER (DB2.ssnm.applenv)



In this command, *applenv* is the name of the application environment that is associated with the stored procedure. For example, assume that you want to define the following profile names:

- DB2.DB2T.TESTPROC
- DB2.DB2P.PAYROLL
- DB2.DB2P.QUERY

To define these profile names, use the following RACF command:


```
RDEFINE SERVER (DB2.DB2T.TESTPROC DB2.DB2P.PAYROLL DB2.DB2P.QUERY)
```

3. Activate the resource class. For example, you can issue the following command:  
SETROPTS RACLIST(SERVER) REFRESH

4. Grant read access to the user IDs that are associated with the stored procedures address space. For example, you can issue the following commands:

```
PERMIT DB2.DB2T.TESTPROC CLASS(SERVER) ID(SYSDSP) ACCESS(READ)
PERMIT DB2.DB2P.PAYROLL CLASS(SERVER) ID(SYSDSP) ACCESS(READ)
PERMIT DB2.DB2P.QUERY CLASS(SERVER) ID(SYSDSP) ACCESS(READ)
```

#### Related information

 Summary of RACF commands (CICS Transaction Server for z/OS)

### Managing authorizations for creation of stored procedures in WLM environments

You can group and isolate applications into different WLM environments based on their security requirements. You can then authorize or prevent users from creating stored procedures in a security-sensitive environment.

DB2 invokes RACF to determine if users are allowed to create or run stored procedures in a WLM environment. The WLM ENVIRONMENT keyword on the CREATE PROCEDURE statement identifies the WLM environments that are used for running stored procedures. Attempts fail when unauthorized users try to create or run stored procedures.

To manage authorizations of users for creating stored procedures in a specific WLM environment:

Use RACF commands to manage authorizations for individual users or groups in the creation of stored procedures in a specific WLM environment:

- To authorize individual users or groups of users to create stored procedures in a specific WLM environment, issue the RACF PERMIT command. For example, you can authorize the user whose ID is DB2USER1 to create stored procedures on the DB2 subsystem DB2A (non-data sharing) in a WLM environment named PAYROLL:

```
PERMIT DB2A.WLMENV.PAYROLL CLASS(DSNR) ID(DB2USER1) ACCESS(READ)
```


When user ID DB2USER1 attempts to create a stored procedure in the PAYROLL WLM environment, DB2 performs a resource authorization check by using the DSNR RACF class and grants permission.

- To prevent users on a particular DB2 subsystem from creating stored procedures, issue the RACF **DEFINE** command. You can also use this command to revoke the default universal access of a WLM environment and set it to NONE,

For example, you can issue the following command to prevent all users on DB2 subsystem DB2A (non-data sharing) from creating stored procedures or user-defined functions in the WLM environment named PAYROLL:

```
RDEFINE DSNR (DB2A.WLMENV.PAYROLL) UACC(NONE)
```

## Related information

 Summary of RACF commands (CICS Transaction Server for z/OS)

## Authorizing users to refresh WLM environments

When you prepare a new version of a stored procedure in a WLM application environment, you need to activate the updated stored procedure by refreshing the application environment.

You can refresh the WLM environment by issuing a VARY REFRESH command at a z/OS command line. Alternatively, you can execute the WLM\_REFRESH stored procedure, which is supplied by DB2 and executes the VARY REFRESH command. This stored procedure is useful when users need to refresh the WLM environment but are not authorized to issue operator commands.

To authorize users to use the WLM\_REFRESH stored procedure:

1. Grant access to the RACF resource profile for each application environment. For example, assume that you want to authorize RACF group DEVL7083 to access the WLM\_REFRESH RACF resource profile for application environment DB9AWLM on subsystem DB9A. To authorize the RACF group in this way, you can issue this command:

```
RDEFINE DSNR (DB9A.WLM_REFRESH.DB9AWLM)
PE DB9A.WLM_REFRESH.DB9AWLM +
CLASS(DSNR) ID(DEVL7083) ACCESS(READ)
END
```

2. Grant the EXECUTE privilege on the stored procedure to users or groups who need to refresh the environment. For example, you can issue the following GRANT statement to authorize the RACF group DEVL7083 to execute the WLM\_REFRESH stored procedure on application environment DB9AWLM:

 **GUI**

```
GRANT EXECUTE ON PROCEDURE SYSPROC.WLM_REFRESH TO DEVL7083;
```

 **GUI**


You need to grant the EXECUTE privilege only once because you supply the application environment name as a variable when you execute the stored procedure.

## Related reference

 WLM\_REFRESH stored procedure (DB2 Application programming and SQL)

 GRANT (function or procedure privileges) (DB2 SQL)

## Related information

 Summary of RACF commands (CICS Transaction Server for z/OS)

## Controlling stored procedure access to non-DB2 resources by using RACF

You can control DB2 stored procedure access to non-DB2 resources (such as VSAM files) by using RACF (or another external security product).

 **GUI**

To control access to non-DB2 resources for an existing stored procedure that does not require RACF (or another external security product):





1. Issue the ALTER PROCEDURE statement with the SECURITY USER clause.

2. Ensure that the user ID that calls the stored procedure has RACF authority to the resources.
3. Enable RACF checking for the caller's ID.
4. For improved performance, specify the following keywords in the COFVLFxx member of library SYS1.PARMLIB to cache the RACF profiles in the virtual look-aside facility (VLF) of z/OS. For example:  

```
CLASS NAME (IRRACEE)
EMAJ (ACEE)
```

#### GUI

#### Related reference

-  [CREATE PROCEDURE \(SQL - external\) \(DB2 SQL\)](#)
-  [CREATE PROCEDURE \(external\) \(DB2 SQL\)](#)
-  [ALTER PROCEDURE \(SQL - external\) \(DB2 SQL\)](#)
-  [ALTER PROCEDURE \(external\) \(DB2 SQL\)](#)

#### Related information

-  [COFVLFxx \(virtual lookaside facility parameters\) \(MVS Initialization and Tuning Reference\)](#)

### Granting the CREATEIN privilege on schemas for stored procedures

When a stored procedure is created, it is explicitly or implicitly qualified by a schema. Users must have the required CREATEIN privilege on the schema before they can create stored procedures.

#### GUI

Many users create stored procedures in the same schema at an application level. These users need the CREATEIN privilege on the schema. You can grant this privilege to a secondary ID or role that is associated with individual users. Those users can then issue a SET CURRENT SQLID statement to the secondary ID or role prior to creating stored procedures in the schema.

To grant the CREATEIN privilege on schemas for stored procedures:

Issue a GRANT statement with the appropriate options, depending on whether you are granting the privilege to a secondary ID or to a role.

- For a secondary ID, issue a GRANT statement with the CREATEIN ON SCHEMA clause. Specify the schema name and secondary ID. For example, assume that you want a user with the secondary ID of PAOLORW to be able to create stored procedures in a schema named DEVL7083. To give this user the necessary privilege, you can issue this statement:

```
GRANT CREATEIN ON SCHEMA DEVL7083 TO PAOLORW;
```


If the ID PAOLORW issues a CREATE PROCEDURE statement without having the required CREATEIN privilege on the schema, an error occurs, and the procedure is not created.

- For a role, issue a GRANT statement with the CREATEIN ON SCHEMA clause. Specify the schema name and the role that will be in effect when the stored procedure is created. (For users to be associated with a role, the trusted context that links them to the role needs to be defined with the ROLE AS OBJECT OWNER AND QUALIFIER clause.) For example, assume that you want to grant


the CREATEIN privilege to a role named ADMINISTRATOR so that users who are associated with the ADMINISTRATOR role can create stored procedures in a schema named DEVL7083. To grant this privilege, you can issue this statement:

```
GRANT CREATEIN ON SCHEMA DEVL7083 TO ROLE ADMINISTRATOR;
```

If a user who is associated with the role named ADMINISTRATOR issues a CREATE PROCEDURE statement without having the required CREATEIN privilege on the schema, an error occurs, and the procedure is not created.

After a secondary ID or role is granted the CREATEIN privilege for a stored procedure and then creates a stored procedure, that ID or role is the owner of that stored procedure. 

#### Related reference

 [GRANT \(schema privileges\) \(DB2 SQL\)](#)

### Granting privileges for using distinct types

Stored procedures can pass parameters that have a distinct type as a data type. When a distinct type is used as a stored procedure parameter, users who create the stored procedure need the USAGE privilege on the distinct type.

When you create a distinct type, you, as the owner of that type, implicitly have the USAGE privilege on the type. You also have the EXECUTE privilege on the associated cast functions. If other users want to create stored procedures that pass a parameter with that distinct type, you need to explicitly grant the USAGE privilege to them.

To grant privileges for using distinct types:

Issue the GRANT statement with the USAGE ON TYPE clause, and specify the name of the distinct type.

- You can grant privileges for using distinct types to an authorization ID. For example, assume that you want the user whose authorization ID is PAOLORW to be able to use the US\_DOLLARS distinct type, which you created. Specifically, this user needs to create a stored procedure that passes a parameter with this data type. To grant this privilege, you can issue this statement:



```
GRANT USAGE ON TYPE US_DOLLARS TO PAOLORW;
```



- You can grant privileges for using distinct types to a role. For example, if you want the role named ADMINISTRATOR to be able to use the US\_DOLLARS distinct type, you can issue this statement:



```
GRANT USAGE ON TYPE US_DOLLARS TO ROLE ADMINISTRATOR;
```



## Related reference

 GRANT (type or JAR file privileges) (DB2 SQL)

## Granting privileges for using JAR files

To use Java archive (JAR) files, you need to have the USAGE privilege on the JAR.

If you have the USAGE privilege on the JAR, you can specify a JAR file in the EXTERNAL NAME clause of a stored procedure with a language type of Java.

To grant privileges for using JAR files:

Issue the GRANT statement, specifying the USAGE clause. For example, assume that you want the user whose AUTHID is PAOLORW to create a Java stored procedure, EMPDTL1J. Assume that the external name of the stored procedure is to be DEVL7083.EmpJar:EmpDtl1J.GetEmpDtls, where:

**DEVL7083.EmpJar**

Is the JAR file name.

**EmpDtl1J**

Is the class name.

**GetEmpDtls**

Is the method name.

AUTHID PAOLORW needs the USAGE privilege (from the JAR file owner ID or schema that was used for executing the INSTALL\_JAR stored procedure). The following statement grants this privilege:

**GUIP**

```
GRANT USAGE ON JAR DEVL7083.EmpJar TO PAOLORW;
```

**GUIP**

In addition, if specified, the contents of the JAR file must already be installed in the DB2 catalog at the time the stored procedure is created.

## Related reference

 GRANT (type or JAR file privileges) (DB2 SQL)

## Granting privileges for executing stored procedures and stored procedure packages

After you create a stored procedure, you need to grant EXECUTE privilege to users who plan to run the stored procedure and the stored procedure package. You can use the GRANT statement to grant the required privileges.

**GUIP**

Invoking a stored procedure requires the EXECUTE privilege on the stored procedure. For external stored procedures (including external SQL procedures), additional authority is needed for the stored procedure package and for most packages that run in the stored procedure.

To grant privileges for executing stored procedures and stored procedure packages:

1. Issue the SQL GRANT statement with the EXECUTE ON PROCEDURE clause to the appropriate authorization ID or role.
  - To grant the EXECUTE privilege to an authorization ID, use the GRANT statement with the EXECUTE ON PROCEDURE clause. For example, to

grant EXECUTE privilege for a stored procedure named SPNAME to a user whose authorization ID is PAOLORW, you can issue the following statement:

```
GRANT EXECUTE ON PROCEDURE SPNAME TO PAOLORW;
```

- To grant the EXECUTE privilege to a role, use the GRANT statement with the EXECUTE ON PROCEDURE clause and the ROLE clause. For example, to grant EXECUTE privilege for a stored procedure named SPNAME to a role named ADMINISTRATOR, you can issue the following statement:

```
GRANT EXECUTE ON PROCEDURE SPNAME TO ROLE ADMINISTRATOR;
```

The DYNAMICRULES behavior for the plan or package that contains the CALL statement determines which authorization ID or role holds the privilege. For more information about the authorization requirements, see CALL (DB2 SQL).

2. Issue the SQL GRANT EXECUTE ON PACKAGE statement with the appropriate options, depending on whether you are granting the privilege to an authorization ID or a role:

- To grant the EXECUTE privilege on the package to an authorization ID, issue the GRANT statement with the EXECUTE ON PACKAGE clause. For example, to grant the privilege to execute a package named PKGNAME to a user whose authorization ID is PAOLORW, you can issue this statement:

```
GRANT EXECUTE ON PACKAGE PKGNAME TO PAOLORW;
```


- To grant the EXECUTE privilege on the package to a role, issue the GRANT statement with the EXECUTE ON PACKAGE clause and the ROLE clause. For example, to grant this privilege to execute a package named PKGNAME to a role named ADMINISTRATOR, you can issue this statement:

```
GRANT EXECUTE ON PACKAGE PKGNAME TO ROLE ADMINISTRATOR;
```

The complete syntax of the GRANT statement that you should use depends on the type of package. For more information about the options for the GRANT statement, see GRANT (function or procedure privileges) (DB2 SQL) and

GRANT (package privileges) (DB2 SQL). 

#### Related reference


 CALL (DB2 SQL)

 GRANT (function or procedure privileges) (DB2 SQL)

 GRANT (package privileges) (DB2 SQL)

### Controlling remote execution of stored procedures by using trusted contexts

You can use trusted contexts and roles to control how a stored procedure can be executed. A *trusted context* is an independent database entity that is based on a system authorization ID (SYSTEM AUTHID) and connection trust attributes.

 For a remote stored procedure CALL, the SYSTEM AUTHID is derived from the system user ID that is provided by an external entity, such as a middleware server. This ID is derived when the connection is initiated. The connection trust attributes are as follows, specified in the CREATE TRUSTED CONTEXT statement:

#### ADDRESS

IP address or domain name. (The protocol is restricted to TCP/IP only.)

#### SERVAUTH

A resource in the RACF SERVAUTH class.

## ENCRYPTION

Minimum level of encryption for the connection:

### NONE

No encryption. This is the default value.

**LOW** DRDA data stream encryption.

**HIGH** Secure Sockets Layer (SSL) encryption.

To call a stored procedure in trusted contexts:

1. Define a role by issuing the CREATE ROLE statement. A *role* is a database entity that groups together one or more privileges and that can be assigned to users by using a trusted context. A role can be used in conjunction with a trusted context and stored procedures to identify one or more authorization IDs that can execute a stored procedure. For example, assume that you want to call stored procedure DEVL7083.EMPDTL1C, which resides on DB2 subsystem DB9A by using authorization ID PAOLORW. Assume also that you want to define a role called SP\_CALLER for use by PAOLORW. You can issue the following SQL statement:

```
CREATE ROLE SP_CALLER;
```

2. Grant the EXECUTE privilege on a stored procedure to that role. For example, to grant the EXECUTE privilege to the role called SP\_CALLER for the stored procedure named EMPDTL1C, you can issue the following statement:

```
GRANT EXECUTE ON PROCEDURE DEVL7083.EMPDTL1C TO ROLE SP_CALLER;
```

3. Have an authorized user bind the stored procedure package. The user either needs SYSADM authority or must have explicitly bind authority for that stored procedure. For example, assume that an authorized user wants to bind stored procedure DEVL7083.EMPDTL1C into stored procedure package DEVL7083.EMPDTL1CPKG. You can issue the following statement:

```
BIND PACKAGE(DEVL7083) MEMBER(EMPDTL1CPKG)
```

4. Grant the EXECUTE privilege on the stored procedure package to the authorization ID or role that needs to run it. For example, to grant the EXECUTE privilege on stored procedure package DEVL7083.EMPDTL1CPKG to the role named SP\_CALLER, you can issue this statement:

```
GRANT EXECUTE ON PACKAGE DEVL7083.EMPDTL1CPKG TO ROLE SP_CALLER;
```

5. Define the trusted context. For example, assume that you want to define a trusted context named TRUSTED\_EMPDTL1C that uses:

- System authorization ID PAOLORW
- Default role SP\_CALLER
- IP address 9.30.28.113

To define this trusted context, you can issue the following statement:

```
CREATE TRUSTED CONTEXT TRUSTED_EMPDTL1C  
BASED UPON CONNECTION USING SYSTEM AUTHID PAOLORW  
ATTRIBUTES (ADDRESS '9.30.28.113')  
DEFAULT ROLE SP_CALLER  
ENABLE;
```

### GUI

6. Optional: Verify that the authorization ID can execute the stored procedure by running the application program that invokes the stored procedure and looking at the system output. For example, assume that an application named CALLEMPD uses a CALL *:host-variable* statement to invoke the stored



procedure named DEVL7083.EMPDTL1C. Assume also that the application program generates trace output. You might see the following system output:

```
DEVL7083.CALLEMPD - Run started.  
Data returned in result sets is limited to the first 50 rows.  
Data returned in result set columns is limited to the first 100  
bytes or characters.  
DEVL7083.CALLEMPD - Calling the stored procedure.  
DEVL7083.CALLEMPD - Run completed.
```

#### Related reference

➞ CREATE TRUSTED CONTEXT (DB2 SQL)

➞ CREATE ROLE (DB2 SQL)

➞ GRANT (function or procedure privileges) (DB2 SQL)

## Protecting connection requests that use the TCP/IP protocol

You can set your DB2 subsystem to send or receive connection requests that use the TCP/IP network protocol. You need to authorize the started task user ID (SYSDSP) that is associated with the DB2 distributed address space (ssnmDIST) to use the z/OS UNIX System Services.

To secure connection requests over TCP/IP:

1. Create an OMVS segment in the RACF user profile for the started task user ID (SYSDSP)
2. Specify a z/OS UNIX user identifier of 0 and the maximum number of files of that the user is allowed to have concurrently active to 131702 in the following command:

```
ADDUSER ddfluid OMVS(UID(0) FILEPROCMA(131702))
```

If the *ddfluid* ID already exists, use:

```
ALTUSER ddfluid OMVS(UID(0) FILEPROCMA(131702))
```

The started task user ID of the DB2 distributed address space only needs a z/OS UNIX user identifier of 0 (UID(0)). A UID 0 is considered a superuser. If you don't want to grant the superuser authority to the started task user ID that is associated with the ssnmDIST address space during the DB2 installation, you can specify a value other than 0 for the UID parameter. Make sure that the value is a valid z/OS UNIX user identifier.

3. If you want to assign a z/OS group name to the address space, assign an OMVS segment to the z/OS group name by using one of the following RACF commands:

```
ADDGROUP ddfgnm OMVS(GID(nnn))...
```

```
ALTGROUP ddfgnm OMVS(GID(nnn))...
```

where *ddfgnm* is the z/OS group name and *nnn* is any valid, unique identifier.

The standard way to assign a z/OS userid and a z/OS group name to a started address space is to use the z/OS Security Server (RACF) STARTED resource class. This method enables you to dynamically assign a z/OS user ID by using commands instead of requiring an IPL to have the assignment take effect. The alternative method to assign a z/OS user ID and a z/OS group name to a started address space is to change the RACF started procedures table, ICHRIN03.

You can also manage TCP/IP requests in a trusted context. A trusted context allows you to use a trusted connection without needing additional authentication and to acquire additional privileges through the definition of roles.



The TCP/IP Already Verified (DSN6FAC TCPALVER) controls whether DB2 accepts TCP/IP connection requests that contain only a user ID. However, in the case of a trusted context, it is the definition of the trusted context, not the TCPALVER setting, handles the requirement for switching users of a trusted connection.

Do not set DSN6FAC TCPALVER to YES if you use a trusted context. If you set TCPALVER to YES in the definition of the trusted context, you need to define the authorization ID that establishes the trusted connection in the USER clause to enforce the authentication requirement.

#### Related tasks

“Defining DB2 resources to RACF” on page 127

“Permitting RACF access” on page 129

“Managing authorization for stored procedures” on page 137

“Establishing Kerberos authentication through RACF”

## Establishing Kerberos authentication through RACF

*Kerberos security* is a network security technology that was developed at the Massachusetts Institute of Technology. The Kerberos security technology does not require passwords to flow in readable text because it uses encrypted tickets that contain authentication information for the users.

DB2 can use Kerberos security services to authenticate remote users. With Kerberos security services, remote users need to issue their Kerberos name and password to access DB2. They can use the same name and password for access throughout the network, which makes a separate password to access DB2 unnecessary.

A remote user who is authenticated to DB2 by means of Kerberos authentication must be registered in RACF profiles. An organization that runs a Kerberos server establishes its own *realm*. The name of the realm in which a client is registered is part of the client's name and can be used by the application server to accept or reject a request.

To authenticate and register a remote user in RACF profiles:

1. Define the Kerberos realm to RACF by issuing the following command:

```
RDEFINE REALM KERBDFLT KERB(KERBNAME(localrealm) PASSWORD(mykerpw)
```

You must specify the name of the local realm in the definition. You must also specify a Kerberos password for RACF to grant Kerberos tickets.

2. Define local principals to RACF by issuing the following command:

```
AU RONTOMS KERB(KERBNAME(rontoms))  
ALU RONTOMS PASSWORD(new1pw) NOEXPIRE
```

Make sure to change RACF passwords before the principals become active Kerberos users.

3. Map foreign Kerberos principals by defining KERBLINK profiles to RACF with a command similar to the following:

```
RDEFINE KERBLINK /.../KERB390.ENDICOTT.IBM.COM/RWH APPLDATA('RONTOMS')
```

You must also define a principal name for the user ID that is used in the *ssnmDIST* started task address space, as shown in the following example:

```
ALU SYSDSP PASSWORD(pw) NOEXPIRE KERB(KERBNAME(SYSDSP))
```

The `ssnmDIST` address space must have the RACF authority to use its SAF ticket parsing service. The user ID that is used for the `ssnmDIST` started task address space is `SYSDSP`.

4. Define foreign Kerberos authentication servers to the local Kerberos authentication server by issuing the following command:

```
RDEFINE REALM /.../KRB390.ENDICOTT.IBM.COM/KRBTGT/KER2000.ENDICOTT.IBM.COM +  
KERB(PASSWORD(realmpw))
```

You must supply a password for the key to be generated. REALM profiles define the trust relationship between the local realm and the foreign Kerberos authentication servers. PASSWORD is a required keyword, so all REALM profiles have a KERB segment.

**Data sharing environment:** Data sharing Sysplex environments that use Kerberos security must have a Kerberos Security Server instance running on each system in the Sysplex. The instances must either be in the same realm and share the same RACF database, or have different RACF databases and be in different realms.

#### Related tasks

“Defining DB2 resources to RACF” on page 127

“Permitting RACF access” on page 129

“Managing authorization for stored procedures” on page 137

“Protecting connection requests that use the TCP/IP protocol” on page 146

---

## Implementing DB2 support for enterprise identity mapping

*Enterprise identity mapping* (EIM) enables the mapping of user identities across servers that are integrated but that do not share user registries. DB2 supports the EIM capability by implementing the SAF user mapping plug-in callable service, which is part of the z/OS Security Server (RACF).

You can exploit the EIM support by using the IBM Websphere Application Server 6.0.1, the IBM DB2 Driver for JDBC and SQLJ, and the IBM DB2 Driver for ODBC and CLI.

You must install z/OS V1.8 or higher to use the SAF user mapping plug-in service and implement the DB2 support for the EIM.

To implement the DB2 support for EIM:

1. Configure the z/OS LDAP server with a TDBM backend
2. Set up RACF for the LDAP server
3. Configure the z/OS EIM domain controller
4. Add the SAF user mapping data set to LNKLIST

If you enable DB2 support for EIM, DB2 can retrieve the mapped user ID from the SAF user mapping plug-in and specify the information in the ICTX structure. During the ENVIR=CREATE processing, DB2 passes the information to RACF through the RACROUTE REQUEST=VERIFY macro service. When RACF successfully authenticates the user, the ICTX structure is anchored in the ACEEICTX field.

**Note:** The SAF user identity mapping plug-in service will not be supported in the future release of DB2 for z/OS.

## Related reference

 [z/OS Security Server RACF Command Language Reference](#)

 [z/OS Integrated Security Services LDAP Server Administration and Use](#)

 [z/OS Integrated Security Services Enterprise Identity Mapping \(EIM\) Guide and Reference](#)

## Configuring the z/OS LDAP server

When DB2 receives an authenticated user registry name, it invokes the SAF user mapping plug-in service. This service uses the EIM domain, which is an LDAP server, to retrieve the z/OS user ID that is used as the primary authorization ID.

You can use the LDAP configuration (**ldapcnf**) utility to configure and set up a z/OS LDAP server. The LDAP configuration utility requires the `ldap.profile` input file that is shipped in the `/usr/lpp/ldap/etc` directory. The `ldap.profile` file contains the settings that you need to set up the LDAP server.

To configure a z/OS LDAP server:

1. Copy and modify the `ldap.profile` file based on your own environment.
2. Issue the following command to run the LDAP configuration utility with the `ldap.profile` file that you modified:

```
ldapcnf -i ldap.profile
```

The LDAP configuration utility generates the following output files:

- SLAPDCNF member as the LDAP server configuration file
- SLAPDENV member as the LDAP server environment variable file
- PROG member for APF authorization
- GLDSRV procedure for starting the LDAP server
- DSNAOINI configuration file for DB2 CLI
- TDBSPUFI DB2 SQL DDL statements for creating the TDBM environment
- DBCLI DB2 SQL BIND statements for binding the CLI/ODBC packages and plan
- RACF member for creating the RACF profiles that protect the LDAP server service task and grant permissions for the user ID to run the LDAP server

These output files are stored in the `OUTPUT_DATASET_NAME` that you specified in the `ldap.profile` file.

3. Submit the following output JCL files after DB2 is started:
  - DBCLI member file
  - RACF member file
4. Submit the TDBSPUFI member file by using the DB2 SPUFI interactive tool.
5. Start the LDAP server from SDSF or the operator's console.

The name of the LDAP server procedure file is the same as the user ID that is specified on the `LDAPUSRID` statement. The pre-assigned value is `GLDSRV`.

To start the LDAP server from SDSF, enter:

```
/s GLDSRV
```

To start the LDAP server from the operator's console, enter:

```
s GLDSRV
```

6. Copy the `schema.user.ldif` file from the `/usr/lpp/ldap/etc` directory to a local directory

7. Use the following **ldapmodify** utility to modify the schema entry for the TDBM backend

```
ldapmodify -h ldaphost -p ldapport -D binddn -w passwd -f file
```

The following example shows how to use the **ldapmodify** utility:

```
ldapmodify -h v25ec099.svl.ibm.com -p 3389  
-D "cn=LDAP Administrator"  
-w secret -f schema.user.ldif
```

At the top of the `schema.user.ldif` file, find the following line, and supply the appropriate TDBM suffix in that line

```
dn: cn=schema, <suffix>
```

The suffix is the same value that is used in the `TDBM_SUFFIX` statement in the `ldap.profile` file, as in the following example:

```
dn: cn=schema, o=IBM, c=US
```

8. Use the **ldapadd** utility to load the suffix entry and to create a user ID that is used by the SAF user mapping plug-in for binding with the LDAP server. You can use the following **ldapadd** utility statement:

```
ldapadd -h ldaphost -p ldapport -D binddn -w passwd -f file
```

The following is an example of using the **ldapadd** utility:

```
ldapadd -h v25ec099.svl.ibm.com -p 3389  
-D "cn=LDAP Administrator"  
-w secret -f setup.ldap.ldif
```

## Setting up RACF for the z/OS LDAP server

After you configure the z/OS LDAP server, you need to set up RACF to activate identity mapping. You also need to grant DB2 authority to use the SAF user mapping plug-in service.

To set up RACF for the z/OS LDAP server:

1. Enable identity mapping by activating the FACILITY class.

The FACILITY class must be active to enable identity mapping. Use the following **SETROPTS** command if it is not already active at your installation:

```
SETROPTS CLASSACT(FACILITY)
```

2. Define a KEYMSTR profile to store an encryption key.

Make sure to choose a key that is known only to the security administrator, and store it in the KEYMSTR profile that you defined, as shown in the following example:

```
RDEF KEYMSTR LDAP.BINDPW.KEY SSIGNON(KEYMASKED(0123456789ABCDEF))
```

The LDAP BIND passwords are encrypted with the key that is stored in the `LDAP.BINDPW.KEY` profile. The value of the key in this example is `0123456789ABCDEF`.

3. Authorize DB2 to request lookup services by defining and granting READ access to the SYSDSP user in the following RACF profiles:

```
RDEF FACILITY IRR.RGETINFO.EIM UACC(NONE)  
PE IRR.RGETINFO.EIM ACCESS(READ) ID(SYSDSP) CL(FACILITY)
```

```
RDEF FACILITY IRR.RDCEKEY UACC(NONE)  
PE IRR.RDCEKEY ACCESS(READ) ID(SYSDSP) CL(FACILITY)
```

4. Define the IRR.PROXY.DEFAULTS profile in the FACILITY class, as follows:

```
RDEF FACILITY IRR.PROXY.DEFAULTS  
PROXY(LDAPHOST('ldap://v25ec099.svl.ibm.com:3389'))  
BINDDN('cn=eim user,o=IBM,c=US') BINDPW('secret'))
```

```
EIM(DOMAINDN('ibm-eimDomainName=My Domain,o=IBM,c=US'))
LOCALREG('My Target Registry'))
```

```
SETOPTS RACLIST(FACILITY) REFRESH
```

5. Grant DB2 the authority to use the SAF user mapping plug-in service by issuing the following commands:

```
RDEF PROGRAM IRRSPIM ADDMEM ('USER.PRIVATE.DLLLIB'//NOPADCHK)
PE IRRSPIM ACCESS(READ) ID(SYSDSP) CL(PROGRAM)
```

```
RDEF PROGRAM IRRSPIME ADDMEM ('USER.PRIVATE.DLLLIB'//NOPADCHK)
PE IRRSPIME ACCESS(READ) ID(SYSDSP) CL(PROGRAM)
```

```
SETOPTS WHEN(PROGRAM) REFRESH
```

## Setting up the EIM domain controller

After you set up the LDAP server and RACF, you need to use the RACF *eimadmin* utility to create and configure an EIM domain controller.

To create an EIM domain controller in this situation:

1. Create an EIM domain by issuing the following command:

```
eimadmin -aD -d 'ibm-eimDomainName=My Domain,o=IBM,c=US'
-h ldap://v25ec099.svl.ibm.com:3389
-b 'cn=LDAP Administrator' -w secret
```

The example shows that the new domain name is "My Domain." It also shows that the TDBM\_SUFFIX statement in the ldap.profile file is defined as o=IBM,c=US.

2. Grant the EIM user access to the EIM domain for performing lookup services by issuing the following command:

```
eimadmin -aC -c MAPPING -q 'cn=eim user, o=IBM, c=US' -f DN
-d 'ibm-eimDomainName=My Domain,o=IBM,c=US'
-h ldap://v25ec099.svl.ibm.com:3389
-b 'cn=LDAP Administrator' -w secret
```

3. Create the source registry in the EIM domain by issuing the following command:

```
eimadmin -aR -r "My Source Registry" -y KERBEROS
-d 'ibm-eimDomainName=My Domain,o=IBM,c=US'
-h ldap://v25ec099.svl.ibm.com:3389
-b 'cn=LDAP Administrator' -w secret
```

4. Create the target registry in the EIM domain by issuing the following command:

```
eimadmin -aR -r "My Target Registry" -y RACF
-d 'ibm-eimDomainName=My Domain,o=IBM,c=US'
-h ldap://v25ec099.svl.ibm.com:3389
-b 'cn=LDAP Administrator' -w secret
```

5. Add the enterprise identifier "Cat" to the EIM domain by issuing the following command:

```
eimadmin -aI -i "Cat" -d 'ibm-eimDomainName=My Domain,o=IBM,c=US'
-h ldap://v25ec099.svl.ibm.com:3389
-b 'cn=LDAP Administrator' -w secret
```

You can add multiple enterprise identifiers to the same EIM domain at any time.

6. Associate registry user IDs with the identifiers in the EIM domain by issuing the following commands:

```
eimadmin -aA -u "Kitty" -r "My Source Registry" -t SOURCE
-i "Cat" -d 'ibm-eimDomainName=My Domain,o=IBM,c=US'
-h ldap://v25ec099.svl.ibm.com:3389
```

```
-b 'cn=LDAP Administrator' -w secret

eimadmin -aA -u "Buffy" -r "My Target Registry" -t TARGET
-o "db2/stlec1/valadist" -i "Cat"
-d 'ibm-eimDomainName=My Domain,o=IBM,c=US'
-h ldap://v25ec099.svl.ibm.com:3389
-b 'cn=LDAP Administrator' -w secret
```

Specify the "-o" flag with the "db2/location-name/subsystem-name"+ "dist" value when you define a user ID for DB2 to use as the primary authorization ID in your target registry. As the examples show, when DB2 calls the SAF user mapping plug-in service to retrieve the primary authorization ID, DB2 specifies the additional db2/location-name/subsystem-name"+ "dist" information for the plug-in service to look up.

If a target identity is found with the same information, the target identity "Buffy" is returned. If the target identity does not contain any additional information, user ID "Buffy" is also returned to DB2. However, if the target registry contains multiple user identities and if none of them contains the recommended additional information, no user identity is returned to DB2.

## Adding the SAF user mapping plug-in data set to LNKLIST

The SAF user mapping plug-in IRRSPIME resides in a z/OS data set. This data set must be included in the LNKLIST. If the data set is not included, you need to add it to the LNKLIST.

To add the z/OS data set that contains the SAF user mapping plug-in to the LNKLIST:

1. Define a new LNKLIST by issuing the following command from the operator console:  

```
SETPROG LNKLIST,DEFINE,NAME=MYLNKLIST,COPYFROM=CURRENT
```
2. Add the USER.PRIVATE.DLLLIB data set on the USER01 volume to the new MYLNKLIST by issuing the following command:  

```
SETPROG LNKLIST,ADD,NAME=MYLNKLIST,DSNAME=USER.PRIVATE.DLLLIB,
VOLUME=USER01
```
3. Activate the new MYLNKLIST by issuing the following command:  

```
SETPROG LNKLIST,ACTIVATE,NAME=MYLNKLIST
```
4. Use the MYLNKLIST to update the current tasks in the system by issuing the following command:  

```
SETPROG LNKLIST,UPDATE,JOB=*
```

---

## Implementing DB2 support for distributed identity filters

A *distributed identity filter* is a RACF mapping association between a RACF user ID and one or more distributed user identities. You can use the RACF **RACMAP** command to associate a distributed user identity with a RACF user ID.

RACF distributed identity filters are implemented through z/OS identify propagation. You must install and run z/OS Version 1 Release 11 to use distributed identity filters.

DB2 provides support for z/OS identify propagation and distributed identity filters. You need to create distributed identity filters to take advantage of this support.

To create a distributed identity filter:

1. Activate the RACF general resource IDIDMAP class and enable it for RACLIST processing by issuing the following command:

```
SETOPTS CLASSACT(IDIDMAP) RACLIST(IDIDMAP)
```

2. Define a distributed identity filter and associate the distributed user name with a RACF user ID by issuing the RACF **RACMAP** command. To define a filter for a non-LDAP user name, specify the user name as a simple character string to be defined in a non-LDAP registry. Suppose that the distributed user name is 'MARY' which is defined in user registry 'Registry01'. If you want to map this user name to RACF user ID 'DB2USER1', you can issue the **RACMAP** command, as follows

```
RACMAP ID(DB2USER1) MAP
      USERIDFILTER(NAME('MARY'))
      REGISTRY(NAME('Registry01'))
      WITHLABEL('Filter for MARY from Registry01')
```

3. Refresh the IDIDMAP class profile by issuing the following command:

```
SETOPTS RACLIST(IDIDMAP) REFRESH
```

4. If necessary, review the distributed identity filter by issuing the following RACMAP LISTMAP command:

```
RACMAP ID(DB2USER1) LISTMAP
```

If the new filter is successfully created, the following output is returned:

```
Mapping information for user DB2USER1:
Label: Filter for MARY from Registry01
Distributed Identity User Name Filter:
>MARY<
Registry name:
>Registry01<
```

The new filter assigns RACF user ID DB2USER1 when the distributed identity is user MARY from Registry01. When user MARY authenticates her identity at her distributed application server and performs tasks that access a remote DB2 server system, DB2 passes distributed user name MARY and registry name Registry01 as character strings to RACF.

During DB2 remote connection processing, DB2 calls the RACF RACROUTE REQUEST=VERIFY ENVIR=CREATE macro service. RACF uses these data values to search the IDIDMAP profiles for a matching filter. RACF finds the matching filter labeled 'Filter for MARY from Registry01' and assigns it the DB2USER1 user ID. The remote connection then executes its transactions with the authority of the DB2USER1 user ID. If in place, audit records for this transaction contains both RACF user ID DB2USER1, distributed user MARY, and registry name Registry01 that DB2 passes to RACF.

#### Related reference

 z/OS Security Server RACF Security Administrator's Guide

---

## Managing connection requests from local applications

Different local processes enter the access control procedure at different points, depending on the environment in which they originate.

The following processes go through connection processing only:

- Requests originating in TSO foreground and background (including online utilities and requests through the call attachment facility)
- JES-initiated batch jobs



- Requests through started task control address spaces (from the z/OS START command)

The following processes go through connection processing and can later go through the sign-on processing:

- The IMS control region.
- The CICS recovery coordination task.
- DL/I batch.
- Applications that connect using the Resource Recovery Services attachment facility (RRSAF).

The following processes go through sign-on processing:

- Requests from IMS dependent regions (including MPP, BMP, and Fast Path)
- CICS transaction subtasks

IMS, CICS, RRSAF, and DDF-to-DDF connections can send a sign-on request, typically to execute an application plan. That request must provide a primary ID, and can also provide secondary IDs. After a plan is allocated, it need not be deallocated until a new plan is required. A different transaction can use the same plan by issuing a new sign-on request with a new primary ID.

## Processing of connection requests

A connection request makes a new connection to DB2; it does not reuse an application plan that is already allocated. Therefore, an essential step in processing the request is to check that the ID is authorized to use DB2 resources.

DB2 completes the following steps to process a connection request:

1. DB2 obtains the initial primary authorization ID. As shown in the following table, the source of the ID depends on the type of address space from which the connection was made.

*Table 36. Sources of initial primary authorization IDs*

Source	Initial primary authorization ID
TSO	TSO logon ID.
BATCH	USER parameter on JOB statement.
IMS control region or CICS	USER parameter on JOB statement.
IMS or CICS started task	Entries in the started task control table.
Remote access requests	Depends on the security mechanism used.

2. RACF is called through the z/OS system authorization facility (SAF) to check whether the ID that is associated with the address space is authorized to use the following resources:

- The DB2 resource class (CLASS=DSNR)
- The DB2 subsystem (SUBSYS=ssnm)
- The requested connection type

The SAF return code (RC) from the invocation determines the next step, as follows:

- **If RC > 4**, RACF determined that the RACF user ID is not valid or does not have the necessary authorization to access the resource name. DB2 rejects the request for a connection.
- **If RC = 4**, the RACF return code is checked.



- If RACF return code value is **equal to 4**, the resource name is not defined to RACF and DB2 rejects the request with reason code X'00F30013'.
  - If RACF return code value is **not equal to 4**, RACF is not active. DB2 continues with the next step, but the connection request and the user are not verified.
  - **If RC = 0**, RACF is active and has verified the RACF user ID; DB2 continues with the next step.
3. If RACF is active and has verified the RACF user ID, DB2 runs the connection exit routine. To use DB2 secondary IDs, you must replace the exit routine. If you do not want to use secondary IDs, do nothing. The IBM-supplied default connection exit routine continues the connection processing. The process has the following effects:
- The DB2 primary authorization ID is set based on the following rules:
    - If a value for the initial primary authorization ID exists, the value becomes the DB2 primary ID.
    - If no value exists (the value is blank), the primary ID is set by default, as shown in the following table.

*Table 37. Sources of default authorization identifiers*

Source	Default primary authorization ID
TSO	TSO logon ID
BATCH	USER parameter on JOB statement
Started task, or batch job with no USER parameter	Default authorization ID set when DB2 was installed (UNKNOWN AUTHID on installation panel DSNTIPP)
Remote request	None. The user ID is required and is provided by the DRDA requester.

- The SQL ID is set equal to the primary ID.
  - No secondary IDs exist.
4. DB2 determines if TSO and BATCH connections that use DSN, RRSAP, and Utilities are trusted.
- For a TSO and BATCH connection that uses DSN, RRSAP, and Utilities, DB2 checks to see if a matching trusted context is defined for the primary authorization ID and the job name. If a matching trusted context is found, the connection is established as trusted.

#### **Related concepts**

“Connection routines and sign-on routines” on page 229

#### **Related tasks**

“Using sample connection and sign-on exit routines for CICS transactions” on page 159

“Specifying connection and sign-on routines” on page 229

“Debugging connection and sign-on routines” on page 238

#### **Related reference**

“Processing of sign-on requests” on page 157

## **Using secondary IDs for connection requests**

If you want to use DB2 secondary authorization IDs, you must replace the default connection exit routine. If you want to use RACF group names as DB2 secondary IDs, the easiest method is to use the IBM-supplied sample routine.

The following table lists the difference between the default and sample connection exit routines.

*Table 38. Differences between the default and sample connection exit routines*

Default connection exit routine	Sample connection exit routine
Supplied as object code.	Supplied as source code. You can change the code.
Installed as part of the normal DB2 installation procedure.	Must be compiled and placed in the DB2 library.
Provides values for primary IDs and SQL IDs, but does not provide values for secondary IDs.	Provides values for primary IDs, secondary IDs, and SQL IDs.

The sample connection exit routine has the following effects:

- The sample connection exit routine sets the DB2 primary ID in the same way that the default routine sets the DB2 primary ID, and according to the following rules:
  - If the initial primary ID is not blank, the initial ID becomes the DB2 primary ID.
  - If the initial primary ID is blank, the sample routine provides the same default value as does the default routine.
  - If the sample routine cannot find a nonblank primary ID, DB2 uses the default ID (UNKNOWN AUTHID) from the DSNTIPP installation panel. In this case, no secondary IDs are supplied.
- The sample connection exit routine sets the SQL ID based on the following criteria:
  - The routine sets the SQL ID to the TSO data set name prefix in the TSO user profile table if the following conditions are true:
    - The connection request is from a TSO-managed address space, including the call attachment facility, the TSO foreground, and the TSO background.
    - The TSO data set name prefix is equal to the primary ID or one of the secondary IDs.
  - In all other cases, the routine sets the SQL ID equal to the primary ID.
- The secondary authorization IDs depend on RACF options:
  - If RACF is not active, no secondary IDs exist.
  - If RACF is active but its “list of groups” option is not active, one secondary ID exists (the default connected group name) if the attachment facility supplied the default connected group name.
  - If RACF is active and the “list of groups” option is active, the routine sets the list of DB2 secondary IDs to the list of group names to which the RACF user ID is connected. Those RACF user IDs that are in REVOKE status do not become DB2 secondary IDs. The maximum number of groups is 1012. The list of group names is obtained from RACF and includes the default connected group name.

If the default connection exit routine and the sample connection exit routine do not provide the flexibility and features that your subsystem requires, you can write your own exit routine.

## Processing of sign-on requests

Requests can come from IMS-dependent regions, CICS transaction subtasks, or RRS connections. For each of these types of requests, the initial primary ID is obtained immediately before a plan for the transaction is allocated. A new sign-on request can run the same plan without de-allocating and reallocating the plan.

Unlike the connection processing, the sign-on processing does not check the RACF for the user ID of the address space. DB2 completes the following steps to process sign-on requests:

1. DB2 determines the initial primary ID as follows:

**For IMS sign-ons** from message-driven regions, if the user has signed on, the initial primary authorization ID is the user's sign-on ID. IMS passes to DB2 the IMS sign-on ID and the associated RACF connected group name, if one exists. If the user has not signed on, the primary ID is the LTERM name, or if that is not available, the PSB name. For a batch-oriented region, the primary ID is the value of the USER parameter on the job statement, if that is available. If that is not available, the primary ID is the program's PSB name.

For remote requests, the source of the initial primary ID is determined by entries in the SYSIBM.USERNAMES table. For connections using Resource Recovery Services attachment facility, the processing depends on the type of signon request:

- SIGNON
- AUTH SIGNON
- CONTEXT SIGNON

For SIGNON, the primary authorization ID is retrieved from ACEEUSRI if an ACEE is associated with the TCB (TCBSENV). This is the normal case. However, if an ACEE is not associated with the TCB, SIGNON uses the primary authorization ID that is associated with the address space, that is, from the ASXB. If the new primary authorization ID was retrieved from the ACEE that is associated with the TCB and ACEEGRPN is not null, DB2 uses ACEEGRPN to establish secondary authorization IDs.

With AUTH SIGNON, an APF-authorized program can pass a primary authorization ID for the connection. If a primary authorization ID is passed, AUTH SIGNON also uses the value that is passed in the secondary authorization ID parameter to establish secondary authorization IDs. If the primary authorization ID is not passed, but a valid ACEE is passed, AUTH SIGNON uses the value in ACEEUSRI for the primary authorization ID if ACEEUSRI is not 0. If ACEEUSRI is used for the primary authorization ID, AUTH SIGNON uses the value in ACEEGRPN as the secondary authorization ID if ACEEGRPL is not 0.

For CONTEXT SIGNON, the primary authorization ID is retrieved from data that is associated with the current RRS context using the context\_key, which is supplied as input. CONTEXT SIGNON uses the CTXSDTA and CTXRDTA functions of RRS context services. An authorized function must use CTXSDTA to store a primary authorization ID prior to invoking CONTEXT SIGNON. Optionally, CTXSDTA can be used to store the address of an ACEE in the context data that has a context\_key that was supplied as input to CONTEXT SIGNON. DB2 uses CTXRDTA to retrieve context data. If an ACEE address is passed, CONTEXT SIGNON uses the value in ACEEGRPN as the secondary authorization ID if ACEEGRPL is not 0.

2. DB2 runs the sign-on exit routine. **User action:** To use DB2 secondary IDs, you must replace the exit routine.

If you **do not** want to use secondary IDs, do nothing. Sign-on processing is then continued by the IBM-supplied default sign-on exit routine, which has the following effects:

- The initial primary authorization ID remains the primary ID.
- The SQL ID is set equal to the primary ID.
- No secondary IDs exist.

You can replace the exit routine with one of your own, even if it has nothing to do with secondary IDs. If you do, remember that IMS and CICS recovery coordinators, their dependent regions, and RRSF take the exit routine only if they have provided a user ID in the sign-on parameter list.

3. DB2 determines if the user of a trusted RRSF SIGNON connection is allowed to switch.

For a RRSF SIGNON connection that is trusted, DB2 checks to see if the primary authorization ID is allowed to switch in the trusted connection. If the primary authorization ID is not allowed to switch, the connection is returned to the unconnected state.

#### **Related concepts**

“Connection routines and sign-on routines” on page 229

#### **Related tasks**

“Using sample connection and sign-on exit routines for CICS transactions” on page 159

“Specifying connection and sign-on routines” on page 229

“Debugging connection and sign-on routines” on page 238

#### **Related reference**

“Processing of sign-on requests” on page 157

## **Using secondary IDs for sign-on requests**

If you want the primary authorization ID to be associated with DB2 secondary authorization IDs, you must replace the default sign-on exit routine.

The procedure is similar to that for connection processing. If you want to use RACF group names as DB2 secondary IDs, the easiest method is to use the IBM-supplied sample routine. An installation job can automatically replace the default routine with the sample routine.

Distinguish carefully between the two routines. The default sign-on routine provides no secondary IDs and has the following effects:

- The initial primary authorization ID remains the primary ID.
- The SQL ID is set equal to the primary ID.
- No secondary IDs exist.

Like the sample connection routine, the sample sign-on routine supports DB2 secondary IDs and has the following effects:

- The initial primary authorization ID is left unchanged as the DB2 primary ID.
- The SQL ID is made equal to the DB2 primary ID.
- The secondary authorization IDs depend on RACF options:
  - If RACF is not active, no secondary IDs exist.
  - If RACF is active but its “list of groups” option is not active, one secondary ID exists; it is the name passed by CICS or by IMS.
  - If RACF is active and you have selected the option for a list of groups, the routine sets the list of DB2 secondary IDs to the list of group names to which

the RACF user ID is connected, up to a limit of 1012 groups. The list of group names includes the default connected groupname.

## Using sample connection and sign-on exit routines for CICS transactions

For a CICS transaction to use the sample connection or sign-on exit routines, the external security system, such as RACF, must be defined to CICS.

Define an external security system, such as RACF, to CICS with the following specifications:

- The CICS system initialization table must specify external security.
  - For CICS Version 4 or later, specify SEC=YES.
  - For earlier releases of CICS, specify EXTSEC=YES.

If you are using the CICS multiple region option (MRO), you must specify SEC=YES or EXTSEC=YES for every CICS system that is connected by interregion communication (IRC).

- If your version of CICS uses a sign-on table (SNT), the CICS sign-on table must specify EXTSEC=YES for each signed on user that uses the sign-on exit.
- When the user signs on to a CICS terminal-owning region, the terminal-owning region must propagate the authorization ID to the CICS application-owning region.

You must change the sample sign-on exit routine (DSN3SSGN) before using it if the following conditions are all true:

- You have the RACF list-of-groups option active.
- You have transactions whose initial primary authorization ID is not defined to RACF.

### Related concepts

“Connection routines and sign-on routines” on page 229

### Related reference

“Processing of connection requests” on page 154

“Processing of sign-on requests” on page 157

“Sample connection and sign-on routines” on page 230

“Exit parameter list for connection and sign-on routines” on page 231

---

## Managing connection requests from remote applications

If you control requests from remote applications, your DB2 subsystem might be accepting requests from applications that use SNA network protocols, TCP/IP network protocols, or both.

## Security mechanisms for DRDA and SNA

SNA and DRDA have different security mechanisms. DRDA allows a user to be authenticated by using SNA security mechanisms or DRDA mechanisms, which are independent of the underlying network protocol.

For an SNA network connection, a DRDA requester can send security tokens by using a SNA attach or a DRDA command. DB2 for z/OS as a requester uses SNA security mechanisms if it uses a SNA network connection (except for Kerberos) and DRDA security mechanisms for TCP/IP network connections (or when Kerberos authentication is chosen, regardless of the network type).

## Security mechanisms for DB2 for z/OS as a requester

As a requester, DB2 for z/OS chooses SNA or DRDA security mechanisms based on the network protocol and the authentication mechanisms that you use.

If you use SNA protocols, DB2 supports the following SNA authentication mechanisms:

- User ID only (already verified)
- User ID and password
- User ID and PassTicket

Authentication is performed based on SNA protocols, which means that the authentication tokens are sent in an SNA attach (FMH-5).

If you use TCP/IP protocols, DB2 supports the following DRDA authentication mechanisms:

- User ID only (already verified)
- User ID and password
- User ID and PassTicket

If you use TCP/IP protocols with the z/OS Integrated Cryptographic Service Facility, DB2 also supports the following DRDA authentication mechanisms:

- Encrypted user ID and encrypted password
- Encrypted user ID and encrypted security-sensitive data

Authentication is performed based on DRDA protocols, which means that the authentication tokens are sent in DRDA security flows.

## Security mechanisms for DB2 for z/OS as a server

As a server, DB2 for z/OS can accept either SNA or DRDA authentication mechanisms. It can authenticate remote users from the security tokens that are obtained from the SNA ATTACH (FMH-5) or from the DRDA security commands that described by each of the protocols.

DB2 for z/OS accepts connection requests from remote clients that use AES or DES encryption algorithm to protect user IDs and passwords over a TCP/IP network. Specifically, it supports the following authentication methods:

- User ID only (already verified at the requester)
- User ID and password
- User ID and PassTicket
- Kerberos tickets
- Unencrypted user ID and encrypted password
- Encrypted user ID and encrypted password
- User ID, password, and new password

DB2 for z/OS as a server also supports the following authentication mechanisms if the z/OS Integrated Cryptographic Service Facility is installed and active:

- Encrypted user ID and encrypted security-sensitive data
- Encrypted user ID, encrypted password, and encrypted security-sensitive data
- Encrypted user ID, encrypted password, encrypted new password, and encrypted security-sensitive data
- Encrypted user ID, encrypted password, encrypted new password, and encrypted security-sensitive data

## Communications database for the server

The *communications database* (CDB) is a set of DB2 catalog tables that let you control aspects of how requests leave a DB2 subsystem and how requests come in. Columns in the SYSIBM.LUNAMES and SYSIBM.USERNAMES tables pertain to security on the inbound side (the server).

### SYSIBM.LUNAMES columns

The SYSIBM.LUNAMES table is used only for requests that use SNA protocols.

#### GUIP

#### LUNAME CHAR(8)

The LUNAME of the remote system. A blank value identifies a default row that serves requests from any system that is not specifically listed elsewhere in the column.

#### SECURITY\_IN CHAR(1)

The *acceptance option* for a remote request from the corresponding LUNAME:

- V** The option is “verify.” An incoming request must include one of the following authentication entities:
- User ID and password
  - User ID and RACF PassTicket
  - User ID and RACF encrypted password (not recommended)
  - Kerberos security tickets
  - User ID and DRDA encrypted password
  - User ID, password, and new password
  - User ID and encrypted password, or encrypted user ID and encrypted password
- A** The option is “already verified.” This is the default. With A, a request does not need an authentication token, although the token is checked if it is sent.

With this option, an incoming connection request is accepted if it includes any of the following authentication tokens:

- User ID only
- All authentication methods that option V supports

If the USERNAMES column of SYSIBM.LUNAMES contains I or B, RACF is not invoked to validate incoming connection requests that contain only a user ID.

#### ENCRYPTPSWDS CHAR(1)

This column only applies to DB2 for z/OS or DB2 for z/OS partners when passwords are used as authentication tokens. It indicates whether passwords received from and sent to the corresponding LUNAME are encrypted:

- Y** Yes, passwords are encrypted. For outbound requests, the encrypted password is extracted from RACF and sent to the server. For inbound requests, the password is treated as if it is encrypted.
- N** No, passwords are not encrypted. This is the default; any character other than Y is treated as N. Specify N for CONNECT statements that contain a USER parameter.



**Recommendation:** When you connect to a DB2 for z/OS partner that is at Version 5 or a subsequent release, use RACF PassTickets (SECURITY\_OUT='R') instead of using passwords.

#### **USERNAMES CHAR(1)**

This column indicates whether an ID accompanying a remote request, sent from or to the corresponding LUNAME, is subject to translation and “come from” checking. When you specify I, O, or B, use the SYSIBM.USERNAMES table to perform the translation.

- I** An inbound ID is subject to translation.
- O** An outbound ID, sent to the corresponding LUNAME, is subject to translation.
- B** Both inbound and outbound IDs are subject to translation.
- blank** No IDs are translated.



#### **SYSIBM.USERNAMES columns**

The SYSIBM.USERNAMES table is used by both SNA and TCP/IP connections.



#### **TYPE CHAR(1)**

Indicates whether the row is used for inbound or outbound translation:

- S** The row is used to obtain the system authorization ID for establishing a trusted connection.
- I** The row applies to inbound IDs (not applicable for TCP/IP connections).
- O** The row applies to outbound IDs.

The field should contain only I or O. Any other character, including blank, causes the row to be ignored.

#### **AUTHID VARCHAR(128)**

An authorization ID that is permitted and perhaps translated. If blank, any authorization ID is permitted with the corresponding LINKNAME; all authorization IDs are translated in the same way. Outbound translation is not performed on CONNECT statements that contain an authorization ID for the value of the USER parameter.

#### **LINKNAME CHAR(8)**

Identifies the VTAM or TCP/IP network locations that are associated with this row. A blank value in this column indicates that this name translation rule applies to any TCP/IP or SNA partner.

If you specify a nonblank value for this column, one or both of the following situations must be true:

- A row exists in table SYSIBM.LUNAMES that has a LUNAME value that matches the LINKNAME value that appears in this column.
- A row exists in table SYSIBM.IPNames that has a LINKNAME value that matches the LINKNAME value that appears in this column.

#### **NEWAUTHID VARCHAR(128)**

The translated authorization ID. If blank, no translation occurs.

## Enabling change of user passwords

You can specify YES in the EXTENDED SECURITY field of the DSNTIPR installation panel so that DB2 can return information about errors and expired passwords to the DRDA requester.

When the DRDA requester is notified that the RACF password has expired, and the requester has implemented function to allow passwords to be changed, the requester can prompt the end user for the old password and a new password. The requester sends the old and new passwords to the DB2 server. This function is supported through DB2 Connect<sup>™</sup>.

With the extended security option, DB2 passes the old and new passwords to RACF. If the old password is correct, and the new password meets the installation's password requirements, the end user's password is changed and the DRDA connection request is honored.

When a user changes a password, the user ID, the old password, and the new password are sent to DB2 by the client system. The client system can optionally encrypt these three tokens before they are sent.

## Authorization failure code

If the EXTENDED SECURITY field is set to YES on the DSNTIPR installation panel, DB2 returns detailed reason codes to a DRDA client when a DDF connection request fails.

When using SNA protocols, the requester must have included support for extended security sense codes. One such product is DB2 Connect.

If the proper requester support is present, the requester generates SQLCODE -30082 (SQLSTATE '08001') with a specific indication for the failure. Otherwise, a generic security failure code is returned.

## Managing inbound SNA-based connection requests

Requests from a remote LU are subject to security checks before they come into contact with DB2. Those checks control what LUs can attach to the network and verify the identity of a partner LU.

In addition, DB2 itself imposes several checks before accepting an attachment request.

### Processing of remote attachment requests

The DB2 server completes a specific sequence of authentication process before accepting a remote attachment request that uses the SNA protocol.

1. As the following diagram shows, if the remote request has no authentication token, DB2 checks the security acceptance option in the SECURITY\_IN column of table SYSIBM.LUNAMES. No password is sent or checked for the plan or package owner that is sent from a DB2 subsystem.

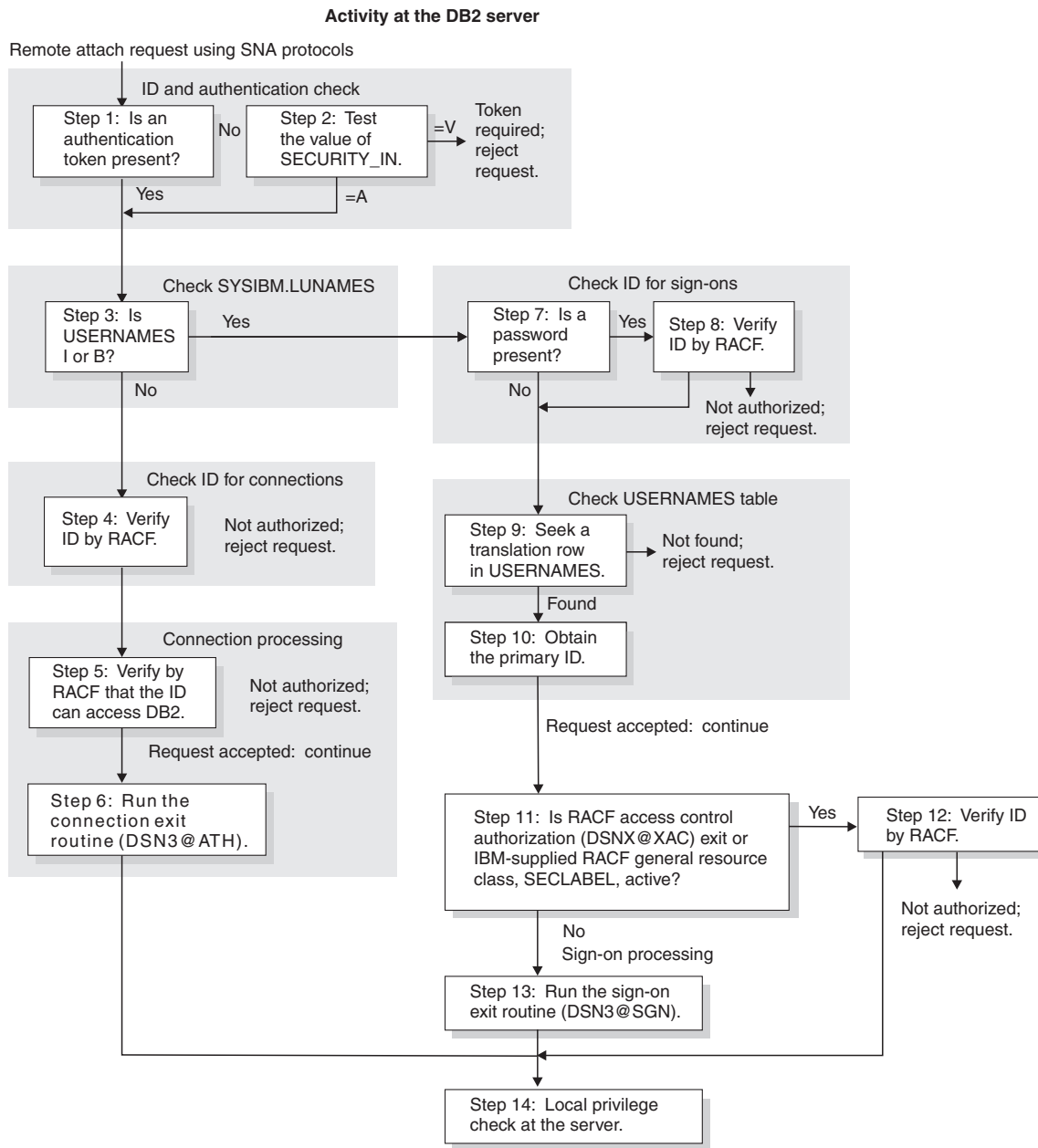


Figure 8. DB2 processing of remote attachment requests

2. If the acceptance option is “verify” (SECURITY\_IN = V), a security token is required to authenticate the user. DB2 rejects the request if the token missing.
3. If the USERNAMES column of SYSIBM.LUNAMES contains I or B, the authorization ID, and the plan or package owner that is sent by a DB2 subsystem, are subject to translation under control of the SYSIBM.USERNAMES table. If the request is allowed, it eventually goes through sign-on processing. If USERNAMES does not contain I or B, the authorization ID is not translated.
4. DB2 calls RACF by the RACROUTE macro with REQUEST=VERIFY to check the ID. DB2 uses the PASSCHK=NO option if no password is specified and ENCRYPT=YES if the ENCRYPTPSWDS column of SYSIBM.LUNAMES contains Y. If the ID, password, or PassTicket cannot be verified, DB2 rejects the request.

In addition, depending on your RACF environment, the following RACF checks may also be performed:

- If the RACF APPL class is active, RACF verifies that the ID has been given access to the DB2 APPL. The APPL resource that is checked is the LU name that the requester used when the attachment request was issued. This is either the local DB2 LU name or the generic LU name.
  - If the RACF APPCPORT class is active, RACF verifies that the ID is authorized to access z/OS from the Port of Entry (POE). The POE that RACF uses in the verify call is the requesting LU name.
5. The remote request is now treated like a local connection request with a DIST environment for the DSNR resource class. DB2 calls RACF by the RACROUTE macro with REQUEST=AUTH, to check whether the authorization ID is allowed to use DB2 resources that are defined to RACF.  
The RACROUTE macro call also verifies that the user is authorized to use DB2 resources from the requesting system, known as the port of entry (POE).
  6. DB2 invokes the connection exit routine. The parameter list that is passed to the routine describes where a remote request originated.
  7. If no password exists, RACF is not called. The ID is checked in SYSIBM.USERNAMES.
  8. If a password exists, DB2 calls RACF through the RACROUTE macro with REQUEST=VERIFY to verify that the ID is known with the password. ENCRYPT=YES is used if the ENCRYPTPSWDS column of SYSIBM.LUNAMES contains Y. If DB2 cannot verify the ID or password, the request is rejected.
  9. DB2 searches SYSIBM.USERNAMES for a row that indicates how to translate the ID. The need for a row that applies to a particular ID and sending location imposes a “come-from” check on the ID: If no such row exists, DB2 rejects the request.
  10. If an appropriate row is found, DB2 translates the ID as follows:
    - If a nonblank value of NEWAUTHID exists in the row, that value becomes the primary authorization ID.
    - If NEWAUTHID is blank, the primary authorization ID remains unchanged.
  11. The remote request is now treated like a local sign-on request. DB2 invokes the sign-on exit routine. The parameter list that is passed to the routine describes where a remote request originated.
  12. The remote request now has a primary authorization ID, possibly one or more secondary IDs, and an SQL ID. A request from a remote DB2 is also known by a plan or package owner. Privileges and authorities that are granted to those IDs at the DB2 server govern the actions that the request can take.

## Controlling LU attachments to the network

VTAM checks to prevent an unauthorized logical unit (LU) from attaching to the network and presenting itself to other LUs as an acceptable partner in communication. It requires each LU that attaches to the network to identify itself by a password.

If that requirement is in effect for your network, your DB2 subsystem, like every other LU on the network, must:

1. Choose a VTAM password.
2. Code the password with the PRTCT parameter of the VTAM APPL statement, when you define your DB2 to VTAM.

## Verifying partner LUs

RACF and VTAM check the identity of a logical unit (LU) that sends a request to your DB2 subsystem.

Perform the following steps to specify partner-LU verification:

1. Code `VERIFY=REQUIRED` on the VTAM APPL statement, when you define your DB2 to VTAM.
2. Establish a RACF profile for each LU from which you permit a request.

## Accepting remote attachment requests

When VTAM has established a conversation for a remote application, that application sends a remote request, which is a request to attach to your local DB2 subsystem.

Make sure that you do not confuse the remote request with a local attachment request that comes through one of the DB2 attachment facilities—IMS, CICS, TSO, and so on. A remote attachment request is defined by Systems Network Architecture and LU 6.2 protocols; specifically, it is an SNA Function Management Header 5.

In order to accept remote attachment requests, you must first define your DB2 to VTAM with the conversation-level security set to “already verified”. That is, you need to code `SECACPT=ALREADYV` on the VTAM APPL statement. The `SECACPT=ALREADYV` setting provides more options than does `SECACPT=CONV` or “conversation”, which is not recommended.

The primary tools for controlling remote attachment requests are entries in tables `SYSIBM.LUNAMES` and `SYSIBM.USERNAMES` in the communications database. You need a row in `SYSIBM.LUNAMES` for each system that sends attachment requests, a dummy row that allows any system to send attachment requests, or both. You might need rows in `SYSIBM.USERNAMES` to permit requests from specific IDs or specific LUNAMES, or to provide translations for permitted IDs.

## Managing inbound IDs through DB2

If you manage incoming IDs through DB2, you can avoid calls to RACF. You can accept many IDs by specifying them in a single row in the `SYSIBM.USERNAMES` table.

To manage incoming IDs through DB2, put an I in the `USERNAMES` column of `SYSIBM.LUNAMES` for the particular LU. If an O is already specified because you are also sending requests to that LU, change O to B. Attachment requests from that LU now go through the sign-on processing, and its IDs are subject to translation.

## Managing inbound IDs through RACF

If you manage incoming IDs through RACF, you must register every acceptable ID with RACF, and DB2 must call RACF to process every request.

You can use RACF or Kerberos can authenticate the user. Kerberos cannot be used if you do not have RACF on the system.

To manage incoming IDs through RACF, leave `USERNAMES` blank for that LU, or leave the O unchanged, if already specified. Requests from that LU now go through the connection processing, and its IDs are not subject to translation.

## Authenticating partner LUs

If RACF has already validated the identity of an LU and if you trust incoming IDs from the LU, you do not need to validate them by an authentication token.

Put an A in the SECURITY\_IN column of the row in SYSIBM.LUNAMES that corresponds to the other LU; your acceptance level for requests from that LU is now “already verified”. Requests from that LU are accepted without an authentication token. (In order to use this option, you must have defined DB2 to VTAM with SECACPT=ALREADYV.

If an authentication token does accompany a request, DB2 calls RACF to check the authorization ID against it. To require an authentication token from a particular LU, put a V in the SECURITY\_IN column in SYSIBM.LUNAMES; your acceptance level for requests from that LU is now “verify”. You must also register every acceptable incoming ID and its password with RACF.

**Performance considerations:** Each request to RACF to validate authentication tokens results in an I/O operation, which has a high performance cost.

**Recommendation:** To eliminate the I/O, allow RACF to cache security information in VLF. To activate this option, add the IRRACEE class to the end of z/OS VLF member COFVLFxx in SYS1.PARMLIB, as follows:

```
CLASS NAME(IRRACEE)
EMAJ (ACEE)
```

## Encrypting passwords

You can encrypt passwords to secure network connection requests.

You can encrypt passwords by using one of the following methods:

- RACF using PassTickets.
- DRDA password encryption support. DB2 for z/OS as a server supports DRDA encrypted passwords and encrypted user IDs with encrypted passwords.
- The SET ENCRYPTION PASSWORD statement. This encryption method should not be used for distributed access because the unencrypted passwords in the SET ENCRYPTION PASSWORD statement flow from the client to the server.

## Authenticating users through Kerberos

If your distributed environment uses Kerberos to manage users and perform user authentication, DB2 for z/OS can use Kerberos security services to authenticate remote users.

## Translating inbound IDs

Duplication of authorization IDs on different logical units (LUs) is a serious security exposure. For tighter security, make sure that each of the authorization IDs has the same meaning throughout your entire network.

**Example:** Suppose that the ID DBADM1 is known to the local DB2 and has DBADM authority over certain databases there; suppose also that the same ID exists in some remote LU. If an attachment request comes in from DBADM1, and if nothing is done to alter the ID, the wrong user can exercise privileges of DBADM1 in the local DB2. The way to protect against that exposure is to translate the remote ID into a different ID before the attachment request is accepted.

You must be prepared to translate the IDs of plan owners, package owners, and the primary IDs of processes that make remote requests. Do not plan to translate all IDs in the connection exit routine—the routine does not receive plan and package owner IDs.

If you have decided to manage inbound IDs through DB2, you can translate an inbound ID to some other value. Within DB2, you grant privileges and authorities only to the translated value. The “translation” is not affected by anything you do in your connection or sign-on exit routine. The *output* of the translation becomes the *input* to your sign-on exit routine.

**Recommendation:** Do not translate inbound IDs in an exit routine; translate them only through the SYSIBM.USERNAMES table.

The examples in the following table shows the possibilities for translation and how to control translation by SYSIBM.USERNAMES. You can use entries to allow requests only from particular LUs or particular IDs, or from combinations of an ID and an LU. You can also translate any incoming ID to another value.

*Table 39. Your SYSIBM.USERNAMES table. (Row numbers are added for reference.)*

Row	TYPE	AUTHID	LINKNAME	NEWAUTHID
1	I	blank	LUSNFRAN	blank
2	I	BETTY	LUSNFRAN	ELIZA
3	I	CHARLES	blank	CHUCK
4	I	ALBERT	LUDALLAS	blank
5	I	BETTY	blank	blank

The following table shows the search order of the SYSIBM.USERNAMES table.

*Table 40. Precedence search order for SYSIBM.USERNAMES table*

AUTHID	LINKNAME	Result
Name	Name	If NEWAUTHID is specified, AUTHID is translated to NEWAUTHID for the specified LINKNAME.
Name	Blank	If NEWAUTHID is specified, AUTHID is translated to NEWAUTHID for all LINKNAMEs.
Blank	Name	If NEWAUTHID is specified, it is substituted for AUTHID for the specified LINKNAME.
Blank	Blank	Unavailable resource message (SQLCODE -904) is returned.

DB2 searches SYSIBM.USERNAMES to determine how to translate for each of the requests that are listed in the following table.

*Table 41. How DB2 translates inbound authorization ids*

Request	How DB2 translates request
ALBERT requests from LUDALLAS	DB2 searches for an entry for AUTHID=ALBERT and LINKNAME=LUDALLAS. DB2 finds one in row 4, so the request is accepted. The value of NEWAUTHID in that row is blank, so ALBERT is left unchanged.



Table 41. How DB2 translates inbound authorization ids (continued)

Request	How DB2 translates request
BETTY requests from LUDALLAS	DB2 searches for an entry for AUTHID=BETTY and LINKNAME=LUDALLAS; none exists. DB2 then searches for AUTHID=BETTY and LINKNAME=blank. It finds that entry in row 5, so the request is accepted. The value of NEWAUTHID in that row is blank, so BETTY is left unchanged.
CHARLES requests from LUDALLAS	DB2 searches for AUTHID=CHARLES and LINKNAME=LUDALLAS; no such entry exists. DB2 then searches for AUTHID=CHARLES and LINKNAME=blank. The search ends at row 3; the request is accepted. The value of NEWAUTHID in that row is CHUCK, so CHARLES is translated to CHUCK.
ALBERT requests from LUSNFRAN	DB2 searches for AUTHID=ALBERT and LINKNAME=LUSNFRAN; no such entry exists. DB2 then searches for AUTHID=ALBERT and LINKNAME=blank; again no entry exists. Finally, DB2 searches for AUTHID=blank and LINKNAME=LUSNFRAN, finds that entry in row 1, and the request is accepted. The value of NEWAUTHID in that row is blank, so ALBERT is left unchanged.
BETTY requests from LUSNFRAN	DB2 finds row 2, and BETTY is translated to ELIZA.
CHARLES requests from LUSNFRAN	DB2 finds row 3 before row 1; CHARLES is translated to CHUCK.
WILBUR requests from LUSNFRAN	No provision is made for WILBUR, but row 1 of the SYSIBM.USERNAMES table allows any ID to make a request from LUSNFRAN and to pass without translation. The acceptance level for LUSNFRAN is "already verified", so WILBUR can pass without a password check by RACF. After accessing DB2, WILBUR can use only the privileges that are granted to WILBUR and to PUBLIC (for DRDA access).
WILBUR requests from LUDALLAS	Because the acceptance level for LUDALLAS is "verify" as recorded in the SYSIBM.LUNAMES table, WILBUR must be known to the local RACF. DB2 searches in succession for one of the combinations WILBUR/LUDALLAS, WILBUR/blank, or blank/LUDALLAS. None of those is in the table, so the request is rejected. The absence of a row permitting WILBUR to request from LUDALLAS imposes a "come-from" check: WILBUR can attach from some locations (LUSNFRAN), and some IDs (ALBERT, BETTY, and CHARLES) can attach from LUDALLAS, but WILBUR cannot attach if coming from LUDALLAS.

In the process of accepting remote attachment requests, any step that calls RACF is likely to have a relatively high performance cost. To trade some of that cost for a somewhat greater security exposure, have RACF check the identity of the other LU just once. Then trust the partner LU, translating the inbound IDs and not requiring or using passwords. In this case, no calls are made to RACF from within DB2; the penalty is only that you make the partner LU responsible for verifying IDs.

If you update tables in the CDB while the distributed data facility is running, the changes might not take effect immediately. If incoming authorization IDs are managed through DB2 and if the ICSF is installed and properly configured, you can use the DSNLEUSR stored procedure to encrypt translated authorization IDs

and store them in the NEWAUTHID column of the SYSIBM.USERNAMES table. DB2 decrypts the translated authorization IDs during connection processing.

### **Associating inbound IDs with secondary IDs**

Your decisions on password encryption and ID translation determine the value that you use for the primary authorization ID on an attachment request.

They also determine whether those requests are next treated as connection requests or as sign-on requests. That means that the remote request next goes through the same processing as a local request, and that you have the opportunity to associate the primary ID with a list of secondary IDs in the same way you do for local requests.

## **Managing inbound TCP/IP-based connection requests**

DRDA connections that use TCP/IP have fewer security controls than do connections that use SNA protocols. When planning to control inbound TCP/IP connection requests, you must decide whether you want the requests to have authentication information, such as RACF passwords, RACF PassTickets, and Kerberos tickets, passed along with authorization IDs.

If you require authentication, specify NO on the TCP/IP ALREADY VERIFIED field of installation panel DSN TIP5, which is the default option, to indicate that you require this authentication information. Also, ensure that the security subsystem at your server is properly configured to handle the authentication information that is passed to it. If you do not specify NO, all incoming TCP/IP requests can connect to DB2 without any authentication.

- For requests that use RACF passwords or PassTickets, enter the following RACF command to indicate which user IDs that use TCP/IP are authorized to access DDF (the distributed data facility address space):

```
PERMIT ssnm.DIST CLASS(DSNR) ID(yyy) ACCESS(READ)
      WHEN(APPCPORT(TCPIP))
```

Consider the following questions:

***Do you permit access by TCP/IP?*** If the serving DB2 for z/OS subsystem has a DRDA port and resynchronization port specified in the BSDS, DB2 is enabled for TCP/IP connections.

***Do you manage inbound IDs through DB2 or RACF?*** All IDs must be passed to RACF or Kerberos for processing. No option exists to handle incoming IDs through DB2.

***Do you trust the partner?*** TCP/IP does not verify partner LUs as SNA does. If your requesters support mutual authentication, use Kerberos to handle this on the requester side.

***If you use passwords, are they encrypted?*** Passwords can be encrypted through:

- RACF using PassTickets
- DRDA password encryption support. DB2 for z/OS as a server supports DRDA encrypted passwords and encrypted user IDs with encrypted passwords.

***If you use Kerberos, are users authenticated?*** If your distributed environment uses Kerberos to manage users and perform user authentication, DB2 for z/OS can use Kerberos security services to authenticate remote users.

***Do you translate inbound IDs?*** Inbound IDs are not translated when you use TCP/IP.

*How do you associate inbound IDs with secondary IDs?* To associate an inbound ID with secondary IDs, modify the default connection exit routine (DSN3@ATH). TCP/IP requests do not use the sign-on exit routine.

- To receive requests from a DB2 for z/OS requester over TCP/IP connections that use RACF-protected user IDs and RACF PassTickets (as passwords), you must take the following additional actions in RACF:

1. Create a RACF PTKTDATA resource profile at the server system or sysplex by issuing one of the following commands:

```
RDEFINE PTKTDATA IRRPTAUTH.applname.userid
RDEFINE PTKTDATA IRRPTAUTH.applname.*
```

where

- *applname* is either the generic LU name, IPNAME assigned to each member of a serving data sharing group, or LUNAME or IPNAME assigned to the serving non-data sharing subsystem
- *userid* is either an asterisk ("\*") or a RACF-protected user ID that you want to allow into the serving subsystem or a member of a data sharing group.

2. Refresh and load the PTKTDATA resource profile by issuing the following command:

```
SETROPTS RACLIST(PTKTDATA) REFRESH
```

3. Permit the user ID that is assigned in STDATA of the STARTED profile in the ssidDIST address space to read the new profile by issuing one of the following commands:

```
PERMIT IRRPTAUTH.applname.userid CLASS(PTKTDATA) -
  ID(dist_userid) ACCESS(READ)
PERMIT IRRPTAUTH.applname.* CLASS(PTKTDATA) -
  ID(dist_userid) ACCESS(READ)
```

where *userid* and *dist\_userid* are not the same

You do not need to take these additional actions in RACF if RACF-protected user IDs are not used in connection requests from a DB2 for z/OS requester to a DB2 for z/OS server.

## Processing of TCP/IP-based connection requests

The DB2 server completes a sequence of authentication tasks when handling a remote connection request that uses the TCP/IP protocol.

1. As the following diagram shows, DB2 checks to see if an authentication token (RACF encrypted password, RACF PassTicket, DRDA encrypted password, or Kerberos ticket) accompanies the remote request.

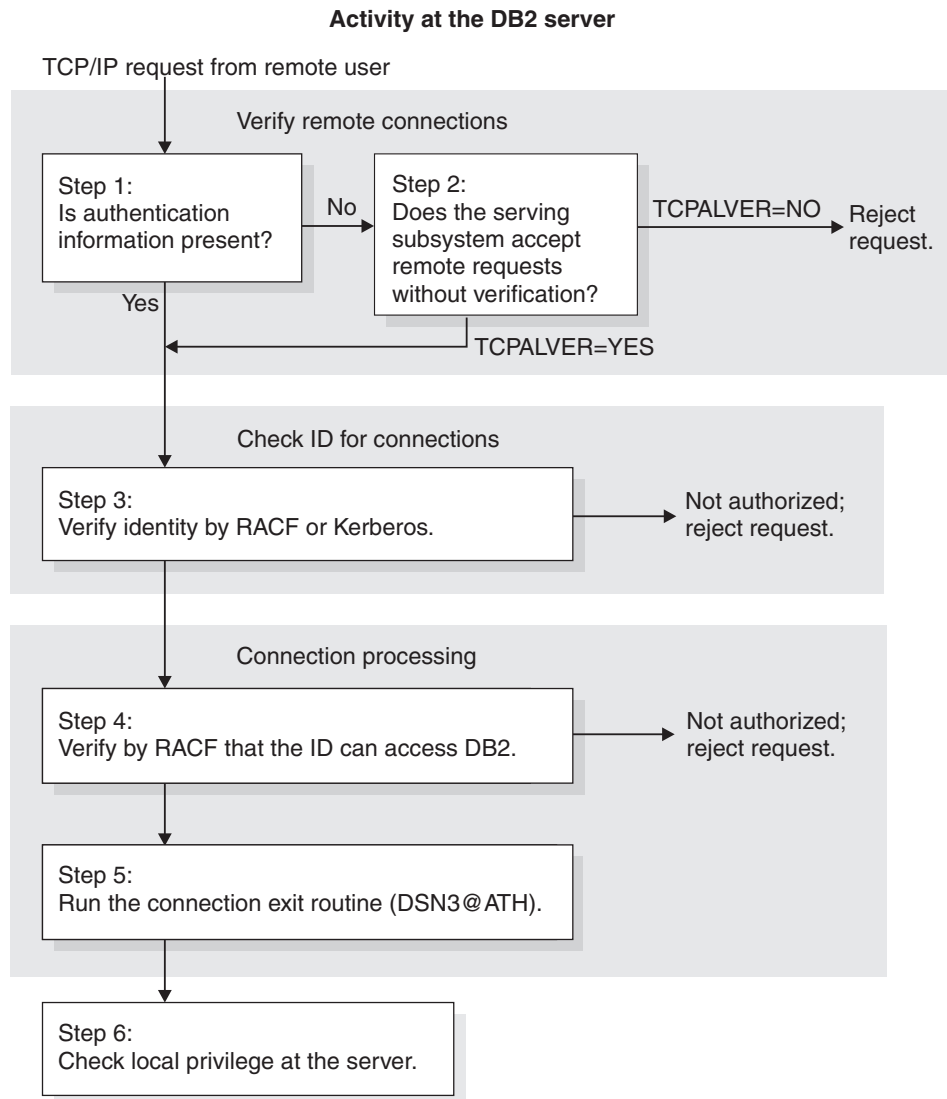


Figure 9. DB2 processing of TCP/IP-based connection requests

2. If no authentication token is supplied, DB2 checks the TCPALVER subsystem parameter to see if DB2 accepts IDs without authentication information.
  - If TCPALVER=NO | SERVER, DB2 requires the minimum of a userid and a password.
  - If TCPALVER=SERVER\_ENCRYPT, DB2 requires a userid and a password. In addition, it requires that the security credentials be AES-encrypted or that the connection is accepted on a port that ensures AT-TLS policy protection, such as a DB2 Security Port (SECPORT). Kerberos tickets are accepted. RACF PassTickets, or non-encrypted security credentials, are accepted only when the connection is secured by the TCP/IP network.
  - If TCPALVER=YES | CLIENT, DB2 accepts TCP/IP connection requests that contain only a userid.
3. The identity is a RACF ID that is authenticated by RACF if a password or PassTicket is provided, or the identity is a Kerberos principal that is validated by Kerberos Security Server, if a Kerberos ticket is provided. Ensure that the ID is defined to RACF in all cases. When Kerberos tickets are used, the RACF ID

is derived from the Kerberos principal identity. To use Kerberos tickets, ensure that you map Kerberos principal names with RACF IDs.

In addition, depending on your RACF environment, the following RACF checks may also be performed:

- If the RACF APPL class is active, RACF verifies that the ID has access to the DB2 APPL. The APPL resource that is checked is the LU name that the requester used when the attachment request was issued. This is either the local DB2 LU name or the generic LU name.
- If the RACF APPCPORT class is active, RACF verifies that the ID is authorized to access z/OS from the port of entry (POE). The POE that RACF uses in the RACROUTE VERIFY call depends on whether all the following conditions are true:
  - The current operating system is z/OS V1.5 or later
  - The TCP/IP Network Access Control is configured
  - The RACF SERVAUTH class is active

If all these conditions are true, RACF uses the remote client's POE security zone name that is defined in the TCP/IP Network Access Control file. If one or more of these conditions is not true, RACF uses the literal string 'TCPIP'.

If this is a request to change a password, the password is changed.

4. The remote request is now treated like a local connection request (using the DIST environment for the DSNR resource class). DB2 calls RACF to check the ID's authorization against the *ssnm.DIST* resource.
5. DB2 invokes the connection exit routine. The parameter list that is passed to the routine describes where the remote request originated.
6. The remote request has a primary authorization ID, possibly one or more secondary IDs, and an SQL ID. (The SQL ID cannot be translated.) The plan or package owner ID also accompanies the request. Privileges and authorities that are granted to those IDs at the DB2 server govern the actions that the request can take.

#### Related reference

"Processing of outbound connection requests" on page 180

## Managing denial-of-service attacks

With DB2, you can manage denial-of-service attacks in the network connections to a DB2 server.

The most common type of denial-of-service attack occurs when an attacker "floods" a network with connection requests to a DB2 server. If this occurs, the attacker quickly exhausts the threshold for the maximum number of remote connections that are defined to the DB2 server system. As a result, no additional remote connections can be accepted by the DB2 server, including those from legitimate client systems.

To prevent the typical denial-of-service attacks, DB2 monitors the traffic of inbound connections and terminates those that don't contain data for establishing a valid connection.

## Preventing SQL injection attacks

SQL injection attacks might occur when dynamic SQL statements are constructed from user input and the input is inadequately checked. You can use several techniques to prevent or reduce SQL injection attacks.

To eliminate or reduce the risk of SQL injection attacks:

- Avoid dynamic SQL, whenever possible.
- Use pureQuery or SQLJ rather than JDBC for Java.
- Use system security techniques, such as views and access control mechanisms, whenever possible.

Understand the limitations of security within an application. System security can use security and integrity mechanisms that are not available to application programs. The level of assurance that can be provided in system security can be much higher. If the applications are run on the client or have fewer protection layers and firewalls than the database, make sure to address those limitations.

- Use row permissions and column masks to protect data even if the statement is compromised by SQL injection attacks.
- Put input data into host variables with just the value or use a parameter marker in dynamic SQL.
- Make sure to check all input:
  - Check that the input is the intended data type and format. This is generally required for all programs to ensure that they work properly but especially crucial for data intended as part of an SQL statement.
  - Accept numbers for a numeric comparison only.
  - Do not allow special characters if they do not apply.

## Managing outbound connection requests

If you plan to send requests to another DB2 subsystem, you need to consider the subsystem's security measures for network connections. You need to know what those measures are and make entries in your CDB to correspond to them.

If you are planning to send remote requests to a DBMS that is not DB2 for z/OS, you need to satisfy the requirements of that system.

DB2 chooses how to send authentication tokens based on the network protocols that are used (SNA or TCP/IP). If the request is sent using SNA, the authentication tokens are sent in the SNA attachment request (FMH5), unless you are using Kerberos. If you use Kerberos, authentication tokens are sent with DRDA security commands. If the request uses TCP/IP, the authentication tokens are always sent using DRDA security commands.

At least one authorization ID is always sent to the server to be used for authentication. That ID is one of the following values:

- The primary authorization ID of the process.
- If you connect to the server using a CONNECT statement with the USER keyword, the ID that you specify as the USER ID. The CONNECT statement allows non-RACF user IDs on the USER keyword. If connecting to a remote location, the user ID is not authenticated by RACF.

However, other IDs can accompany some requests. You need to understand what other IDs are sent because they are subject to translation. You must include these other IDs in table SYSIBM.USERNAMES to avoid an error when you use outbound translation. The following table shows the IDs that you send in the different situations:

Table 42. IDs that accompany the primary ID on a remote request

In this situation:	You send this ID also:
An SQL query, using DB2 DRDA-protocol access	The plan owner
A remote BIND, COPY, or REBIND PACKAGE command	The package owner

If the connection is to a remote non-DB2 for z/OS server using DRDA protocol and if the outbound translation is specified, a row for the plan owner in the USERNAMES table is optional.

### Communications database for the requester

The *communications database* (CDB) is a set of DB2 catalog tables that let you control aspects of remote requests. Columns in the SYSIBM.LUNAMES, SYSIBM.IPNames, SYSIBM.USERNAMES, and SYSIBM.LOCATIONS tables pertain to security that related to the requesting system.

#### SYSIBM.LUNAMES columns:

The SYSIBM.LUNAMES table is used only for outbound requests that use SNA protocols.

#### GUPI

#### LUNAME CHAR(8)

The LUNAME of the remote system. A blank value identifies a default row that serves requests from any system that is not specifically listed elsewhere in the column.

#### SECURITY\_OUT (CHAR 1)

Indicates the security option that is used when local DB2 SQL applications connect to any remote server that is associated with the corresponding LUNAME.

- A** The letter A signifies the security option of already verified, and it is the default. With A, outbound connection requests contain an authorization ID and no authentication token. The value that is used for an outbound request is either the DB2 user's authorization ID or a translated ID, depending on the value in the USERNAMES column.
- R** The letter R signifies the RACF PassTicket security option. Outbound connection requests contain a user ID and a RACF PassTicket. The LUNAME column is used as the RACF PassTicket application name.  
  
The value that is used for an outbound request is either the DB2 user's authorization ID or a translated ID, depending on the value in the USERNAMES column. The translated ID is used to build the RACF PassTicket. Do not specify R for CONNECT statements with a USER parameter.
- P** The letter P signifies the password security option. Outbound connection requests contain an authorization ID and a password. The password is obtained from RACF if ENCRYPTPSWDS=Y, or from SYSIBM.USERNAMES if ENCRYPTPSWDS=N. If you get the password from SYSIBM.USERNAMES, the USERNAMES column



of SYSIBM.LUNAMES must contain B or O. The value that is used for an outbound request is the translated ID.

#### ENCRYPTPSWDS CHAR(1)

Indicates whether passwords received from and sent to the corresponding LUNAME are encrypted. This column only applies to DB2 for z/OS and DB2 for z/OS partners when passwords are used as authentication tokens.

- Y** Yes, passwords are encrypted. For outbound requests, the encrypted password is extracted from RACF and sent to the server. For inbound requests, the password is treated as encrypted.
- N** No, passwords are not encrypted. This is the default; any character but Y is treated as N.

**Recommendation:** When you connect to a DB2 for z/OS partner that is at Version 5 or a subsequent release, use RACF PassTickets (SECURITY\_OUT='R') instead of encrypting passwords.

#### USERNAMES CHAR(1)

Indicates whether an ID accompanying a remote attachment request, which is received from or sent to the corresponding LUNAME, is subject to translation and “come from” checking. When you specify I, O, or B, use the SYSIBM.USERNAMES table to perform the translation.

- I** An inbound ID is subject to translation.
- O** An outbound ID, sent to the corresponding LUNAME, is subject to translation.
- B** Both inbound and outbound IDs are subject to translation.
- blank** No IDs are translated.

#### GUPI

#### SYSIBM.IPNAMES columns:

The SYSIBM.IPNAMES table is used only for outbound requests that use TCP/IP protocols.

#### GUPI

#### LINKNAME CHAR(8)

The name used in the LINKNAME column of SYSIBM.LOCATIONS to identify the remote system.

#### IPADDR

Specifies an IP address or domain name of a remote TCP/IP host.

#### SECURITY\_OUT

Indicates the DRDA security option that is used when local DB2 SQL applications connect to any remote server that is associated with this TCP/IP host.

- A** The letter A signifies the security option of already verified, and it is the default. Outbound connection requests contain an authorization ID and no password. The value that is used for an outbound request is either the DB2 user's authorization ID or a translated ID, depending on the value in the USERNAMES column.

The authorization ID is not encrypted when it is sent to the partner. For encryption, see option D.

- R** The letter R signifies the RACF PassTicket security option. Outbound connection requests contain a user ID and a RACF PassTicket. When a RACF PassTicket is generated, the LINKNAME column value is used as the RACF PassTicket application name and must match the following at the target server
- LUNAME - if the remote site is a DB2 subsystem that is defined with only an LUNAME value and no GENERIC LU name value or IPNAME value
  - GENERIC - if the remote site is a DB2 subsystem that is defined with a GENERIC LU name value in addition to an LUNAME value but no IPNAME value
  - IPNAME - if the remote site is a DB2 subsystem that is defined with an IPNAME value that triggers the remote DB2 subsystem's DDF to activate only its TCP/IP communications support.

The value that is used for an outbound request is either the DB2 user's authorization ID or a translated ID, depending on the value in the USERNAMES column. The translated ID is used to build the RACF PassTicket. Do not specify R for CONNECT statements with a USER parameter.

The authorization ID is not encrypted when it is sent to the partner.

- D** The letter D signifies the security option of user ID and security-sensitive data encryption. Outbound connection requests contain an authorization ID and no password. The authorization ID that is used for an outbound request is either the DB2 user's authorization ID or a translated ID, depending on the USERNAMES column.

This option indicates that the user ID and the security-sensitive data are to be encrypted. If you do not require encryption, see option A.

- E** The letter E signifies the security option of user ID, password, and security-sensitive data encryption. Outbound connection requests contain an authorization ID and a password. The password is obtained from the SYSIBM.USERNAMES table. The USERNAMES column must specify "O".

This option indicates that the user ID, password, and security-sensitive data are to be encrypted. If you do not require security-sensitive data encryption, see option P.

- P** The letter P signifies the password security option. Outbound connection requests contain an authorization ID and a password. The password is obtained from the SYSIBM.USERNAMES table. If you specify P, the USERNAMES column must specify "O".

If you specify P and the server supports encryption, the user ID and the password are encrypted. If the server does not support encryption, the user ID and the password are sent to the partner in clear text. If you also need to encrypt security-sensitive data, see option E.

## USERNAMES CHAR(1)

This column indicates whether an outbound request translates the authorization ID. When you specify O, use the SYSIBM.USERNAMES table to perform the translation.

- O** The letter O signifies an outbound ID that is subject to translation. Rows in the SYSIBM.USERNAMES table are used to perform ID translation. If a connection to any remote server is to be established as trusted, a row in the SYSIBM.USERNAMES table is used to obtain the system authorization ID.
- S** The letter S signifies the system authorization ID, within a trusted context, obtained from the SYSIBM.USERNAMES table. If the system authorization ID that is specified in the AUTHID column is different from the primary authorization ID, DB2 sends the user switch request on behalf of the primary authorization ID after successfully establishing the trusted connection.

**blank** No translation is done.

### GUPI

## SYSIBM.USERNAMES columns:

The SYSIBM.USERNAMES table is used by outbound connection requests that use SNA and TCP/IP protocols.

### GUPI

## TYPE CHAR(1)

Indicates whether the row is used for inbound or outbound translation:

- S** The row is used to obtain the outbound system authorization ID for establishing a trusted connection.
- I** The row applies to inbound IDs.
- O** The row applies to outbound IDs.

The field should contain only I, O, or S. Any other character, including blank, causes the row to be ignored.

## AUTHID VARCHAR(128)

An authorization ID that is permitted and perhaps translated. If blank, any authorization ID is permitted with the corresponding LINKNAME, and all authorization IDs are translated in the same way.

## LINKNAME CHAR(8)

Identifies the VTAM or TCP/IP network locations that are associated with this row. A blank value in this column indicates that this name translation rule applies to any TCP/IP or SNA partner.

If you specify a nonblank value for this column, one or both of the following situations must be true:

- A row exists in table SYSIBM.LUNAMES that has a LUNAME value that matches the LINKNAME value that appears in this column.
- A row exists in table SYSIBM.IPNames that has a LINKNAME value that matches the LINKNAME value that appears in this column.

### **NEWAUTHID VARCHAR(128)**

The translated authorization ID. If blank, no translation occurs.

### **PASSWORD VARCHAR(255)**

A password or password phrase that is sent with an outbound request if passwords or phrases are not encrypted by RACF. The column is not used if passwords or phrases are encrypted or if the row is for inbound requests. A password or password phrase can be stored as encrypted data by calling the DSNLEUSR stored procedure. To send the encrypted value of the PASSWORD column through a network, you must specify one of the encryption options in the SYSIBM.IPNAMES table.

#### **GUPI**

### **SYSIBM.LOCATIONS columns:**

The SYSIBM.LOCATIONS table contains a row for every accessible remote server. Each row associates a LOCATION name with the TCP/IP or SNA network attributes for the remote server. Requesters are not defined in the SYSIBM.LOCATIONS table.

#### **GUPI**

### **LOCATION CHAR(16)**

Indicates the unique location name by which the the remote server is known to local DB2 SQL applications.

### **LINKNAME CHAR(8)**

Identifies the VTAM or TCP/IP network locations that are associated with this row. A blank value in this column indicates that this name translation rule applies to any TCP/IP or SNA partner.

If you specify a nonblank value for this column, one or both of the following situations must be true:

- A row exists in table SYSIBM.LUNAMES that has an LUNAME value that matches the LINKNAME value that appears in this column.
- A row exists in table SYSIBM.IPNAMES that has a LINKNAME value that matches the LINKNAME value that appears in this column.

### **PORT CHAR(32)**

Indicates that TCP/IP is used for outbound connections when the following statement is true:

- A row exists in SYSIBM.IPNAMES, where the LINKNAME column matches the value that is specified in the SYSIBM.LOCATIONS LINKNAME column.

If the previously mentioned row is found, and the SECURE column has a value of 'N', the value of the PORT column is interpreted as follows:

- If PORT is blank, the default DRDA port (446) is used.
- If PORT is nonblank, the value that is specified for PORT can take one of two forms:
  - If the value in PORT is left-justified with one to five numeric characters, the value is assumed to be the TCP/IP port number of the remote database server.

- Any other value is assumed to be a TCP/IP service name, which you can convert to a TCP/IP port number by using the TCP/IP getservbyname socket call. TCP/IP service names are not case-sensitive.

If the previously mentioned row is found, and the SECURE column has a value of 'Y', the value of the PORT column is interpreted as follows:

- If PORT is blank, the default secure DRDA port (448) is used.
- If PORT is nonblank, the value that is specified for PORT takes the value of the configured secure DRDA port at the remote server.

#### **TPN VARCHAR(64)**

Used only when the local DB2 begins an SNA conversation with another server. When used, TPN indicates the SNA LU 6.2 transaction program name (TPN) that will allocate the conversation. A length of zero for the column indicates the default TPN. For DRDA conversations, this is the DRDA default, which is X'07F6C4C2'.

#### **DBALIAS(128)**

Used to access a remote database server. If DBALIAS is blank, the location name is used to access the remote database server. This column does not change the name of any database objects sent to the remote site that contains the location qualifier.

#### **TRUSTED**

Indicates whether the connection to the remote server can be trusted. This is restricted to TCP/IP only. This column is ignored for connections that use SNA.

- |          |   |
|----------|---|
| <b>Y</b> | The location is trusted. Access to the remote location requires a trusted context that is defined at the remote location. |
| <b>N</b> | The location is not trusted.  |

#### **SECURE**

Indicates the use of the Secure Socket Layer (SSL) protocol for outbound connections when local DB2 applications connect to the remote database server by using TCP/IP.

- |          |  |
|----------|--|
| <b>Y</b> | A secure SSL connection is required for the outbound connection. |
| <b>N</b> | A secure connection is not required for the outbound connection. |



### **Processing of outbound connection requests**

A DB2 subsystem completes a sequence of tasks when sending out a connection request.

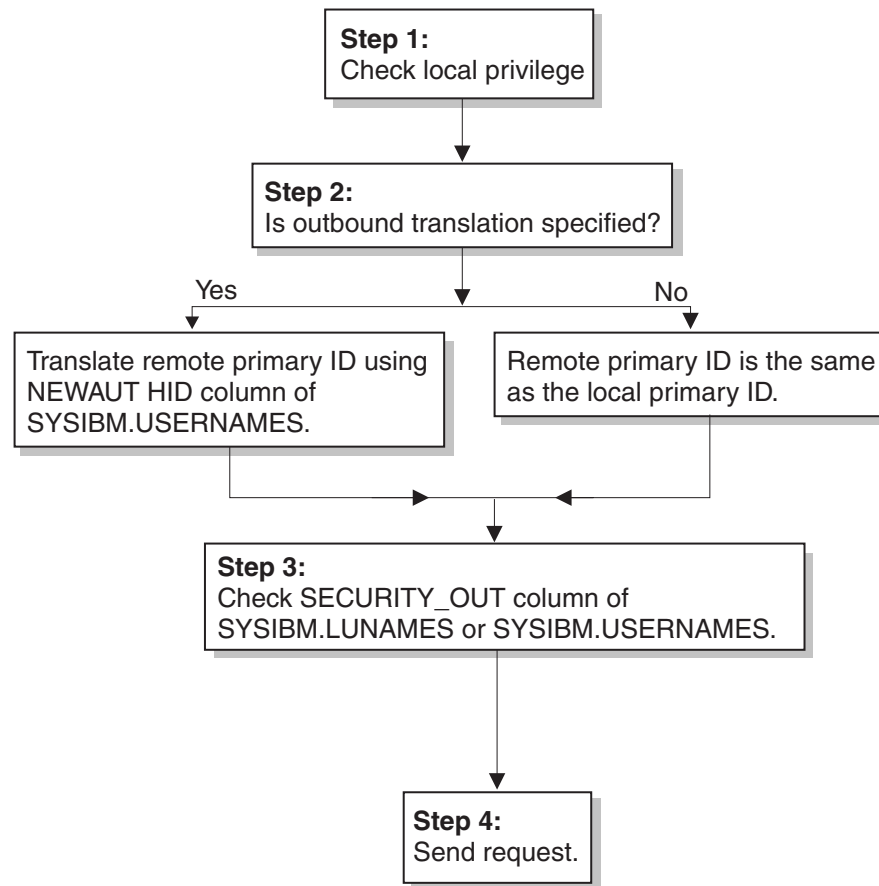


Figure 10. Steps in sending a request from a DB2 subsystem

1. The DB2 subsystem that sends the request checks whether the primary authorization ID has the privilege to execute the plan or package.  
DB2 determines which value in the LINKNAME column of the SYSIBM.LOCATIONS table matches either the LUNAME column in the SYSIBM.LUNAMES table or the LINKNAME column in the SYSIBM.IPNAMES table. This check determines whether SNA or TCP/IP protocols are used to carry the DRDA request.
2. When a plan is executed, the authorization ID of the plan owner is sent with the primary authorization ID. When a package is bound, the authorization ID of the package owner is sent with the primary authorization ID. If the USERNAMES column of the SYSIBM.LUNAMES table contains O or B, or if the USERNAMES column of the SYSIBM.IPNAMES table contains O, both IDs are subject to translation under control of the SYSIBM.USERNAMES table. Ensure that these IDs are included in SYSIBM.USERNAMES, or SQLCODE -904 is issued. DB2 translates the ID as follows:
  - If a nonblank value of NEWAUTHID is in the row, that value becomes the new ID.
  - If NEWAUTHID is blank, the ID is not changed.

If the SYSIBM.USERNAMES table does not contain a new authorization ID to which the primary authorization ID is translated, the request is rejected with SQLCODE -904.

If the USERNAMES column does not contain O or B, the IDs are not translated.

3. SECURITY\_OUT is checked for outbound security options as shown in the following diagram.

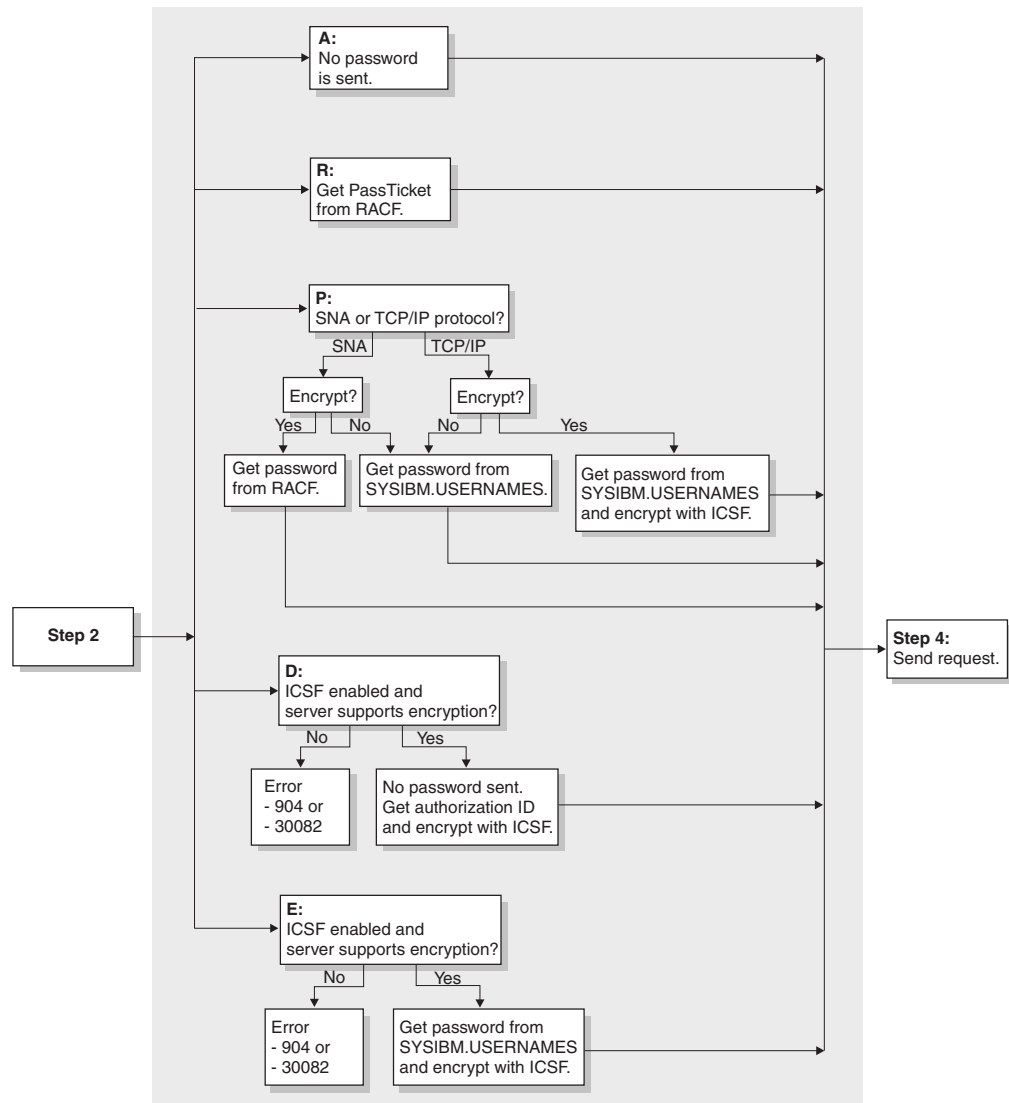


Figure 11. Details of Step 3

- A** Already verified. No password is sent with the authorization ID. This option is valid only if the server accepts already verified requests.
- For SNA, the server must have specified A in the SECURITY\_IN column of SYSIBM.LUNAMES.
  - For TCP/IP, the server must have specified YES in the TCP/IP ALREADY VERIFIED field of installation panel DSNTIP5.
- R** RACF PassTicket. If the primary authorization ID was translated, that translated ID is sent with the PassTicket.
- P** Password. The outbound request must be accompanied by a password:
- If the requester is DB2 for z/OS and uses SNA protocols, passwords can be encrypted if you specify Y in the ENCRYPTPSWDS column of SYSIBM.LUNAMES. If passwords are encrypted, the password is obtained from RACF. If passwords are not encrypted, the password is obtained from the PASSWORD column of SYSIBM.USERNAMES.



- If the requester uses TCP/IP protocols, the password is obtained from the PASSWORD column of SYSIBM.USERNAMES. If the Integrated Cryptographic Service Facility is enabled and properly configured and the server supports encryption, the password is encrypted.

**Recommendation:** Use RACF PassTickets to avoid sending unencrypted passwords over the network.

- D** User ID and security-sensitive data encryption. No password is sent with the authorization ID. If the Integrated Cryptographic Service Facility (ICSF) is enabled and properly configured and the server supports encryption, the authorization ID is encrypted before it is sent. If the ICSF is not enabled or properly configured, SQL return code -904 is returned. If the server does not support encryption, SQL return code -30082 is returned.
- E** User ID, password, and security-sensitive data encryption. If the ICSF is enabled and properly configured and the server supports encryption, the password is encrypted before it is sent. If the ICSF is not enabled or properly configured, SQL return code -904 is returned. If the server does not support encryption, SQL return code -30082 is returned.

4. Send the request.

#### Related reference

“Processing of TCP/IP-based connection requests” on page 171

## Translating outbound IDs

If an ID on your system is duplicated on a remote system, you can translate outbound IDs to avoid confusion. You can also translate IDs to ensure that they are accepted by the remote system.

To indicate that you want to translate outbound user IDs, perform the following steps:

1. Specify an O in the USERNAMES column of table SYSIBM.IPNAMES or SYSIBM.LUNAMES.
2. Use the NEWAUTHID column of SYSIBM.USERNAMES to specify the ID to which the outbound ID is translated.

**Example 1:** Suppose that the remote system accepts from you only the IDs XXGALE, GROUP1, and HOMER.

1. Specify that outbound translation is in effect for the remote system LUXXX by specifying in SYSIBM.LUNAMES the values that are shown in the following table.

*Table 43. SYSIBM.LUNAMES to specify that outbound translation is in effect for the remote system LUXXX*

LUNAME	USERNAMES
LUXXX	O

If your row for LUXXX already has I for the USERNAMES column (because you translate inbound IDs that come from LUXXX), change I to B for both inbound and outbound translation.

2. Translate the ID GALE to XXGALE on all outbound requests to LUXXX by specifying in SYSIBM.USERNAMES the values that are shown in the following table.

Table 44. Values in SYSIBM.USERNAMES to translate GALE to XXGALE on outbound requests to LUXXX

TYPE	AUTHID	LINKNAME	NEWAUTHID	PASSWORD
O	GALE	LUXXX	XXGALE	GALEPASS

- Translate EVAN and FRED to GROUP1 on all outbound requests to LUXXX by specifying in SYSIBM.USERNAMES the values that are shown in the following table.

Table 45. Values in SYSIBM.USERNAMES to translate EVAN and FRED to GROUP1 on outbound requests to LUXXX

TYPE	AUTHID	LINKNAME	NEWAUTHID	PASSWORD
O	EVAN	LUXXX	GROUP1	GRP1PASS
O	FRED	LUXXX	GROUP1	GRP1PASS

- Do not translate the ID HOMER on outbound requests to LUXXX. (HOMER is assumed to be an ID on your DB2, and on LUXXX.) Specify in SYSIBM.USERNAMES the values that are shown in the following table.

Table 46. Values in SYSIBM.USERNAMES to not translate HOMER on outbound requests to LUXXX

TYPE	AUTHID	LINKNAME	NEWAUTHID	PASSWORD
O	HOMER	LUXXX	(blank)	HOMERSPW

- Reject any requests from BASIL to LUXXX before they are sent. To do that, leave SYSIBM.USERNAMES empty. If no row indicates what to do with the ID BASIL on an outbound request to LUXXX, the request is rejected.

**Example 2:** If you send requests to another LU, such as LUYYY, you generally need another set of rows to indicate how your IDs are to be translated on outbound requests to LUYYY.

However, you can use a single row to specify a translation that is to be in effect on requests to all other LUs. For example, if HOMER is to be sent untranslated everywhere, and DOROTHY is to be translated to GROUP1 everywhere, specify in SYSIBM.USERNAMES the values that are shown in the following table.

Table 47. Values in SYSIBM.USERNAMES to not translate HOMER and to translate DOROTHY to GROUP1

TYPE	AUTHID	LINKNAME	NEWAUTHID	PASSWORD
O	HOMER	(blank)	(blank)	HOMERSPW
O	DOROTHY	(blank)	GROUP1	GRP1PASS

You can also use a single row to specify that all IDs that accompany requests to a single remote system must be translated. For example, if every one of your IDs is to be translated to THEIRS on requests to LUYYY, specify in SYSIBM.USERNAMES the values that are shown in the following table.

Table 48. Values in SYSIBM.USERNAMES to translate every ID to THEIRS

TYPE	AUTHID	LINKNAME	NEWAUTHID	PASSWORD
O	(blank)	LUYYY	THEIR	THEPASS

If the ICSF is installed and properly configured, you can use the DSNLEUSR stored procedure to encrypt the translated outbound IDs that are specified in the NEWAUTHID column of SYSIBM.USERNAMES. DB2 decrypts the translated outbound IDs during connection processing.

## Sending passwords or password phrases

DB2 provides several security mechanisms to send password or password phrase information.

Specifically, DB2 supports the following security mechanisms:

- RACF encrypted passwords
- RACF PassTickets
- Kerberos tickets
- DRDA-encrypted passwords or password phrases or DRDA-encrypted user IDs with encrypted passwords or password phrases.

If you have to send passwords or password phrases through the network, you can put the password or password phrase for a user ID in the PASSWORD column of the SYSIBM.USERNAMES table.

**Recommendation:** Use the DSNLEUSR stored procedure to encrypt passwords or password phrases in SYSIBM.USERNAMES. If the ICSF is installed and properly configured, you can use the DSNLEUSR stored procedure to encrypt passwords or password phrases in the SYSIBM.USERNAMES table. DB2 decrypts the password or password phrase during connection processing.

DB2 for z/OS allows the use of RACF encrypted passwords or RACF PassTickets. However, workstations, such as Windows workstations, do not support these security mechanisms. RACF encrypted passwords are not a secure mechanism because they can be replayed. RACF PassTickets are not compatible with SECURITY\_ENCRYPT; they are allowed only when the connections are secured by the TCP/IP network.

**Recommendation:** Do not use RACF encrypted passwords unless you are connecting to a previous release of DB2 for z/OS.

## Sending RACF-encrypted passwords

For DB2 subsystems that use SNA protocols to communicate with each other, you can specify password encryption in the SYSIBM.LUNAMES table.

Table 49. Specifying password encryption in SYSIBM.LUNAMES

LUNAME	USERNAMES	ENCRYPTPSWDS
LUXXX	O	Y

The partner DB2 must also specify password encryption in its SYSIBM.LUNAMES table. Both partners must register each ID and its password with RACF. Then, for every request to LUXXX, your DB2 calls RACF to supply an encrypted password to accompany the ID. With password encryption, you do not use the PASSWORD column of SYSIBM.USERNAMES, so the security of that table becomes less critical.

## Sending RACF PassTickets

To send RACF PassTickets with your remote requests to a particular remote system, you can specify 'R' in the SECURITY\_OUT column of the SYSIBM.IPNAMES or SYSIBM.LUNAMES table for that system.

To set up RACF to generate PassTickets:

1. Activate the RACF PTKTDATA class by issuing the following RACF commands:  

```
SETROPTS CLASSACT(PTKTDATA)  
SETROPTS RACLIST(PTKTDATA)
```
2. Define a RACF profiles for each remote system by entering the system name as it appears in the LINKNAME column in the SYSIBM.LOCATIONS table.  
For example, issue the following command defines a profile for a remote system, DB2A, in the RACF PTKTDATA class:  

```
RDEFINE PTKTDATA DB2A SSIGNON(KEYMASKED(E001193519561977))
```
3. Refresh the RACF PTKTDATA definition with the new profiles by issuing the following command:  

```
SETROPTS RACLIST(PTKTDATA) REFRESH
```

## Sending encrypted passwords or password phrases from DB2 for z/OS clients

As a requester, a DB2 for z/OS client can send connection requests that use 256-bit Advanced Encryption Standard (AES) or 56-bit Data Encryption Standards (DES) encryption security through a TCP/IP network to remote servers.

If the DB2 for z/OS client supports DRDA Security Manager (SECMGR) 9 (or higher) and if z/OS ICSF is configured and started, it can send AES requests to a server. After the first successful connection, it can determine whether or not a remote server supports AES encryption security. If the remote server also supports DRDA SECMGR 9 (or higher), it accepts AES requests and encrypts the user IDs and passwords or password phrases that the client sends in AES.

As a client, DB2 for z/OS only supports IPNAMES.SECURITY\_OUT option 'P' ("password") for AES encryption and decryption. It does not support IPNAMES.SECURITY\_OUT option 'D' ("user ID and security-sensitive data encryption") or 'E' ("user ID, password, and security-sensitive data encryption"). These outbound security options remain encrypted in DES.

### Related concepts

 Encrypted password, user ID, or user ID and password security under the IBM Data Server Driver for JDBC and SQLJ (Application Programming for Java)

## Sending encrypted passwords from workstation clients

As a server, DB2 for z/OS can accept requests from remote workstation clients that use 256-bit Advanced Encryption Standard (AES) or 56-bit Data Encryption Standards (DES) encryption security over a TCP/IP network connection.

Depending on the DRDA level, a remote client can use AES or DES encryption algorithm for sending passwords, user IDs and passwords, or other security-sensitive data to a DB2 for z/OS server. If both the DB2 for z/OS server and the remote client support DRDA Security Manager (SECMGR) 9 or higher and even if the client does not explicitly request for AES, AES becomes the default encryption algorithm for user IDs and passwords, and DES remains the default encryption algorithm for security-sensitive data. In other words, if the client

| explicitly requests for AES encryption, only user IDs, passwords, or both are  
| encrypted in AES, and any data in the request is still encrypted in DES. Any  
| persistent attempt to encrypt the data in AES will cause the client itself to reject the  
| connection request.

| To enable the DB2 for z/OS AES server support, you must install and configure  
| z/OS Integrated Cryptographic Services Facility (ICSF). During DB2 startup,  
| DSNXINIT invokes the MVS LOAD macro service to load various ICSF services,  
| including the ICSF CSNESYE and CSNESYD modules that DB2 calls for processing  
| AES encryption and decryption requests. If ICSF is not installed or if ICSF services  
| are not available, DB2 will not be able to provide AES support. Instead, it will use  
| DES for processing remote requests if the client does not explicitly request for AES  
| encryption.

To use the DES encryption, you can enable DB2 Connect to send encrypted passwords by setting database connection services (DCS) authentication to DCS\_ENCRYPT in the DCS directory entry. When a client application issues an SQL CONNECT, the client negotiates this support with the database server. If supported, a shared private key is generated by the client and server using the Diffie-Hellman public key technology and the password is encrypted using 56-bit DES with the shared private key. The encrypted password is non-replayable, and the shared private key is generated on every connection. If the server does not support password encryption, the application receives SQLCODE -30073 (DRDA security manager level 6 is not supported).

#### **Related concepts**

 Encrypted password, user ID, or user ID and password security under the IBM Data Server Driver for JDBC and SQLJ (Application Programming for Java)



---

## Chapter 4. Managing access through row permissions and column masks

*Row and column access control* enables you to manage access to a table at the level of a row, a column, or both. You can implement row access control through row permissions and column access control through column masks.

---

### Row and column access control

Row and column access control is a DB2 security solution that uses SQL to control access to a table at the level of a row, a column, or both.

Traditionally, access control at the row and column level is implemented through views. Using views as an access control method works well only when access rules, restrictions, and conditions are monolithic and simple. It however becomes ineffective when view definitions become too complex because of the complexity and granularity of privacy and security policies. It also becomes costly when a large number of views must be manually updated and maintained. In addition, the ability to update views proves to be challenging. As privacy and security policies evolve, required updates to views may negatively affect the security logic particularly when database applications reference the views directly by name. DB2 row and column access control helps resolve all these problems.

Implemented through SQL and managed by the DB2 security administrator, row and column access control allows you to manage access to a table with filtering and data masking. Unlike multilevel security, row and column access control is integrated into a database system, and all applications and tools that use SQL to access the database are automatically subject to the same control. This effectively eliminates the need to filter security-sensitive data at the application level and ensures that the data is protected when the applications and tools use SQL to access it.

Row and column access control is based on a security policy that specifies the rules and conditions under which a user, group, or role can access rows, columns, or both of a base table. The access control is not needed at the view level because the view automatically receives row and column access control that is activated on the underlying base table. The row and column access control rules do not affect how a read-only view is determined. All users access the same base table (as opposed to alternative views of a table), but access restrictions are based on individual user permissions and masks that are specified by a policy associated with the table.

An authorization ID or role with the SECADM authority can manage row and column access control. The SECADM authority can activate or deactivate row and column access control for a table, grant or revoke the CREATE\_SECURE\_OBJECT system privilege, and create, alter, or drop row permissions and column masks. The SYSADM authority can perform the same tasks if the SEPARATE SECURITY system parameter on panel DSNTIPP1 is set to NO during installation.

Row and column access control can be activated for a table before or after row permissions or column masks are created for the table. If row permissions or column masks already exist, activating row and column access control simply makes the permissions or masks become effective. If row permissions or column



masks do not yet exist, activating row access control for a table means that DB2 will generate a default row permission that prevents any access to the table by SQL, and activating column access control means to wait for the column masks to be created.

When a table is activated for row or column access control, all users, including the table owner and the SECADM, SYSADM, or DBADM authorities, are subject to the same security rules and restrictions. This ensures that access to security-sensitive data is truly on a need basis and prevents system and database administrators from unnecessarily accessing it. Since security policies or rules are expressed and enforced through SQL, row and column access control is inherently flexible.

Row access control and multilevel security are mutually exclusive. If a table is activated for row access control, it cannot be altered to include a security label column; if a table has a security label column, it cannot be activated for row access control. Column access control, on the other hand, is not affected by multilevel security. If a table is activated for column level access control, it can be altered to include a security label column, and vice versa.

#### **Related concepts**

“Row permission”

“Column mask” on page 191

#### **Related reference**

“Explicit system privileges” on page 27

“SECADM” on page 41

---

## **Row permission**

A *row permission* is a database object that describes a specific row access control rule for a table. In the form of an SQL search condition, the rule specifies the conditions under which a user, group, or role can access the rows of data in the table.

Stored in the system catalog, row permissions can be created on all base tables except materialized query tables, and they are maintained on an individual basis. The definition of each row permission may reference the user, group, or role in the search condition.

If multiple row permissions are defined for a table and when row access control is activated, the search condition in each row permission is connected by the logical OR operator to form the row access control search condition. This row access control search condition is applied whenever the table is accessed. It acts as a filter to the table before any other user-specified operations, such as predicates and ordering, are processed. It also acts like the WITH CHECK OPTION clause of a view to ensure that a row to be inserted or updated conforms to the definitions of the row permissions in an INSERT, UPDATE, or MERGE statement.

Only an authorization ID or role with the SECADM or SYSADM authority can manage row permissions. If the SEPARATE\_SECURITY system parameter on panel DSNTPP1 is set to YES during installation or migration, you must have the SECADM authority to create, alter, or drop row permissions. If SEPARATE\_SECURITY is set to NO, you must have the SECADM or SYSADM authority.

### Related concepts

“Column mask”

### Related tasks

“Creating row permissions” on page 196

“Creating column masks” on page 198

### Related reference

“Rules of row and column access control” on page 192

“Explicit system privileges” on page 27

“SECADM” on page 41

---

## Column mask

A *column mask* is a database object that describes a specific column access control rule for a column. In the form of an SQL CASE expression, the rule specifies the condition under which a user, group, or role can receive the masked values that are returned for a column.

Stored in the system catalog, column masks can be created on all base tables except materialized query tables and maintained on an individual basis. The definition of each column mask may reference the user, group, or role in the search conditions in the CASE WHEN clause.

While multiple columns in a table may have column masks, only one column mask can be created for a single column. When column access control is activated for the table, the CASE expression in the column mask definition is applied to an output column to determine the masked values that are returned to an application. The application of column masks affects the final output only; it does not impact the operations, such as predicates and ordering, in an SQL statement. In addition, the application of column masks must not change the values in a row that is being inserted or updated in an INSERT, UPDATE, or MERGE statement.

Only an authorization ID or role with the SECADM or SYSADM authority can manage column masks. If the SEPARATE\_SECURITY system parameter on panel DSNTIPP1 is set to YES during installation or migration, you must have the SECADM authority to create, alter, or drop column masks. If SEPARATE\_SECURITY is set to NO, you must have the SECADM or SYSADM authority.

### Related concepts

“Row permission” on page 190

### Related tasks

“Creating row permissions” on page 196

“Creating column masks” on page 198

### Related reference

“Rules of row and column access control”

“Explicit system privileges” on page 27

“SECADM” on page 41

## Rules of row and column access control

The rules of row and column access control apply to both read and write operations on a table. The conditions that are specified in row permissions and column masks apply to both data retrieval operations and data change operations.

### GUPI

The following table shows an example of how row and column access control rules are applied depending on the types of data operations. Assume that tables T1 and T2 are activated for row and column access control and that both tables include columns C1 and C2.

Table 50. Rules and access types for row and column access control

SQL statement	Row permission	Column mask (defined for column C1)
SELECT SUBSTR( C1,8,4) FROM T1;	<ul style="list-style-type: none"><li>• If user-defined row permissions exist for the table, only the rows that satisfy the permissions are returned.</li><li>• If no user-defined row permissions exist for the table, the default row permission is applied and no row is returned.</li></ul>	<ul style="list-style-type: none"><li>• The column mask is applied to column C1 that is referenced in the select list of the outermost SELECT clause. It does not interfere with the operations of other clauses within the statement, such as the WHERE, GROUP BY, HAVING, SELECT DISTINCT, or ORDER BY clauses. Some column mask restrictions may apply to the other clauses within the statement.</li><li>• The masked value that is determined by the evaluation of the CASE expression in the column mask is returned in place of the column value in the output row. If column C1 is embedded in an expression, the column mask is applied to the input column before the evaluation of the expression takes place.</li></ul>

Table 50. Rules and access types for row and column access control (continued)

SQL statement	Row permission	Column mask (defined for column C1)
INSERT INTO T1(C1, C2) VALUES('A', 'B');	<p>For each row to be inserted:</p> <ul style="list-style-type: none"> <li>• If a user-defined row permission exists, the row can be inserted only when that row can be subsequently retrieved by the authorization ID of the INSERT statement. If the row cannot be inserted, the INSERT statement returns an error.</li> <li>• If no user-defined row permissions exist for the table only the default row permission is applied and no row is inserted. The INSERT statement returns an error.</li> </ul> <p>The ENFORCED FOR ALL ACCESS clause ensures that users cannot insert data that they cannot read.</p>	

Table 50. Rules and access types for row and column access control (continued)

SQL statement	Row permission	Column mask (defined for column C1)
INSERT INTO T1(C1) SELECT SUBSTR( T2.C1, 8, 4) FROM T2 WHERE T2.C2 > 10;		<ul style="list-style-type: none"> <li>When the columns are used to derive the new values for an INSERT statement, the original column values, not the masked values, are used. If the columns have column masks, those column masks are applied to ensure the evaluation of the access control rules at run time masks the column to itself, not to a constant or an expression. This ensures that the masked values are the same as the original column values. If a column mask does not mask the column to itself, the new row is not inserted and an error is returned at run time.</li> </ul> <p>For example, column T2.C1 is used to derive the value of a new row for INSERT. The column value of T2.C1, not the masked value, is used to derive the new value. Because column T2.C1 has a column mask, the column mask is applied to ensure the evaluation of the access control rule in the column mask masks column T2.C1 to itself, not to a constant or an expression. This ensures the masked value is the same as the original column value. If the column mask of T2.C1 does not mask column T2.C1 to itself, the new value cannot be used and an error is returned at run time.</p> <ul style="list-style-type: none"> <li>The column mask rules that apply to the new value for INSERT are the same as those for SELECT.</li> </ul>

Table 50. Rules and access types for row and column access control (continued)

SQL statement	Row permission	Column mask (defined for column C1)
UPDATE T1 SET C2 = (SELECT SUBSTR(T2.C1, 8, 4) FROM T2 WHERE T2.C2 > 10);	<p>The following rules are applied in the order as shown:</p> <ol style="list-style-type: none"> <li>Identify candidate rows for updates: <ul style="list-style-type: none"> <li>If a user-defined row permission exists, only the rows of the table that satisfy the row permission can be the candidate rows for UPDATE.</li> <li>If no user-defined row permissions exist for the table, only the default row permission is applied and no rows are updated.</li> </ul> </li> <li>If there are rows to be updated, for each row to be updated: <ul style="list-style-type: none"> <li>When the columns are used to derive the new values for an UPDATE statement, the original column values, not the masked values, are used. If the columns have column masks, those column masks are applied to ensure that the evaluation of the access control rules at run time masks the column to itself, not to a constant or an expression. This ensures the masked values are the same as the original column values. If a column mask does not mask the column to itself, the new value cannot be used for the update and an error is returned at run time.</li> <li>For example, column T2.C1 is used to derive the new value for the update. The column value of T2.C1, not the masked value, is used to derive the new value. Because column T2.C1 has a column mask, the column mask is applied to ensure that the evaluation of the access control rule in the column mask masks column T2.C1 to itself, not to a constant or an expression. This ensures that the masked value is the same as the original column value. If the column mask of T2.C1 does not mask column T2.C1 to itself, the new value cannot be used for the update and an error is returned at run time.</li> <li>The column mask rules that apply to the new value for UPDATE are the same as those for SELECT.</li> </ul> </li> <li>If there are rows to be updated, for each row to be updated: <ul style="list-style-type: none"> <li>If a user-defined row permission exists, the row can be updated only when that row can be subsequently retrieved by the authorization ID of the UPDATE statement. If the row cannot be updated, the UPDATE statement returns an error. The ENFORCED FOR ALL ACCESS clause ensures that users cannot update data that they cannot read.</li> <li>The column mask is not applicable in this retrieval.</li> </ul> </li> </ol>	
MERGE	The row and column access control rules for the UPDATE and INSERT operations in the MERGE statement are the same as those for the UPDATE and INSERT statements.	
DELETE	<ul style="list-style-type: none"> <li>If a user-defined row permission exists for the table, only the rows that satisfy the permission are the candidate rows for an DELETE statement.</li> <li>If no user-defined row permissions exist for the table, the default row permission is applied and no row can be deleted.</li> </ul>	

## GUIP

### Related concepts

“Row permission” on page 190

“Column mask” on page 191

### Related tasks

“Creating row permissions”

“Creating column masks” on page 198

“Using INSERT on tables with row access control” on page 202

“Creating triggers for tables with row and column access control” on page 203

“Modifying column masks to reference UDFs” on page 200

## Creating row permissions

A *row permission* specifies the conditions under which users can access a row. With the SECADM authority, you can use the CREATE PERMISSION statement to create a row permission.

If the SEPARATE\_SECURITY system parameter on panel DSNTIPP1 is set to YES during installation or migration, you must have the SECADM authority to create a row permission. If SEPARATE\_SECURITY is set to NO, you must have the SECADM or SYSADM authority.

Suppose that you are a data security administrator (SECADM) for a national health organization (NetHMO) and responsible for safeguarding sensitive patient information. You want to create a data privacy and security policy and implement it through row permissions on tables that are enabled with row access control. The permission definitions prescribe the conditions under which patients, physicians, pharmacists, or account administrators can only receive certain rows based on their roles or account authentication IDs.

To create a row permission:

1. Issue the CREATE TABLE statement to create table HOSPITAL.PATIENT.

The HOSPITAL.PATIENT table contains columns for recording a patient's social security number (SSN), account authorization ID (USERID), name (NAME), address (ADDRESS), pharmacy (PHARMACY), account balance

(ACCT\_BALANCE), and doctor (PCP\_ID), as shown below: **GUIP**

```
CREATE TABLE HOSPITAL.PATIENT (  
    SSN CHAR(11),  
    USERID VARCHAR(18),  
    NAME VARCHAR(128),  
    ADDRESS VARCHAR(128),  
    PHARMACY VARCHAR(5000),  
    ACCT_BALANCE DECIMAL(12,2) WITH DEFAULT,  
    PCP_ID VARCHAR(18));
```

## GUIP

2. Issue the CREATE ROLE statements to create the following roles and determine the rules for each role to access the HOSPITAL.PATIENT table

The row access control rules specify the specific types of information that users in a specific role can access and the conditions under which the role can access the information. **GUIP**



```

CREATE ROLE PCP;
CREATE ROLE DRUG_RESEARCH;
CREATE ROLE ACCOUNTING;
CREATE ROLE MEMBERSHIP;
CREATE ROLE PATIENT;

```

#### GUIP

3. Issue the CREATE PERMISSION statement to create row permissions that allow each role to access data in specific rows.

You can use the built-in function VERIFY\_TRUSTED\_CONTEXT\_ROLE\_FOR\_USER to determine whether the user identified in special register SESSION\_USER is associated with a particular ROLE that is specified as the input argument to the function.

In the following example, Role PATIENT is allowed to access his or her own row. Role PCP is allowed to access his or her patients' rows. Roles MEMBERSHIP, ACCOUNTING, and DRUG\_RESEARCH are allowed to access

all rows. **GUIP**

```

CREATE PERMISSION NETHMO.ROW_ACCESS ON HOSPITAL.PATIENT
FOR ROWS WHERE (VERIFY_TRUSTED_CONTEXT_ROLE_FOR_USER(SESSION_USER,
    'PATIENT') = 1 AND
    HOSPITAL.PATIENT.USERID = SESSION_USER) OR
    (VERIFY_TRUSTED_CONTEXT_ROLE_FOR_USER(SESSION_USER,
    'PCP') = 1 AND
    HOSPITAL.PATIENT.PCP_ID = SESSION_USER) OR
    (VERIFY_TRUSTED_CONTEXT_ROLE_FOR_USER(SESSION_USER,
    'MEMBERSHIP') = 1 OR
    VERIFY_TRUSTED_CONTEXT_ROLE_FOR_USER(SESSION_USER,
    'ACCOUNTING') = 1 OR
    VERIFY_TRUSTED_CONTEXT_ROLE_FOR_USER(SESSION_USER,
    'DRUG_RESEARCH') = 1)
ENFORCED FOR ALL ACCESS
ENABLE;

COMMIT;

```

#### GUIP

The definitions of the new row permissions are stored in the new catalog table SYSIBM.SYSCONTROLS.

4. Issue the ALTER TABLE statement with the ACTIVATE ROW ACCESS CONTROL clause to activate row access control for table HOSPITAL.PATIENT.

#### GUIP

```

ALTER TABLE HOSPITAL.PATIENT ACTIVATE ROW ACCESS CONTROL;

COMMIT;

```

#### GUIP

The ALTER TABLE serialization process takes place and invalidates all packages and dynamic cached statements that reference table HOSPITAL.PATIENT. The value 'R' in the new column SYSTABLES.CONTROL indicates that the table is activated for row access control.

DB2 also implicitly creates a default row permission which restricts all access from HOSPITAL.PATIENT. The default row permission definition is stored in the new catalog table SYSIBM.SYSCONTROLS.

Whenever table HOSPITAL.PATIENT is referenced in a data manipulation statement, all row permissions that have been created for it, including the

default row permission, are implicitly applied by DB2 to control the rows in the table that are accessible. A row access control search condition is derived from the logical OR operators that connect the search condition in each row permission. This search condition acts as a filter to HOSPITAL.PATIENT before any user-specified operations, such as predicates, grouping, ordering, are processed.

If necessary, SECADM can deactivate row access control from table HOSPITAL.PATIENT by simply issuing the following ALTER TABLE statement:

**GUI**

```
ALTER TABLE HOSPITAL.PATIENT DEACTIVATE ROW ACCESS CONTROL;  
  
COMMIT;
```

**GUI**

DB2 invalidates all packages and dynamic cached statements that reference HOSPITAL.PATIENT. DB2 reflects the removal of row access control by setting SYSTABLES.CONTROL to blank. This also means that table HOSPITAL.PATIENT does not have any access control and that users can retrieve data from all its rows.

#### **Related concepts**

“Row permission” on page 190

“Column mask” on page 191

#### **Related tasks**

“Creating column masks”

“Using INSERT on tables with row access control” on page 202

“Creating triggers for tables with row and column access control” on page 203

“Modifying column masks to reference UDFs” on page 200

#### **Related reference**

“Rules of row and column access control” on page 192

“Explicit system privileges” on page 27

“SECADM” on page 41

---

## **Creating column masks**

A *column mask* specifies the rules for users to receive the masked values that are returned for the column. With the SECADM authority, you can use the CREATE MASK statement to create a column mask.

**GUI**

If the SEPARATE\_SECURITY system parameter on panel DSNTIPP1 is set to YES during installation or migration, you must have the SECADM authority to create a column mask. If SEPARATE\_SECURITY is set to NO, you must have the SECADM or SYSADM authority.

Suppose that you are a data security administrator (SECADM) for a national health organization (NetHMO) and responsible for safeguarding sensitive patient information. You want to create a data privacy and security policy and implement it through column masks on tables that are enabled with column access control. The mask definitions prescribe the conditions under which patients, physicians, pharmacists, or account administrators can only receive certain masked values from the column based on their roles or account authentication IDs.

To create a column mask:

1. Issue the CREATE TABLE statement to create table HOSPITAL.PATIENT

The HOSPITAL.PATIENT table contains columns for recording a patient's social security number (SSN), account authorization ID (USERID), name (NAME), address (ADDRESS), pharmacy (PHARMACY), account balance (ACCT\_BALANCE), and doctor (PCP\_ID), as shown below:

```
CREATE TABLE HOSPITAL.PATIENT (  
    SSN CHAR(11),  
    USERID VARCHAR(18),  
    NAME VARCHAR(128),  
    ADDRESS VARCHAR(128),  
    PHARMACY VARCHAR(5000),  
    ACCT_BALANCE DECIMAL(12,2) WITH DEFAULT,  
    PCP_ID VARCHAR(18));
```

2. Issue the CREATE ROLE statements to create the following roles that access the HOSPITAL.PATIENT table.

```
CREATE ROLE PCP;  
CREATE ROLE DRUG_RESEARCH;  
CREATE ROLE ACCOUNTING;  
CREATE ROLE MEMBERSHIP;  
CREATE ROLE PATIENT;
```

3. Issue the CREATE MASK statement to create column masks that allow each role to receive certain masked values from specific columns.

You can use the built-in function VERIFY\_TRUSTED\_CONTEXT\_ROLE\_FOR\_USER to determine whether the user identified in special register SESSION\_USER is associated with a particular ROLE that is specified as the input argument to the function.

The following example shows how column mask SSN\_MASK is created. Roles PATIENT and ACCOUNTING are allowed to receive column values from column SSN. Other roles that access the column will receive masked values.

```
CREATE MASK NETHMO.SSN_MASK ON HOSPITAL.PATIENT FOR  
    COLUMN SSN RETURN  
        CASE WHEN VERIFY_TRUSTED_CONTEXT_ROLE_FOR_USER(SESSION_USER,  
            'PATIENT') = 1 OR  
            VERIFY_TRUSTED_CONTEXT_ROLE_FOR_USER(SESSION_USER,  
            'ACCOUNTING') = 1  
        THEN SSN  
        ELSE CHAR('XX-XX-') || SUBSTR(SSN,8,4)  
        END  
    ENABLE;
```

```
COMMIT;
```

You can issue the CREATE MASK statements to create column masks USERID\_MASK, NAME\_MASK, and ADDRESS\_MASK. The definitions of all these column masks are stored in the new catalog table SYSIBM.SYSCONTROLS.

4. Use the ALTER TABLE statement with the ACTIVATE COLUMN ACCESS CONTROL clause to activate column access control for table HOSPITAL.PATIENT

```
ALTER TABLE HOSPITAL.PATIENT ACTIVATE COLUMN ACCESS CONTROL;
```

```
COMMIT;
```

The ALTER TABLE serialization process takes place and invalidates all packages and dynamic cached statements that reference table HOSPITAL.PATIENT. The value 'C' in the new column SYSTABLES.CONTROL indicates that the table is activated for column access control.

Whenever column SSN of table HOSPITAL.PATIENT is referenced in the outermost SELECT clause of a data manipulation statement, column mask SSN\_MASK is implicitly applied by DB2 to control the masked values that are returned for it.

If necessary, SECADM can deactivate column access control from table HOSPITAL.PATIENT by simply issuing the following ALTER TABLE statement:

```
ALTER TABLE HOSPITAL.PATIENT DEACTIVATE COLUMN ACCESS CONTROL;
```

```
COMMIT;
```

DB2 invalidates all packages and dynamic cached statements that reference HOSPITAL.PATIENT. DB2 reflects the removal of column access control by

setting SYSTABLES.CONTROL to blank. 

#### **Related concepts**

“Row permission” on page 190

“Column mask” on page 191

#### **Related tasks**

“Creating row permissions” on page 196

“Using INSERT on tables with row access control” on page 202

“Creating triggers for tables with row and column access control” on page 203

“Modifying column masks to reference UDFs”

#### **Related reference**

“Rules of row and column access control” on page 192


“Explicit system privileges” on page 27

“SECADM” on page 41

---

## **Modifying column masks to reference UDFs**

With the SECADM authority, you can modify column masks on tables that are activated for column access control.

 If the SEPARATE\_SECURITY system parameter on panel DSNTIPP1 is set to YES during installation or migration, you must have the SECADM authority to modify a column mask. If SEPARATE\_SECURITY is set to NO, you must have the SECADM or SYSADM authority.

Suppose that table HOSPITAL.PATIENT contains columns to record a patient's social security number (SSN), account authorization ID (USERID), name (NAME), address (ADDRESS), pharmacy (PHARMCY), account balance (ACCT\_BALANCE), and doctor (PCD\_ID). The table is activated for column access control.

Also, suppose that database developer Paul has created a new powerful accounting application NetHMLAccountingUDF (an external scalar user-defined function). You want to modify column mask ACCT\_BALANCE\_MASK for column HOSPITAL.PATIENT.ACCT\_BALANCE to include NetHMLAccountingUDF.

To modify column mask ACCT\_BALANCE\_MASK to include NetHMLAccountingUDF:

1. Make sure that all the operations in the new UDF are secure.

Only secure UDFs can be invoked in a column mask. The SECURED attribute is required if the user-defined function is referenced in the definition of a row permission or column mask. This is because the user-defined function may

access sensitive data. The SECURED attribute is also required for a user-defined function that is invoked in an SQL statement when the function arguments reference columns that are activated with column access control.

Make sure that all the operations inside the new application NetHMLAccountingUDF are actually secure. Then, you can issue the following GRANT CREATE\_SECURE\_OBJECT statement to allow userid PAUL the privilege for creating a secure UDF:

```
GRANT CREATE_SECURE_OBJECT TO PAUL;
```

```
COMMIT;
```

DB2 records the grant in SYSUSERAUTH: GRANTOR = GRANTORID, GRANTEE = PAUL, AUTHHOWGOT = E, and CREATESECUREAUTH = Y. This means that authid GRANTORID has used the SECADM authority (AUTHHOWGOT = E) to grant userid PAUL the CREATE\_SECURE\_OBJECT privilege.

With the CREATE\_SECURE\_OBJECT privilege, Paul issues the following ALTER FUNCTION statement to secure NetHMLAccountingUDF:

```
ALTER FUNCTION NETHMOACCOUNTINGUDF(ACCT_BALANCE) SECURED;
```

```
COMMIT;
```

DB2 sets the new column in catalog SYSROUTINES.SECURE to Y and invalidates all packages and dynamic cached statements that reference NetHMOAccountingUDF(ACCT\_BALANCE).

2. After the UDF has been altered to be secure, revoke the CREATE\_SECURE\_OBJECT privilege from userid PAUL by issuing the following REVOKE CREATE\_SECURE\_OBJECT statement:

```
REVOKE CREATE_SECURE_OBJECT FROM PAUL;
```

```
COMMIT;
```

DB2 completes the privilege removal by deleting the row from SYSUSERAUTH with GRANTOR = GRANTORID, GRANTEE = PAUL, AUTHHOWGOT = E, and CREATESECUREAUTH = Y.

3. Drop the existing column mask ACCT\_BALANCE\_MASK for column ACCT\_BALANCE

You can issue the DROP MASK statement to remove the existing column mask, but do not follow it with the COMMIT statement. This will prevent any ongoing transactions from accessing table HOSPITAL.PATIENT before you can put a new column mask in place.

```
DROP MASK ACCT_BALANCE_MASK;
```

DB2 invalidates all packages and dynamic cached statements that reference table HOSPITAL.PATIENT. It also deletes the row for ACCT\_BALANCE\_MASK in the catalog table SYSIBM.SYSCONTROLS. Since there isn't a COMMIT statement immediately after the DROP MASK statement, DB2 keeps possessing the lock on HOSPITAL.PATIENT and doesn't commit the work it has done for the DROP MASK statement. Any transactions that try to access HOSPITAL.PATIENT may be timed out.

4. Create a new column mask ACCT\_BALANCE\_MASK for column ACCT\_BALANCE

You can issue the CREATE MASK statement to create a new column mask and follow it immediately with the COMMIT statement. This will enable DB2 to commit all the work it has done so far for HOSPITAL.PATIENT and allow other transactions to access HOSPITAL.PATIENT.


```

CREATE MASK NETHMO.ACCT_BALANCE_MASK ON HOSPITAL.PATIENT FOR
  COLUMN ACCT_BALANCE RETURN
  CASE WHEN VERIFY_TRUSTED_CONTEXT_ROLE_FOR_USER
    (SESSION_USER, 'ACCOUNTING') = 1
    THEN NETHMOACCOUNTINGUDF(ACCT_BALANCE)
    ELSE 0.00
  END
ENABLE;

```

```
COMMIT;
```

DB2 invalidates all packages and dynamic cached statements that reference table HOSPITAL.PATIENT and inserts the new ACCT\_BALANCE\_MASK definition into the catalog table SYSIBM.SYSCONTROLS. It also records the dependency on NetHMOAccountingUDF in SYSIBM.SYSDEPENDENCIES for ACCT\_BALANCE\_MASK.

The COMMIT statement immediately after the CREATE MASK statement ensures that DB2 commits all the work it has done so far on HOSPITAL.PATIENT and releases the lock from the table. This allows other transactions to access the same table without being timed out. 


---

## Using INSERT on tables with row access control

You can use the INSERT statement on tables that are activated for row access control.

Suppose that you are responsible for managing patient memberships and you are associated with role MEMBERSHIP that is already created for table HOSPITAL.PATIENT. Table HOSPITAL.PATIENT is also activated for row access control, and Role MEMBERSHIP is allowed to access, create, and retrieve rows in the table.

Supposed that table HOSPITAL.PATIENT contains columns to record a patient's social security number (SSN), account authorization ID (USERID), name (NAME), address (ADDRESS), pharmacy (PHARMCY), account balance (ACCT\_BALANCE), and doctor (PCD\_ID). You want to add a new row for a new patient Bob.

 To add a row to a table that is enforced with row access control:

1. Ensure that role MEMBERSHIP is allowed to access, insert, and update rows when row permission rules are set for table HOSPITAL.PATIENT by the SECADM authority.

```

CREATE PERMISSION NETHMO.ROW_ACCESS ON HOSPITAL.PATIENT
  FOR ROWS WHERE (VERIFY_TRUSTED_CONTEXT_ROLE_FOR_USER(SESSION_USER,
    'MEMBERSHIP') = 1 )
  ENFORCED FOR ALL ACCESS
ENABLE;

```

```
COMMIT;
```

2. Issue the INSERT statement to insert a new row for patient Bob:

```

INSERT INTO HOSPITAL.PATIENT(SSN, USERID, NAME, ADDRESS)
VALUES('123-45-6789', 'BobXYZ100', 'Bob', '123 Some St. ');

```

```
COMMIT;
```

3. Verify that Bob was successfully added by issuing the following SELECT statement:

```
SELECT * FROM HOSPITAL.PATIENT WHERE SSN = '123-45-6789';
```

```
COMMIT;
```

The following result is returned and shows that a new row for Bob was successfully added to the table.

SSN	USERID	NAME	ADDRESS	PHARMACY	ACCT_BALANCE	PCP_ID
123-45-6789	BobXYZ100	Bob	123 Some St.	?	0.00	?

```
DSNT400I  SQLCODE = 000, SUCCESSFUL EXECUTION
          SUCCESSFUL RETRIEVAL OF 1 ROW(S)
```

**GUI**

## Creating triggers for tables with row and column access control

With the required authority and privileges, you can create triggers for tables that are activated for row and access control.

If the `SEPARATE_SECURITY` system parameter on panel `DSNTIPP1` is set to `YES` during installation or migration, you must have the `SECADM` authority to create triggers for tables that are activated with row and column access control. If `SEPARATE_SECURITY` is set to `NO`, you must have the `SECADM` or `SYSADM` authority.

Suppose that table `HOSPITAL.PATIENT` is activated for row and column access control. The table contains columns to record a patient's social security number (SSN), account authorization ID (USERID), name (NAME), address (ADDRESS), pharmacy (PHARMCY), account balance (ACCT\_BALANCE), and doctor (PCD\_ID). Paul, a database developer, needs to create a new `AFTER UPDATE` trigger for `HOSPITAL.PATIENT` to monitor the history of the `ACCT_BALANCE` column.

To create a trigger for a table that is enforced with row and access control:

1. Make sure that all operations on the transition variables and transition tables inside the new trigger body are secure.

Only secure triggers can be defined on tables that are already enforced with row and column access control. The `SECURED` attribute is required for a trigger when the associated table is row or column access control enforced or the associated view whose underlying table is enforced with row or column access control. If a trigger exists but is not secure, row or column access control cannot be activated for the associated table.

Make sure that all operations on the transition variables and transition tables inside the new trigger body are actually secure. Then, you can issue the following `GRANT CREATE_SECURE_OBJECT` statement to allow userid `PAUL`

the privilege for creating secure triggers for table `HOSPITAL.PATIENT`:

```
GRANT CREATE_SECURE_OBJECT TO PAUL;
```

```
COMMIT;
```

**GUI**

DB2 records the grant in `SYSUSERAUTH`: `GRANTOR = GRANTORID`, `GRANTEE = PAUL`, `AUTHHOWGOT = E`, and `CREATESECUREAUTH = Y`.



This means that authid GRANTORID has used the SECADM authority (AUTHHOWGOT = E) to grant userid PAUL the CREATE\_SECURE\_OBJECT privilege.

2. Create a new trigger for table HOSPITAL.PATIENT

With the CREATE\_SECURE\_OBJECT privilege, Paul can create a secure NETHMO\_ACCT\_BALANCE\_TRIGGER by issuing the following CREATE

TRIGGER statement: **GUIP**

```
CREATE TRIGGER NETHMO_ACCT_BALANCE_TRIGGER NO CASCADE
  AFTER UPDATE OF ACCT_BALANCE ON HOSPITAL.PATIENT SECURED
  REFERENCING OLD AS O NEW AS N
  FOR EACH ROW MODE DB2SQL
  BEGIN ATOMIC
    INSERT INTO ACCT_HISTORY
      (SSN, BEFORE_BALANCE, AFTER_BALANCE, WHEN, BY_WHO)
    VALUES(O.SSN, O.ACCT_BALANCE, N.ACCT_BALANCE,
      CURRENT_TIMESTAMP, SESSION_USER);
  END!

COMMIT!
```

**GUIP**

DB2 inserts a new row into SYSIBM.SYSTRIGGERS with SYSTRIGGERS.SECURE = 'Y'. DB2 completes other catalog table updates for the trigger creation.

3. After trigger NETHMO\_ACCT\_BALANCE\_TRIGGER is created, revoke the CREATE\_SECURE\_OBJECT privilege from userid PAUL by issuing the following REVOKE CREATE\_SECURE\_OBJECT statement:

**GUIP**

```
REVOKE CREATE_SECURE_OBJECT FROM PAUL;

COMMIT;
```

**GUIP**

DB2 completes the privilege removal by deleting the row with GRANTOR = GRANTORID, GRANTEE = PAUL, AUTHHOWGOT = E, and CREATESECUREAUTH = Y from SYSUSERAUTH. This means that authid GRANTORID has used the SECADM authority (AUTHHOWGOT = E) to revoke the CREATE\_SECURE\_OBJECT privilege from userid PAUL.

---

## Chapter 5. Managing access through trusted contexts

You can use trusted contexts to manage access to your DB2 subsystems, which helps you improve data security.

You can use trusted connections within a trusted context. When you do this, you can reuse the authorization and switch users of the connection without the database server needing to authenticate the IDs.

---

### Trusted contexts

A *trusted context* is an independent database entity that you can define based on a system authorization ID and connection trust attributes.

The trust attributes specify a set of characteristics about a specific connection. These attributes include the IP address, domain name, or SERVAUTH security zone name of a remote client and the job or task name of a local client.

A trusted context allows for the definition of a unique set of interactions between DB2 and the external entity, including the following abilities:

- The ability for the external entity to use an established database connection with a different user without the need to authenticate that user at the DB2 server. This ability eliminates the need to manage end-user passwords by the external entity. Also, a database administrator can assume the identity of other users and perform actions on their behalf.
- The ability for a DB2 authorization ID to acquire one or more privileges within a trusted context that are not available to it outside of that trusted context. This is accomplished by associating a role with the trusted context.

The following client applications provide support for the trusted context:

- The DB2 Universal Java Driver introduces new APIs for establishing trusted connections and switching users of a trusted connection.
- The DB2 CLI/ODBC Driver introduces new keywords for connecting APIs to establish trusted connections and switch users of a trusted connection.
- The Websphere Application Server 6.0 exploits the trusted context support through its "propagate client identity" property.

### Related concepts

“Trusted connections”

“Roles in a trusted context” on page 23

### Related tasks

“Defining trusted contexts”

“Creating local trusted connections” on page 207

“Establishing remote trusted connections by DB2 for z/OS requesters” on page 208

“Establishing remote trusted connections to DB2 for z/OS servers” on page 209

### Related reference

“Establishing plan and package ownership in a trusted context” on page 81

“Ownership of objects within a trusted context” on page 79

---

## Trusted connections

A *trusted connection* is a database connection that is established when the connection attributes match those of a trusted context that is defined at the server. A trusted connection can be established locally or at a remote location.

A trusted context allows you to establish a trusted relationship between DB2 and an external entity, such as a middleware server. DB2 evaluates a series of trust attributes to determine if a specific context is to be trusted. Currently, the only attribute that DB2 considers is the database connection. The relationship between a connection and a trusted context is established when the connection to the server is first created, and that relationship remains as long as that connection exists.

### Related concepts

“Trusted contexts” on page 205

“Roles in a trusted context” on page 23

### Related tasks

“Defining trusted contexts”

“Creating local trusted connections” on page 207

“Establishing remote trusted connections by DB2 for z/OS requesters” on page 208

“Establishing remote trusted connections to DB2 for z/OS servers” on page 209

---

## Defining trusted contexts

Before you can create a trusted connection, you must define a trusted context by using a system authorization ID and connection trust attributes.

A system authorization ID is the DB2 primary authorization ID that is used to establish the trusted connection. For local connections, the system authorization ID is derived as shown in the following table.

*Table 51. System authorization ID for local connections*

Source	System authorization ID
Started task (RRSAF)	USER parameter on JOB statement or RACF USER
TSO	TSO logon ID
BATCH	USER parameter on JOB statement

For remote connections, the system authorization ID is derived from the system user ID that is provided by an external entity, such as a middleware server.

The connection trust attributes identify a set of characteristics about the specific connection. The connection trust attributes are required for the connection to be considered a trusted connection. For a local connection, the connection trust attribute is the job or started task name. For a remote connection, the connection trust attribute is the client's IP address, domain name, or SERVAUTH security zone name. The connection trust attributes are as follows:

#### **ADDRESS**

Specifies the client's IP address or domain name, used by the connection to communicate with DB2. The protocol must be TCP/IP.

#### **SERVAUTH**

Specifies the name of a resource in the RACF SERVAUTH class. This resource is the network access security zone name that contains the IP address of the connection to communicate with DB2.

#### **ENCRYPTION**

Specifies the minimum level of encryption of the data stream (network encryption) for the connection. Supported values are as follows:

- NONE - No encryption. This is the default.
- LOW - DRDA data stream encryption.
- HIGH - Secure Socket Layer (SSL) encryption.

#### **JOBNAME**

Specifies the local z/OS started task or job name. The value of JOBNAME depends on the source of the address space, as shown in the following table.

*Table 52. JOBNAME for local connections*

Source	JOBNAME
Started task (RRSAF)	Job or started task name
TSO	TSO logon ID
BATCH	Job name on JOB statement

The JOBNAME attribute cannot be specified with the ADDRESS, SERVAUTH, or ENCRYPTION attributes.

#### **Related concepts**

“Trusted contexts” on page 205

“Trusted connections” on page 206

“Roles in a trusted context” on page 23

#### **Related tasks**

“Creating local trusted connections”

“Establishing remote trusted connections by DB2 for z/OS requesters” on page 208

“Establishing remote trusted connections to DB2 for z/OS servers” on page 209

---

## **Creating local trusted connections**

You can establish a trusted connection to a local DB2 subsystem by using RRSAF or the DSN command processor under TSO and DB2I.

When you attempt to create a local trusted connection, DB2 searches for a trusted context that matches the primary authorization ID and the job or started task name

that you supply. If DB2 finds a matching trusted context, DB2 checks if the DEFAULT SECURITY LABEL attribute is defined in the trusted context.

If the DEFAULT SECURITY LABEL attribute is defined with a security label, DB2 verifies the security label with RACF. This security label is used for multilevel security verification for the system authorization ID. If verification is successful, the connection is established as trusted. If the verification is not successful, the connection is established as a normal connection without any additional privileges.

In addition, the DB2 online utilities can run in a trusted connection if a matching trusted context is defined, if the primary authorization ID matches the SYSTEM AUTHID value of the trusted context, and if the job name matches the JOBNAME attribute defined for the trusted context.

#### **Related concepts**

“Trusted contexts” on page 205

“Trusted connections” on page 206

“Roles in a trusted context” on page 23

#### **Related tasks**

“Defining trusted contexts” on page 206

“Establishing remote trusted connections by DB2 for z/OS requesters”

“Establishing remote trusted connections to DB2 for z/OS servers” on page 209

---

## **Establishing remote trusted connections by DB2 for z/OS requesters**

A DB2 for z/OS requester can establish a trusted connection to a remote location by setting up the new TRUSTED column in the SYSIBM.LOCATIONS table.

How DB2 obtains the system authorization ID to establish the trusted connection depends on the value of the SECURITY\_OUT option in the SYSIBM.IPNAMES table. The SECURITY\_OUT option in the SYSIBM.IPNAMES table must be 'E', 'P', or 'R'.

When the z/OS requester receives an SQL CONNECT with or without the USER and USING clauses to a remote location or if an application references a remote table or procedure, DB2 looks at the SYSIBM.LOCATIONS table to find a matching row. If DB2 finds a matching row, it checks the TRUSTED column. If the value of TRUSTED column is set to 'Y', DB2 looks at the SYSIBM.IPNAMES table. The values in the SECURITY\_OUT column and USERNAMES column are used to determine the system authorization ID as follows:

#### **SECURITY\_OUT = 'P' or 'E' and USERNAMES = 'S'**

The system authorization ID credentials that are used to establish the trusted connection are obtained from the row in the SYSIBM.USERNAMES table with TYPE 'S'.

DB2 sends the user switch request on behalf of the primary authorization ID without authentication under two conditions. First, the system authorization ID value in the AUTHID column is different from the primary authorization ID. Second, a trusted connection is successfully established.

#### **SECURITY\_OUT='P' or 'E' and USERNAMES = 'O'**

If a row with TYPE 'S' is defined in the SYSIBM.USERNAMES table, the system authorization ID credentials that are used to establish the trusted connection are obtained from the row.

After successfully establishing the trusted connection, DB2 obtains the translated authorization ID information for the primary authorization ID from the row in the SYSIBM.USERNAMES table with TYPE 'O'. DB2 sends the user switch request on behalf of the primary authorization ID with authentication.

If a row with TYPE 'S' is not defined in the SYSIBM.USERNAMES table, DB2 obtains the system authorization ID information that is used to establish the trusted connection from the row in the SYSIBM.USERNAMES table with TYPE 'O'.

**SECURITY\_OUT = 'R' and USERNAMES = ''**

The primary authorization ID is used as the system authorization ID to establish the trusted connection.

**SECURITY\_OUT = 'R' and USERNAMES = 'S'**

The system authorization ID that is used to establish the trusted connection is obtained from the row in the SYSIBM.USERNAMES table with TYPE='S'.

After establishing the trusted connection successfully, DB2 sends the user switch request on behalf of the primary authorization ID without authentication.

**SECURITY\_OUT = 'R' and USERNAMES = 'O'**

The system authorization ID that is used to establish the trusted connection is obtained from the row in the SYSIBM.USERNAMES table with TYPE 'S'.

After successfully establishing the trusted connection, DB2 obtains the translated authorization ID for the primary authorization ID from the row in the SYSIBM.USERNAMES table with TYPE 'O'. DB2 sends the user switch request on behalf of the primary authorization ID with RACF passticket authentication.

If the SECURITY\_OUT option is not correctly set up, DB2 returns an error.

**Related concepts**

“Trusted contexts” on page 205

“Trusted connections” on page 206

“Roles in a trusted context” on page 23

**Related tasks**

“Defining trusted contexts” on page 206

“Creating local trusted connections” on page 207

“Establishing remote trusted connections to DB2 for z/OS servers”

---

## Establishing remote trusted connections to DB2 for z/OS servers

When the DB2 for z/OS server receives a remote request to establish a trusted connection, DB2 checks to see if an authentication token accompanies the request.

The authentication token can be a password, a RACF passticket, or a Kerberos ticket. The requester goes through the standard authorization processing at the server. If the authorization is successful, DB2 invokes the connection exit routine, which associates the primary authorization ID, possibly one or more secondary authorization IDs, and an SQL ID with the remote request. DB2 searches for a matching trusted context. If DB2 finds a matching trusted context, it validates the following attributes:

- If the SERVAUTH attribute is defined for the identified trusted context and TCP/IP provides a RACF SERVAUTH profile name to DB2 during the establishment of the connection, DB2 matches the SERVAUTH profile name with the SERVAUTH attribute value.
- If the SERVAUTH attribute is not defined or the SERVAUTH name does not match the SERVAUTH that is defined for the identified trusted context, DB2 matches the remote client's TCP/IP address with the ADDRESS attribute that is defined for the identified trusted context.
- If the ENCRYPTION attribute is defined, DB2 validates whether the connection is using the proper encryption as specified in the value of the ENCRYPTION attribute.
- If the DEFAULT SECURITY LABEL attribute is defined for the system authorization ID, DB2 verifies the security label with RACF. This security label is used for verifying multilevel security for the system authorization ID. However, if the system authorization ID is also in the ALLOW USER clause with SECURITY LABEL, then that one is used.

If the validation is successful, DB2 establishes the connection as trusted. If the validation is not successful, the connection is established as a normal connection without any additional privileges, DB2 returns a warning, and SQLWARN8 is set.

#### **Related concepts**

“Trusted contexts” on page 205

“Trusted connections” on page 206

“Roles in a trusted context” on page 23

#### **Related tasks**

“Defining trusted contexts” on page 206

“Creating local trusted connections” on page 207

“Establishing remote trusted connections by DB2 for z/OS requesters” on page 208

---

## **Switching users of a trusted connection**

When a trusted connection is established, DB2 enables the trusted connection to be reused by a different user on a transaction boundary.

You can reuse a trusted connection at a local DB2 subsystem by using RRSAF, the DSN command processor under TSO, DB2I, and the SQL CONNECT statement with the USER and USING clauses. To reuse the trusted connection, you must add the specific user to the trusted context. If you specify 'PUBLIC' as the user, DB2 allows the trusted connection to be used by any authorization ID; the trusted connection can be used by a different user with or without authentication. However, you can require authentication by specifying the WITH AUTHENTICATION clause.

You can use RRSAF, the DSN command processor under TSO, and DB2I to switch to a new user on a trusted connection without authentication. If authentication is required, you can use the SQL CONNECT statement with the USER and USING clauses. The SQL CONNECT semantics prevent the use of CONNECT TO with the USER and USING clauses to switch authorization IDs on a remote connection.



### Related tasks

“Enabling users to perform actions on behalf of others” on page 213

“Performing tasks on objects for other users” on page 214

## Reusing a local trusted connection through the DSN command processor and DB2I

You can use the DSN command processor and DB2I to switch the user on a trusted connection if the DSN ASUSER option is specified.

DB2 establishes a trusted connection if the primary authorization ID and job name match a trusted context that is defined in DB2. The user ID that is specified for the ASUSER option goes through the standard authorization processing. If the user ID is authorized, DB2 runs the connection exit routine to associate the primary and secondary IDs.

DB2 then searches to see if the primary authorization ID is allowed to use the trusted connection without authentication. If the primary authorization ID is allowed to use the trusted connection without authentication, DB2 determines if the SECURITY LABEL attribute is defined in the trusted context for the user either explicitly or implicitly. If the SECURITY LABEL attribute is defined with a security label, DB2 verifies the security label with RACF. If the verification of the security label is successful, the trusted connection is established and used by the user ID that is specified for the ASUSER option. DB2 uses the security label for multilevel security verification for the user.

If the primary authorization ID that is associated with the user ID that is specified for the ASUSER option is not allowed or requires authentication information, the connection request fails. If the security label verification is not successful, the connection request fails.

## Reusing a remote trusted connection by DB2 for z/OS requesters

After establishing a trusted connection with a system authorization ID, the DB2 for z/OS requester automatically switches the user on the connection to the primary authorization ID on the remote trusted connection.

The DB2 for z/OS requester reuses a remote trusted connection in the following scenarios:

- The system authorization ID is different from the primary authorization ID that is associated with the application user.
- The system authorization ID is different from the authorization ID that is specified in the SQL CONNECT statement with the USER and USING clauses.
- Outbound translation is required for the primary authorization ID.

## Reusing a remote trusted connection through DB2 for z/OS servers

The DB2 for z/OS server performs a sequence of tasks when it receives a request to switch users on a trusted connection.

1. DB2, on successful authorization, invokes the connection exit routine. The invocation associates the primary authorization ID, possibly one or more secondary authorization IDs, and an SQL ID with the remote request. This new set of IDs replaces the previous set of IDs that was associated with the request.

2. DB2 determines if the primary authorization ID is allowed to use the trusted connection. If the WITH AUTHENTICATION clause is specified for the user, DB2 requires an authentication token for the user. The authentication token can be a password, a RACF passticket, or a Kerberos ticket.
3. Assuming that the primary authorization ID is allowed, DB2 determines the trusted context for any SECURITY LABEL definition. If a specific SECURITY LABEL is defined for this user, it becomes the SECURITY LABEL for this user. If no specific SECURITY LABEL is defined for this user but a DEFAULT SECURITY LABEL is defined for the trusted context, DB2 verifies the validity of this SECURITY LABEL for this user through RACF by issuing the RACROUTE VERIFY request.  

If the primary authorization ID is allowed, DB2 performs a connection initialization. This results in an application environment that truly mimics the environment that is initialized if the new user establishes the connection in the normal DB2 manner. For example, any open cursor is closed, and temporary table information is dropped.
4. If the primary authorization ID is not allowed to use the trusted connection or if SECURITY LABEL verification fails, the connection is returned to an unconnected state. The only operation that is allowed is to establish a valid authorization ID to be associated with the trusted connection. Until a valid authorization is established, if any SQL statement is issued, an error (SQLCODE -900) is returned.

## Reusing a local trusted connection through RRSF

If you use Resource Recovery Services Attachment Facility (RRSAF) to switch to a new user on a trusted connection, DB2 obtains the primary authorization ID and runs the sign-on exit routine.

DB2 then searches to determine if the primary authorization ID is allowed to use the trusted connection without authentication. If the primary authorization ID is allowed, DB2 determines if SECURITY LABEL is explicitly or implicitly defined in the trusted context for the user. If SECURITY LABEL is defined, DB2 verifies the SECURITY LABEL with RACF by using the RACROUTE VERIFY request. If the SECURITY LABEL verification is successful, the trusted connection is used by the new user.

If the primary authorization ID is not allowed to use the trusted connection without authentication, DB2 returns the connection to an unconnected state. The only action that you can take is to try running the sign-on exit routine again. Until a valid authorization is established, any SQL statement that you issue causes DB2 to return an error.

## Reusing a local trusted connection through the SQL CONNECT statement

You can switch users on a trusted connection by using the SQL CONNECT statement with the USER and USING clauses.

DB2, on successful authorization, invokes the connection exit routine if it is defined. The connection then has a primary authorization ID, zero or more secondary IDs, and an SQL ID.

DB2 searches to determine if the primary authorization ID is allowed to use the trusted connection. If the primary authorization ID is allowed, DB2 determines if the SECURITY LABEL attribute is defined in the trusted context for the user either

explicitly or implicitly. If the SECURITY LABEL attribute is defined with a security label, DB2 verifies the security label with RACF. If the security label verification is successful, DB2 switches the user on the trusted connection. DB2 uses the security label for multilevel security verification for the user.

If the primary authorization ID is not allowed to use the trusted connection or if the security label verification is not successful, DB2 returns the connection to an unconnected state. The only action you can take is to establish a valid authorization ID to be associated with the trusted connection. Until a valid authorization is established, any SQL statement that you issue causes DB2 to return an error.

---

## Defining external security profiles

You can control the users who can be switched in a trusted connection by defining an external security profile in RACF and authorizing users to use the profile.

To define an external security profile in RACF:

1. Create a general resource profile in RACF for the DSNR class by issuing the following command:  
`RDEFINE DSNR (TRUSTEDCTX.PROFILE1) UACC(NONE)`
2. Add users to the TRUSTEDCTX.PROFILE1 profile and define their level of access authority by issuing the following command:  
`PERMIT TRUSTEDCTX.PROFILE1 CLASS(DSNR) ID(USER1 USER2) ACCESS(READ)`
3. Associate the profile with the trusted context definition by using the EXTERNAL SECURITY PROFILE keyword in the trusted context user clause definition.

You can remove users who can be switched in a trusted connection individually from the TRUSTEDCTX.PROFILE1 profile in RACF. You can also remove all users by simply dissociating the profile from the trusted context definition.

---

## Enabling users to perform actions on behalf of others

Within a trusted context, you can allow users to perform actions on objects on behalf of others.

You can specify the DSN ASUSER option with the authorization ID of the object owner. During the connection processing, the authorization ID is used to determine if a trusted context exists for this authorization ID. If a trusted context exists, a trusted connection is established. The primary authorization ID that is associated with the user ID and specified in the ASUSER option is used to determine if the user can be switched on the trusted connection.

If the user ID that is specified in the ASUSER option is allowed to use the trusted connection, the user runs under the authorization ID of the object owner and can perform actions on behalf of the object owner. The authorization ID of the original user is traced for audit purposes.

### Related concepts

“Switching users of a trusted connection” on page 210

---

## Performing tasks on objects for other users

If you have DBADM authority, you can assume the identity of other users within a trusted context and perform tasks on their behalf.

After you successfully assume the identity of a view owner, you inherit all the privileges from the ID that owns the view and can therefore perform the CREATE, DROP, and GRANT actions on the view.

To perform tasks on behalf of another user:

1. Define a trusted context. Make sure that the SYSTEM AUTH ID is the primary authorization ID that you use in SPUFI.
2. Specify the primary authorization ID as the JOBNAME for the trusted connection
3. Specify the primary authorization ID of the user whose identity you want to assume
4. Log onto TSO with your primary authorization ID
5. Set the ASUSER option on the DB2I DEFAULTS panel to the primary authorization ID of the user whose identity you want to assume
6. Perform the desired actions by using privileges of the specified user.

Assume that you have DBADM authority, your primary authorization ID is BOB, and you want to drop a view that is owned by user SALLY. You can issue the following statement to create and enable a trusted context called CTXLOCAL in which BOB can drop the selected view on SALLY's behalf:

```
CREATE TRUSTED CONTEXT CTXLOCAL  
  BASED UPON CONNECTION USING SYSTEM AUTHID BOB  
  ATTRIBUTES (JOBNAME 'BOB')  
  ENABLE  
  ALLOW USE FOR SALLY;
```

After logging onto TSO, you can set the ASUSER option to SALLY in the DB2I DEFAULTS panel and invoke SPUFI to process SQL statements. DB2 obtains the primary authorization ID BOB and JOBNAME BOB from the TSO log-on session, authenticates BOB, searches for the matching trusted context (CTXLOCAL), and establishes a trusted connection. DB2 then authenticates the primary authorization ID SALLY and validates all privileges that are assigned to SALLY. After successful authentication and validation, you, BOB, can drop the view that is owned by SALLY.

### Related concepts

“Switching users of a trusted connection” on page 210

---

## Chapter 6. Managing access through data definition control

*Data definition control* is a DB2 security measure that provides additional constraints to existing authorization checks. You can use data definition control to manage access to your DB2 data.

---

### Data definition statements

Data definition control support can control data definition statements.

The following data definition statements are controlled through the DB2 data definition control support.

*Table 53. Data definition Statements*

Object	CREATE statement	ALTER statement	DROP statement
Alias	CREATE ALIAS		DROP ALIAS
Database	CREATE DATABASE	ALTER DATABASE	DROP DATABASE
Index	CREATE INDEX	ALTER INDEX	DROP INDEX
Storage group	CREATE STOGROUP	ALTER STOGROUP	DROP STOGROUP
Synonym	CREATE SYNONYM		DROP SYNONYM
Table	CREATE TABLE	ALTER TABLE	DROP TABLE
Table space	CREATE TABLESPACE	ALTER TABLESPACE	DROP TABLESPACE
View	CREATE VIEW		DROP VIEW

The data definition control support also controls the COMMENT and LABEL statements.

#### Related concepts

“Registration tables” on page 216

#### Related reference

“Data definition control support”

---

### Data definition control support

If you want to use data definition statements for your plans and packages, you must install *data definition control support* on the DB2 DSNTIPZ installation panel.

As shown in the following example, you can specify appropriate values for several installation options to install the data definition control support and to control data definition behaviors.

```

DSNTIPZ      INSTALL DB2 - DATA DEFINITION CONTROL SUPPORT
===>

Enter data below:

 1 INSTALL DD CONTROL SUPT. ===> NO      YES - activate the support
                                         NO - omit DD control support
 2 CONTROL ALL APPLICATIONS ===> NO      YES or NO
 3 REQUIRE FULL NAMES      ===> YES      YES or NO
 4 UNREGISTERED DDL DEFAULT ===> ACCEPT  Action for unregistered DDL:
                                         ACCEPT - allow it
                                         REJECT - prohibit it
                                         APPL - consult ART
                                         Used in ART/ORT Searches
 5 ART/ORT ESCAPE CHARACTER ===>
 6 REGISTRATION OWNER      ===> DSNRGCOL  Qualifier for ART and ORT
 7 REGISTRATION DATABASE   ===> DSNRGFDB  Database name
 8 APPL REGISTRATION TABLE ===> DSN_REGISTER_APPL Table name
 9 OBJT REGISTRATION TABLE ===> DSN_REGISTER_OBJT Table name

Note: ART = Application Registration Table
      ORT = Object      Registration Table

PRESS: ENTER to continue  RETURN to exit  HELP for more information

```

Figure 12. DSNTIPZ installation panel with default values

### Related concepts

“Registration tables”

### Related reference

“Data definition statements” on page 215

“Data definition control support” on page 215

## Registration tables

If you use data definition control support, you must create and maintain an application registration table (ART) and an object registration table (ORT). You can register plans and package collections in the ART and objects that are associated with the plans and collections in the ORT.

DB2 consults these two registration tables before accepting a data definition statement from a process. It denies a request to create, alter, or drop a particular object if the registration tables indicate that the process is not allowed to do so.

Both ART and ORT contain the CREATOR and CHANGER columns. The CREATOR and CHANGER columns are CHAR(26) and large enough for a three-part authorization ID. You need to separate each 8-byte part of the ID with a period in byte 9 and in byte 18. If you enter only the primary authorization ID, consider entering it right-justified in the field (that is, preceded by 18 blanks).

In addition to the CREATOR and CHANGER columns, an ART also contains the following columns, some of which are optional and reserved for administrator use; DB2 does not use these columns.

Table 54. Columns of the ART

Column name	Description
APPLIDENT	Indicates the collection-ID of the package that executes the data definition language. If no package exists, it indicates the name of the plan that executes the data definition language.
APPLIDENTTYPE	Indicates the type of application identifier.

Table 54. Columns of the ART (continued)

Column name	Description
APPLICATIONDESC <sup>1</sup>	Optional data. Provides a more meaningful description of each application than the eight-byte APPLIDENT column can contain.
DEFAULTAPPL	Indicates whether all data definition language should be accepted from this application.
QUALIFIEROK	Indicates whether the application can supply a missing name part for objects that are named in the ORT. Applies only if REQUIRE FULL NAMES = NO.
CREATOR <sup>1, 2</sup>	Optional data. Indicates the authorization ID that created the row.
CREATETIMESTAMP <sup>1</sup>	Optional data. Indicates when a row was created. If you use CURRENT TIMESTAMP, DB2 automatically enters the value of CURRENT TIMESTAMP When you load or insert a row.
CHANGER <sup>1, 2</sup>	Optional data. Indicates the authorization ID that last changed the row.
CHANGETIMESTAMP <sup>1</sup>	Optional data. Indicates when a row was changed. If you use CURRENT TIMESTAMP, DB2 automatically enters the value of CURRENT TIMESTAMP When you update a row.

An ORT also contains the following columns, some of which are optional and reserved for administrator use; DB2 does not use these columns.

Table 55. Columns of the ORT

Column name	Description
QUALIFIER	Indicates the object name qualifier.
NAME	Indicates the unqualified object name.
TYPE	Indicates the type of object.
APPLMATCHREQ	Indicates whether an application that names this object must match the one that is named in the APPLIDENT column.
APPLIDENT	Collection-ID of the plan or package that executes the data definition language.
APPLIDENTTYPE	Indicates the type of application identifier.
APPLICATIONDESC <sup>1</sup>	Optional data. Provides a more meaningful description of each application than the eight-byte APPLIDENT column can contain.
CREATOR <sup>1, 2</sup>	Optional data. Indicates the authorization ID that created the row.
CREATETIMESTAMP <sup>1</sup>	Optional data. Indicates when a row was created. If you use CURRENT TIMESTAMP, DB2 automatically enters the value of CURRENT TIMESTAMP When you load or insert a row.
CHANGER <sup>1, 2</sup>	Optional data. Indicates the authorization ID that last changed the row.
CHANGETIMESTAMP <sup>1</sup>	Optional data. Indicates when a row was changed. If you use CURRENT TIMESTAMP, DB2 automatically enters the value of CURRENT TIMESTAMP When you update a row.



#### Related reference

“Data definition statements” on page 215

“Data definition control support” on page 215

---

## Installing data definition control support

You can install data definition control support that is available through the DB2 DSN TIPZ installation panel.

To install data definition control support:

1. Enter YES for option 1 on the DSN TIPZ installation panel, as shown in the following example.

```
1 INSTALL DD CONTROL SUPT. ====> YES
```

2. Enter the names and owners of the registration tables in your DB2 subsystem and the databases in which these tables reside for options 6, 7, 8, and 9 on the DSN TIPZ installation panel.

The default values for these options are as follows:

```
6 REGISTRATION OWNER      ====> DSNRGCOL
7 REGISTRATION DATABASE    ====> DSNRGFDB
8 APPL REGISTRATION TABLE ====> DSN_REGISTER_APPL
9 OBJT REGISTRATION TABLE ====> DSN_REGISTER_OBJT
```

You can accept the default names or assign names of your own. If you specify your own table names, each name can have a maximum of 17 characters.

3. Enter an escape character for option 5 on the DSN TIPZ installation panel if you want to use the percent character (%) or the underscore character (\_) as a regular character in the ART or ORT.

You can use any special character other than underscore or percent as the escape character. For example, you can use the pound sign (#) as an escape character. If you do, the value for option looks like this:

```
5 ART/ORT ESCAPE CHARACTER ====> #
```

After you specify the pound sign as an escape character, the pound sign can be used in names in the same way that an escape character is used in an SQL LIKE predicate.

4. Register plans, packages, and objects in the ART and ORT.

Choose the plans, packages, and objects to register based on whether you want to control data definition by application name or object name.

5. Enter the values for the three other options on the DSN TIPZ installation panel as follows:

```
2 CONTROL ALL APPLICATIONS ====>
3 REQUIRE FULL NAMES          ====>
4 UNREGISTERED DDL DEFAULT    ====>
```

#### Related reference

“Data definition control support” on page 215

---

## Enabling data definition control

You can use data definition control after you install the DB2 data definition control support and create the application registration table (ART) and the object registration table (ORT).

You can use data definition control in the following four ways:

- Controlling data definition by application name

- Controlling data definition by application name with exceptions
- Controlling data definition by object name
- Controlling data definition by object name with exceptions

#### Related tasks

“Disabling data definition control” on page 225

## Controlling data definition by application name

The simplest way to implement data definition control is to give one or more applications total control over the use of data definition statements in the subsystem.

To control data definition by application name, perform the following steps:

1. Enter YES for the first option on the DSNTIPZ installation panel, as shown:  
2 CONTROL ALL APPLICATIONS ==> YES

When you specify YES, only package collections or plans that are registered in the ART are allowed to use data definition statements.

2. In the ART, register all package collections and plans that you will allow to issue DDL statements, and enter the value Y in the DEFAULTAPPL column for these package collections. You must supply values for the APPLIDENT, APPLIDENTTYPE, and DEFAULTAPPL columns of the ART. You can enter information in other columns for your own use.

**Example:** Suppose that you want all data definition language in your subsystem to be issued only through certain applications. The applications are identified by the following application plan names, collection-IDs, and patterns:

#### PLANA

The name of an application plan

#### PACKB

The collection-ID of a package

#### TRULY%

A pattern name for any plan name beginning with TRULY

**TR%** A pattern name for any plan name beginning with TR

The following table shows the entries that you need in your ART.

Table 56. Table DSN\_REGISTER\_APPL for total subsystem control

APPLIDENT	APPLIDENTTYPE	DEFAULTAPPL
PLANA	P	Y
PACKB	C	Y
TRULY%	P	Y
TR%	P	Y

If the row with TR% for APPLIDENT contains the value Y for DEFAULTAPPL, any plan with a name beginning with TR can execute data definition language. If DEFAULTAPPL is later changed to N to disallow that use, the changed row does not prevent plans beginning with TR from using data definition language; the row merely fails to allow that specific use. In this case, the plan TRXYZ is not allowed to use data definition language. However, the plan TRULYXYZ is allowed to use data definition language, by the row with TRULY% specified for APPLIDENT.

## Controlling data definition by application name with exceptions

You can register an application name with exceptions in the application registration table (ART) as a way to control data definition.

To control data definition by application name with exceptions:

1. Choose not to control all applications. On the DSNTIPZ installation panel, specify the following value for option 2:

2 CONTROL ALL APPLICATIONS ==> NO

When you specify NO, you allow unregistered applications to use data definition statements on some objects.

2. On the DSNTIPZ installation panel, specify the following for option 4:

4 UNREGISTERED DDL DEFAULT ==> APPL

When you specify APPL, you restrict the use of data definition statements for objects that are **not** registered in the ORT. If an object is registered in the ORT, any applications that are not registered in the ART can use data definition language on the object. However, if an object is not registered in the ORT, only applications that are registered in the ART can use data definition language on the object.

3. In the ART, register package collections and plans that you will allow to issue data definition statements on any object. Enter the value Y in the DEFAULTAPPL column for these package collections. Applications that are registered in the ART retain almost total control over data definition. Objects that are registered in the ORT are the only exceptions.
4. In the ORT, register all objects that are exceptions to the subsystem data definition control that you defined in the ART. You must supply values for the QUALIFIER, NAME, TYPE, APPLMATCHREQ, APPLIDENT, and APPLIDENTTYPE columns of the ORT. You can enter information in other columns of the ORT for your own use.

**Example:** Suppose that you want almost all of the data definition language in your subsystem to be issued only through an application plan (PLANA) and a package collection (PACKB).

The following table shows the entries that you need in your ART.

Table 57. Table DSN\_REGISTER\_APPL for total subsystem control with exceptions

APPLIDENT	APPLIDENTTYPE	DEFAULTAPPL
PLANA	P	Y
PACKB	C	Y

However, suppose that you also want the following specific exceptions:

- Object KIM.VIEW1 can be created, altered, or dropped by the application plan PLANC.
- Object BOB.ALIAS can be created, altered, or dropped only by the package collection PACKD.
- Object FENG.TABLE2 can be created, altered, or dropped by **any** plan or package collection.

- Objects with names that begin with SPIFFY.MSTR and exactly one following character can be created, altered, or dropped by any plan that matches the name pattern TRULY%. For example, the plan TRULYJKL can create, alter, or drop the object SPIFFY.MSTRA.

The following table shows the entries that are needed to register these exceptions in the ORT.

*Table 58. Table DSN\_REGISTER\_OBJT for subsystem control with exceptions*

QUALIFIER	NAME	TYPE	APPLMATCHREQ	APPLIDENT	APPLIDENTTYPE
KIM	VIEW1	C	Y	PLANC	P
BOB	ALIAS	C	Y	PACKD	C
FENG	TABLE2	C	N		
SPIFFY	MSTR_	C	Y	TRULY%	P

You can register objects in the ORT individually, or you can register sets of objects.

## Controlling data definition by object name

You can register object names in the object registration table (ORT) as a way to control data definition. You need to control data definition by object names if you want all objects in the subsystem to be registered and if you want some applications to control specific sets of objects.

When you control data definition by object name, all objects are registered regardless of whether they are controlled by specific applications.

To control data definition by object name:

1. Choose not to control all applications. On the DSNTIPZ installation panel, specify the following value for option 2:  
2 CONTROL ALL APPLICATIONS ====> NO

When you specify NO, you allow unregistered applications to use data definition statements on some objects.

2. On the DSNTIPZ installation panel, fill in option 4 as follows:  
4 UNREGISTERED DDL DEFAULT ====> REJECT

When you specify REJECT for option 4, you totally restrict the use of data definition statements for objects that are not registered in the ORT. Therefore, no application can use data definition statements for any unregistered object.

3. In the ORT, register all of the objects in the subsystem, and enter Y in the APPLMATCHREQ column. You must supply values for the QUALIFIER, NAME, TYPE, APPLMATCHREQ, APPLIDENT, and APPLIDENTTYPE columns of the ORT. You can enter information in other columns of the ORT for your own use.
4. In the ART, register any plan or package collection that can use a set of objects that you register in the ORT with an incomplete name. Enter the value Y in the QUALIFIEROK column. These plans or package collections can use data definition language on sets of objects regardless of whether a set of objects has a value of Y in the APPLMATCHREQ column.

**Example:** The following table shows entries in the ORT for a DB2 subsystem that contains the following objects that are controlled by object name:

- Two storage groups (STOG1 and STOG2) and a database (DATB1) that are not controlled by a specific application. These objects can be created, altered, or dropped by a user with the appropriate authority by using any application, such as SPUFI or QMF.
- Two table spaces (TBSP1 and TBSP2) that are not controlled by a specific application. Their names are qualified by the name of the database in which they reside (DATB1).
- Three objects (OBJ1, OBJ2, and OBJ3) whose names are qualified by the authorization IDs of their owners. Those objects might be tables, views, indexes, synonyms, or aliases. Data definition statements for OBJ1 and OBJ2 can be issued only through the application plan named PLANX. Data definition statements for OBJ3 can be issued only through the package collection named PACKX.
- Objects that match the qualifier pattern E%D and the name OBJ4 can be created, altered, or deleted by application plan SPUFI. For example, the objects EDWARD.OBJ4, ED.OBJ4, and EBHARD.OBJ4, can be created, altered, or deleted by application plan SPUFI. Entry E%D in the QUALIFIER column represents all three objects.
- Objects with names that begin with TRULY.MY\_, where the underscore character is actually part of the name. Assuming that you specify # as the escape character, all of the objects with this name pattern can be created, altered, or dropped only by plans with names that begin with TRULY.

Table 59. Table DSN\_REGISTER\_OBJT for total control by object

QUALIFIER	NAME	TYPE	APPLMATCHREQ	APPLIDENT	APPLIDENTTYPE
	STOG1	S	N		
	STOG2	S	N		
	DATB1	D	N		
DATB1	TBSP1	T	N		
DATB1	TBSP2	T	N		
KIM	OBJ1	C	Y	PLANX	P
FENG	OBJ2	C	Y	PLANX	P
QUENTIN	OBJ3	C	Y	PACKX	C
E%D	OBJ4	C	Y	SPUFI	P
TRULY	MY#_%	C	Y	TRULY%	P

Assume the following installation option:

```
3 REQUIRE FULL NAMES      ==> YES
```

The entries do not specify incomplete names. Hence, objects that are not represented in the table cannot be created in the subsystem, except by an ID with installation SYSADM authority.

## Controlling data definition by object name with exceptions

You can register an object name with exceptions in the object registration table (ORT) as a way to control data definition.

You can allow some applications to control specific sets of registered objects while allowing other applications to use data definition statements for unregistered objects.

To control data definition by object name with exceptions:

1. Choose not to control all applications. On the DSNTIPZ installation panel, specify the following value for option 2:  
2 CONTROL ALL APPLICATIONS ==> NO

When you specify NO, you allow unregistered applications to use data definition statements on some objects.

2. On the DSNTIPZ installation panel, fill in option 4 as follows:  
4 UNREGISTERED DDL DEFAULT ==> ACCEPT

This option **does not** restrict the use of data definition statements for objects that are not registered in the ORT. Therefore, **any** application can use data definition language for any unregistered object.

3. Register all controlled objects in the ORT. Use a name and qualifier to identify a single object. Use only one part of a two-part name to identify a set of objects that share just that part of the name. For each controlled object, use APPLMATCHREQ = Y. Enter the name of the plan or package collection that controls the object in the APPLIDENT column.
4. For each set of controlled objects (identified by only a simple name in the ORT), register the controlling application in the ART. You must supply values for the APPLIDENT, APPLIDENTTYPE, and QUALIFIEROK columns of the ART.

**Example:** The following two tables assume that the installation option REQUIRE FULL NAMES is set to NO. The following table shows entries in the ORT for the following controlled objects:

Table 60. Table DSN\_REGISTER\_OBJT for object control with exceptions

QUALIFIER	NAME	TYPE	APPLMATCHREQ	APPLIDENT	APPLIDENTTYPE
KIM	OBJ1	C	Y	PLANX	P
FENG	OBJ2	C	Y	PLANX	P
QUENTIN	OBJ3	C	Y	PACKX	C
EDWARD	OBJ4	C	Y	PACKX	C
	TABA	C	Y	PLANA	P
	TABB	C	Y	PACKB	C

- The objects KIM.OBJ1, FENG.OBJ2, QUENTIN.OBJ3, and EDWARD.OBJ4, all of which are controlled by PLANX or PACKX. DB2 cannot interpret the object names as incomplete names because the objects that control them, PLANX and PACKX, are registered, with QUALIFIEROK=N, in the corresponding ART as shown in the following table:

Table 61. Table DSN\_REGISTER\_APPL for object control with exceptions

APPLIDENT	APPLIDENTTYPE	DEFAULTAPPL	QUALIFIEROK
PLANX	P	N	N
PACKX	C	N	N
PLANA	P	N	Y
PACKB	C	N	Y

In this situation, with the combination of installation options shown previously, any application can use data definition language for objects that are not covered

by entries in the ORT. For example, if HOWARD has the CREATETAB privilege, HOWARD can create the table HOWARD.TABLE10 through any application.

- Two sets of objects, \*.TABA and \*.TABB, are controlled by PLANA and PACKB, respectively.

## Registering object sets

Registering object sets enables you to save time and to simplify object registration.

Registering object sets is not a data definition control method; you must install of the data definition control methods before you can register any object sets.

Because complete two-part names are not required for every object that is registered in the ORT, you can use incomplete names to register sets of objects. To use incomplete names and register sets of objects, fill in option 3 on the DSNTIPZ installation panel as follows:

```
3 REQUIRE FULL NAMES      ==> NO
```

The default value YES requires you to use both parts of the name for each registered object. If you specify the value NO, an incomplete name in the ORT represents a set of objects that all share the same value for one part of a two-part name. Objects that are represented by incomplete names in the ORT require an authorizing entry in the ART.

**Example:** If you specify NO for option 3, you can include entries with incomplete names in the ORT. The following table shows entries in the ORT for the following objects:

Table 62. Table DSN\_REGISTER\_OBJT for objects with incomplete names

QUALIFIER	NAME	TYPE	APPLMATCHREQ	APPLIDENT	APPLIDENTTYPE
	TABA	C	Y	PLANX	P
	TABB	C	Y	PACKY	C
SYSADM		C	N		
DBSYSADM		T	N		
USER1	TABLEX	C	N		

- Two sets of objects, \*.TABA and \*.TABB, which are controlled by PLANX and PACKY, respectively. Only PLANX can create, alter, or drop any object whose name is \*.TABA. Only PACKY can create, alter, or drop any object whose name is \*.TABB. PLANX and PACKY must also be registered in the ART with QUALIFIEROK set to Y, as shown in the following table: That setting allows the applications to use sets of objects that are registered in the ORT with an incomplete name.

Table 63. Table DSN\_REGISTER\_APPL for plans that use sets of objects

APPLIDENT	APPLIDENTTYPE	DEFAULTAPPL	QUALIFIEROK
PLANA	P	N	Y
PACKB	C	N	Y

- Tables, views, indexes, or aliases with names like SYSADM.\*.
- Table spaces with names like DBSYSADM.\*; that is, table spaces in database DBSYSADM.
- Tables with names like USER1.\* **and** tables with names like \*.TABLEX.



*ART entries for objects with incomplete names in the ORT:* APPLMATCHREQ=N and objects SYSADM.\*, DBSYSADM.\*, USER1.\*, and \*.TABLEX can be created, altered, or dropped by any package collection or application plan. However, the collection or plan that creates, alters, or drops such an object must be registered in the ART with QUALIFIEROK=Y to allow it to use incomplete object names.

---

## Disabling data definition control

When data definition control is active, only IDs with the installation SYSADM or installation SYSOPR authority can stop a database, a table space, or an index space that contains a registration table or index.

When the object is stopped, only an ID with one of those authorities can start it again.

An ID with the installation SYSADM authority can execute data definition statements regardless of whether data definition control is active and whether the ART or ORT is available. To bypass data definition control, an ID with the installation SYSADM authority can use the following methods:

- If the ID is the owner of the plan or package that contains the statement, the ID can bypass data definition control by using a static SQL statement.
- If the ID is the current SQL ID, the ID can bypass data definition control through a dynamic CREATE statement.
- If the ID is the current SQL ID, the primary ID, or any secondary ID of the executing process, the ID can bypass data definition control through a dynamic ALTER or DROP statement.

### Related tasks

“Enabling data definition control” on page 218

---

## Managing registration tables and indexes

You can create, update, and drop registration tables and indexes. You can also create table spaces for or add columns to registration tables.

### Creating registration tables and indexes

When you install data definition control support, you create the application registration table (ART), the object registration table (ORT), and the unique indexes that are required on the tables. You can re-create these objects if you drop any of them.

You can use the following CREATE statements to recreate ART, the ORT, or the required unique indexes:

#### CREATE statements for the ART and its index:

##### GUIP

```
CREATE TABLE DSNRGCOL.DSN_REGISTER_APPL
  (APPLIDENT      VARCHAR(128) NOT NULL WITH DEFAULT,
   APPLIDENTTYPE  CHAR(1)      NOT NULL WITH DEFAULT,
   APPLICATIONDESC VARCHAR(30)  NOT NULL WITH DEFAULT,
   DEFAULTAPPL    CHAR(1)      NOT NULL WITH DEFAULT,
   QUALIFIEROK    CHAR(1)      NOT NULL WITH DEFAULT,
   CREATOR        CHAR(26)     NOT NULL WITH DEFAULT,
```

```

    CREATETIMESTAMP    TIMESTAMP    NOT NULL WITH DEFAULT,
    CHANGER            CHAR(26)      NOT NULL WITH DEFAULT,
    CHANGETIMESTAMP    TIMESTAMP    NOT NULL WITH DEFAULT)
IN DSNRGFDB.DSNRGFTS;

CREATE UNIQUE INDEX DSNRGCOL.DSN_REGISTER_APPLI
ON DSNRGCOL.DSN_REGISTER_APPL
(APPLIDENT, APPLIDENTTYPE, DEFAULTAPPL DESC, QUALIFIEROK DESC)
CLUSTER;

```

#### GUIP

**CREATE statements for the ORT and its index:**

#### GUIP

```

CREATE TABLE DSNRGCOL.DSN_REGISTER_OBJT
(QUALIFIER    CHAR(8)      NOT NULL WITH DEFAULT,
NAME          CHAR(18)     NOT NULL WITH DEFAULT,
TYPE          CHAR(1)      NOT NULL WITH DEFAULT,
APPLMATCHREQ  CHAR(1)      NOT NULL WITH DEFAULT,
APPLIDENT     VARCHAR(128) NOT NULL WITH DEFAULT,
APPLIDENTTYPE CHAR(1)      NOT NULL WITH DEFAULT,
APPLICATIONDESC VARCHAR(30) NOT NULL WITH DEFAULT,
CREATOR       CHAR(26)     NOT NULL WITH DEFAULT,
CREATETIMESTAMP TIMESTAMP  NOT NULL WITH DEFAULT,
CHANGER       CHAR(26)     NOT NULL WITH DEFAULT,
CHANGETIMESTAMP TIMESTAMP  NOT NULL WITH DEFAULT)
IN DSNRGFDB.DSNRGFTS;

CREATE UNIQUE INDEX DSNRGCOL.DSN_REGISTER_OBJTI
ON DSNRGCOL.DSN_REGISTER_OBJT
(QUALIFIER, NAME, TYPE) CLUSTER;

```

#### GUIP

You can alter these CREATE statements in the following ways:

- Add columns to the ends of the tables
- Assign an auditing status
- Choose buffer pool or storage options for indexes
- Declare table check constraints to limit the types of entries that are allowed

## Naming registration tables and indexes

Every member of a data sharing group must have the same names for the application registration table (ART) and object registration tables (ORT) table. Avoid changing the names of the ART and ORT tables.

If you change the names, owners, or residing database of your ART and ORT, you must reinstall DB2 in update mode and make the corresponding changes on the DSN TIPZ installation panel.

Name the required index by adding the letter I to the corresponding table name. For example, suppose that you are naming a required index for the ART named ABC. You should name the required index ABCI.

## Dropping registration tables and indexes

If you drop any of the registration tables or their indexes, most data definition statements are rejected until the dropped objects are re-created.

The only data definition statements that are allowed under such circumstances are those that create the following objects:

- Registration tables that are defined during installation
- Indexes of the registration tables that are defined during installation
- Table spaces that contain the registration tables that are defined during installation
- The database that contains the registration tables that are defined during installation

## Creating table spaces for registration tables

The DSNTIJSJG installation job creates a segmented table space that holds the application registration table (ART) and the object registration table (ORT):

You can issue the following statement to create the table space:

```
CREATE TABLESPACE DSNRGFTS IN DSNRGFDB SEGSIZE 4 CLOSE NO;
```

If you want to use a table space with a different name or different attributes, you can modify the DSNTIJSJG job before installing DB2. Alternatively, you can drop the table space and re-create it, the ART and ORT tables, and their indexes.

## Adding columns to registration tables

You can use the ALTER TABLE statement to add columns to the ART or ORT for your own use. If you add columns, the additional columns must come at the end of the table, after existing columns.

Use a special character, such as the plus sign (+), in your column names to avoid possible conflict. If IBM adds columns to the ART or the ORT in future releases, the column names will contain only letters and numbers.

## Updating registration tables

You can use the LOAD utility or the INSERT, UPDATE, or DELETE statements to update the application registration table (ART) and the object registration table (ORT).

Because security provisions are important, allow only a restricted set of authorization IDs, or perhaps only those with the SYSADM authority, to update the ART. Consider assigning a validation exit routine to the ORT, to allow applications to change only those rows that have the same application identifier in the APPLIDENT column.

A registration table cannot be updated until all jobs whose data definition statements are controlled by the table have completed.



---

## Chapter 7. Managing access through exit routines

You can control access to DB2 by using a DB2-supplied exit routine or an exit routine that you write. DB2 provides installation-wide exit points to the connection, sign-on, and access control authorization routines.

### Related concepts

 General guidelines for writing exit routines (DB2 Administration Guide)


### Related information

 Exit routines (DB2 Administration Guide)

---


## Connection routines and sign-on routines

Your DB2 subsystem has two exit points for authorization routines, one in connection processing and one in sign-on processing. Both exit points perform crucial steps in the assignment of values to primary IDs, secondary IDs, and SQL IDs.

 You must have a routine for each exit. Default routines are provided for both. DSN3@ATH is the default exit routine for connections, and DSN3@SGN is the default exit routine for sign-ons.

If your installation has a connection exit routine and you plan to use CONNECT with the USER/USING clause, you should examine your exit routine. DB2 does not update the following information to reflect the user ID and password that are specified in the USER/USING clause of the CONNECT statement:

- The security-related control blocks that are normally associated with the thread
- The address space that your exit routine can access

If you want to use secondary authorization IDs, you must replace the default routines with the sample routines, or with routines of your own. 

### Related concepts

 General guidelines for writing exit routines (DB2 Administration Guide)

### Related tasks

“Using sample connection and sign-on exit routines for CICS transactions” on page 159

“Specifying connection and sign-on routines”

“Debugging connection and sign-on routines” on page 238

### Related reference

“Processing of connection requests” on page 154

“Processing of sign-on requests” on page 157

“Sample connection and sign-on routines” on page 230

“Exit parameter list for connection and sign-on routines” on page 231

## Specifying connection and sign-on routines

Your connection routine must have a CSECT name and entry point of DSN3@ATH. The name of the load module for the connection routine can be the same name, or

it can be a different name. Your sign-on routine must have a CSECT name and entry point of DSN3@SGN. The name of the load module for the sign-on routine can be the same, or it can be a different name.

**PSPI** You can use an ALIAS statement of the linkage editor to provide the entry-point name.

Default routines exist in library *prefix.SDSNLOAD*. To use your routines instead, place your routines in library *prefix.SDSNEXIT*. You can use the install job DSN3@SGN to assemble and link-edit the routines and place them in the new library. If you use any other library, you might need to change the STEPLIB or JOBLIB concatenations in the DB2 start-up procedures.

You can combine both routines into one CSECT and load module if you wish, but the module must include both entry points, DSN3@ATH and DSN3@SGN. Use standard assembler and linkage editor control statements to define the entry points. DB2 loads the module twice at startup, by issuing the z/OS LOAD macro first for entry point DSN3@ATH and then for entry point DSN3@SGN. However, because the routines are reentrant, only one copy of each remains in virtual

storage. **PSPI**

#### Related concepts

“Connection routines and sign-on routines” on page 229

#### Related reference

“Processing of connection requests” on page 154

“Processing of sign-on requests” on page 157

“Sample connection and sign-on routines”

“Exit parameter list for connection and sign-on routines” on page 231

## Sample connection and sign-on routines

The sample DB2 exit routines are provided in the source code as members of *prefix.SDSNSAMP*.

**PSPI** To examine the sample connection routine, list or assemble member DSN3SATH. To examine the sample sign-on routine, list or assemble member DSN3SSGN. You must use the High Level Assembler to assemble them.

**Change required for some CICS users:** You must change the sample sign-on exit routine (DSN3SSGN) before assembling and using it, if the following conditions are true:

- You attach to DB2 with an AUTH parameter other than AUTH=GROUP.
- You have the RACF list-of-groups option active.
- You have transactions whose initial primary authorization ID is not defined to RACF

To change the sample sign-on exit routine (DSN3SSGN), perform the following steps:

1. Locate the following statement in DSN3SSGN as a reference point:  
SSGN035 DS OH BLANK BACKSCAN LOOP REENTRY
2. Locate the following statement, which comes after the reference point:  
B SSGN037 ENTIRE NAME IS BLANK, LEAVE

3. Replace the statement with the following statement:

```
B          SSGN090      NO GROUP NAME...  BYPASS RACF CHECK
```

By changing the statement, you avoid an abend with SQLCODE -922. The routine with the new statement provides no secondary IDs unless you use AUTH=GROUP.

**PSPI**

#### Related concepts

“Connection routines and sign-on routines” on page 229

#### Related tasks

“Using sample connection and sign-on exit routines for CICS transactions” on page 159

“Specifying connection and sign-on routines” on page 229

“Debugging connection and sign-on routines” on page 238

#### Related reference

“Processing of sign-on requests” on page 157

## When connection and sign-on routines are taken

Different local processes enter the access control procedure at different points, depending on the environment from which they originate. Different criteria apply to remote requests.

**PSPI**

The following processes go through connection processing only:

- Requests that originate in TSO foreground and background (including online utilities and requests through the call attachment facility)
- JES-initiated batch jobs
- Requests through started task control address spaces (from the MVS START command)

The following processes go through connection processing, and can later go through the sign-on exit:

- The IMS control region
- The CICS recovery coordination task
- DL/I batch
- Requests through the Resource Recovery Services attachment facility (RRSAF)

The following processes go through sign-on processing:

- Requests from IMS-dependent regions (including MPP, BMP, and Fast Path)
- CICS transaction subtasks
- Scheduled tasks that are executed by the DB2 administrative task scheduler

**PSPI**

## Exit parameter list for connection and sign-on routines

The parameter list of connection and sign-on routines contains pointers to other information, such as the authorization ID list.

**PSPI**

The following diagram shows how the parameter list points to other information.



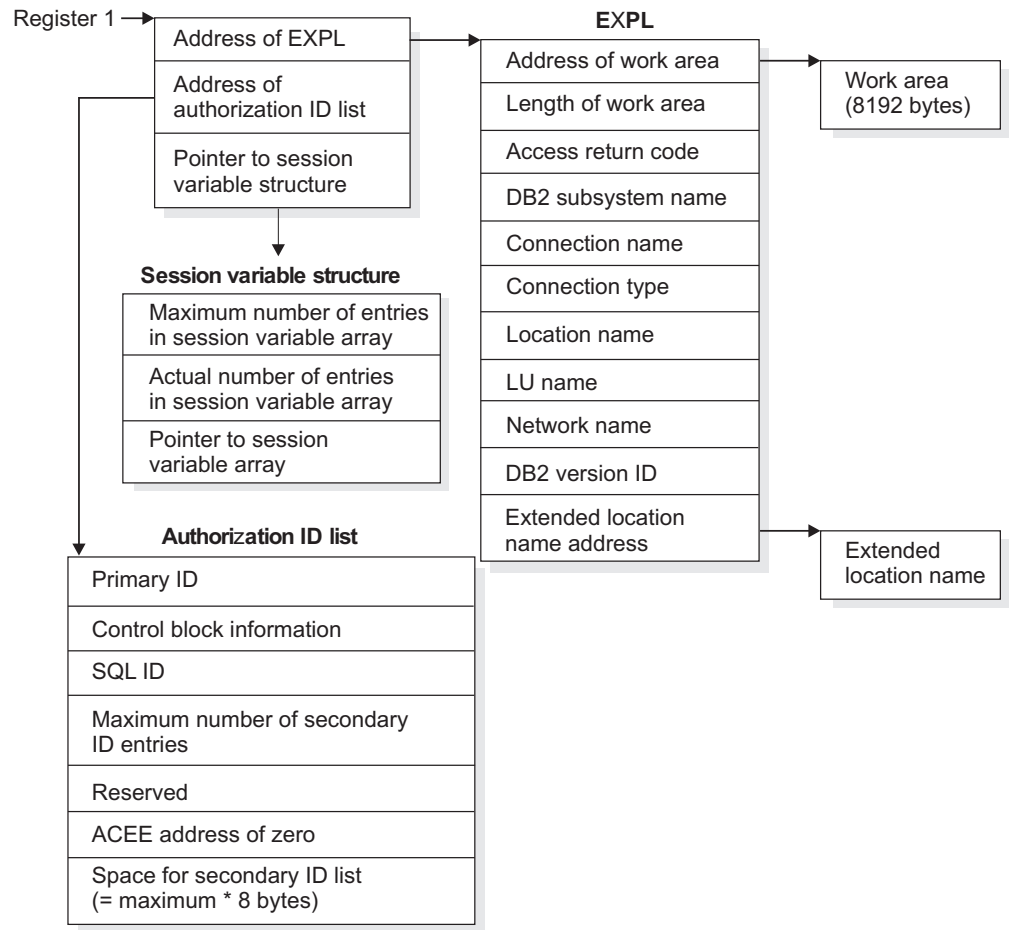


Figure 13. How a connection or sign-on parameter list points to other information

Connection routines and sign-on routines use 28 more bytes of the exit parameter list EXPL than other routines. The following table shows the entire list of connection routines and sign-on routines. The exit parameter list is described by macro DSNDEXPL.

Table 64. Exit parameter list for connection routines and sign-on routines

Name	Hex offset	Data type	Description
EXPLWA	0	Address	Address of a 8192-byte work area to be used by the routine.
EXPLWL	4	Signed 4-byte integer	Length of the work area, in bytes; value is 8192.
EXPLRSV1	8	Signed 2-byte integer	Reserved.
EXPLRC1	A	Signed 2-byte integer	Not used.
EXPLRC2	C	Signed 4-byte integer	Not used.

Table 64. Exit parameter list for connection routines and sign-on routines (continued)

Name	Hex offset	Data type	Description
EXPLARC	10	Signed 4-byte integer	Access return code. Values can be: <b>0</b> Access allowed; DB2 continues processing. <b>12</b> Access denied; DB2 terminates processing with an error.
EXPLSSNM	14	Character, 8 bytes	DB2 subsystem name, left justified; for example, 'DSN '.
EXPLCONN	1C	Character, 8 bytes	Connection name for requesting location.
EXPLTYPE	24	Character, 8 bytes	Connection type for requesting location. For DDF threads, the connection type is 'DIST '.
EXPLSITE	2C	Character, 16 bytes	For SNA protocols, this is the location name of the requesting location or <i>&lt;luname&gt;</i> . For TCP/IP protocols, this is the dotted decimal IP address of the requester. If the value of EXPLSITE_OFF is not 0, EXPLSITE is not used.
EXPLLUNM	3C	Character, 8 bytes	For SNA protocols, this is the locally known LU name of the requesting location. For TCP/IP protocols, this is the character string 'TCPIP'.
EXPLNTID	44	Character, 17 bytes	For SNA protocols, the fully qualified network name of the requesting location. For TCP/IP protocols, this field is reserved.
EXPLVIDS	55	Character, 1 byte	DB2 version identifier
EXPLSITE_OFF	56	Signed 2-byte integer	Offset from the beginning of the work area to the extended location name of the DB2 site that originated the work request. Use this value if the location name is greater than 16 bytes. The extended location name has the following format: <ul style="list-style-type: none"> <li>• Signed, 2-byte integer: Length of the extended location name</li> <li>• Character, 128 bytes: Extended location name</li> </ul>



### Related concepts

“Connection routines and sign-on routines” on page 229

### Related tasks

“Using sample connection and sign-on exit routines for CICS transactions” on page 159

“Specifying connection and sign-on routines” on page 229

“Debugging connection and sign-on routines” on page 238

### Related reference

“Processing of sign-on requests” on page 157

## Authorization ID parameter list for connection and sign-on routines

An authorization ID list contains information that is specific to connection routines and sign-on routines.

The following table includes the authorization ID list for a connection or sign-on exit routine.



Table 65. Authorization ID list for a connection or sign-on exit routine

Name	Hex offset	Data type	Description
AIDLPRIM	0	Character, 8 bytes	Primary authorization ID for input and output; see descriptions in the text.
AIDLCODE	8	Character, 2 bytes	Control block identifier.
AIDLTLEN	A	Signed 2-byte integer	Total length of control block.
AIDLEYE	C	Character, 4 bytes	Eyecatcher for block, “AIDL”.
AIDLSQL	10	Character, 8 bytes	On output, the current SQL ID.
AIDLSCNT	18	Signed 4-byte integer	Number of entries allocated to secondary authorization ID list. Always equal to 1012.
AIDLSAPM	1C	Address	For a sign-on routine only, the address of an 8-character additional authorization ID. If RACF is active, the ID is the user ID's connected group name. If the address was not provided, the field contains zero.
AIDLCKEY	20	Character, 1 byte	Storage key of the ID pointed to by AIDLSAPM. To move that ID, use the “move with key” (MVCK) instruction, specifying this key.
AIDLRV1	21	Character, 3 bytes	Reserved
AIDLRV2	24	Signed 4-byte integer	Reserved
AIDLACEE	28	Signed 4-byte integer	The address of the ACEE structure, if known; otherwise, zero

Table 65. Authorization ID list for a connection or sign-on exit routine (continued)

Name	Hex offset	Data type	Description
AIDLRACL	2C	Signed 4-byte integer	Length of data area returned by RACF, plus 4 bytes
AIDLRACR	30	26 bytes	Reserved
AIDLSEC	4A	Character, maximum x 8 bytes	List of the secondary authorization IDs, 8 bytes each

#### PSPI

## Input values for connection routines

A connection routine can have different input values.

The input values for a connection routine include the following:

- PSPI** The initial primary authorization ID for a local request can be obtained from the z/OS address space extension block (ASXB).  
 The ASXB contains at most only a seven-character value. That is always sufficient for a TSO user ID or a user ID from an z/OS JOB statement, and the ASXB is always used for those cases.  
 For CICS, IMS, or other started tasks, z/OS can also pass an eight-character ID. If an eight-character ID is available, and if its first seven characters agree with the ASXB value, then DB2 uses the eight-character ID. Otherwise it uses the ASXB value.  
 If RACF is active, the field used contains a verified RACF user ID; otherwise, it contains blanks.
- The primary ID for a remote request is the ID passed in the conversation attach request header (SNA FMH5) or in the DRDA SECCHK command.
- The SQL ID contains blanks.
- The list of secondary IDs contains blanks. **PSPI**

## Input values for sign-on routines

A sign-on routine can have different input values.

The input values for a sign-on routine are as follows:

- PSPI** The initial primary ID depends on the sign-on method.
- The SQL ID and all secondary IDs contain blanks.
- Field AIDLSAPM in the authorization ID list can contain the address of an 8-character additional authorization ID, obtained by the CICS attachment facility using the RACROUTE REQUEST=EXTRACT service with the requester's user ID. If RACF is active, this ID is the RACF-connected group name from the ACEE corresponding to the requester's user ID. Otherwise, this field contains blanks. IMS does not pass this parameter.
- Field AIDLCKEY contains the storage key of the identifier pointed to by AIDLSAPM. To move that ID, use the "move with key" (MVCK) instruction, specifying this key.

- Field AIDLACEE contains the ACEE address only for a sign-on through the CICS attachment facility and only when the CICS RCT uses AUTH=GROUP.

**PSPI**

## Expected output for connection and sign-on routines

DB2 uses the output values of the primary ID, the SQL ID, and the secondary IDs. Your routines can set these IDs to any value that is an SQL short identifier.

**PSPI**

If your identifier does not meet the 8-character criteria, the request fails. Therefore, when necessary, add blanks to the end of short identifiers to ensure that they meet the criteria.

If the values that are returned are not blank, DB2 interprets them in the following ways:

- The primary ID becomes the primary authorization ID.
- The list of secondary IDs, down to the first blank entry or to a maximum of 1012 entries, becomes the list of secondary authorization IDs. The space allocated for the secondary ID list is only large enough to contain the maximum number of authorization IDs. This number is in field AIDLSCNT.

**Important:** If you allow more than 1012 secondary authorization IDs, abends and storage overlays can occur.

- The SQL ID is checked to see if it is the same as the primary or one of the secondary IDs. If it is not, the connection or sign-on process fails. Otherwise, the validated ID becomes the current SQL ID.

If the returned value of the primary ID is blank, DB2 takes the following steps:

- In connection processing, the default ID that is defined when DB2 is installed (UNKNOWN AUTHID on panel DSNTIPP) is substituted as the primary authorization ID and the current SQL ID. The list of secondary IDs is set to blanks.
- Sign-on processing abends. No default value exists for the primary ID.

If the returned value of the SQL ID is blank, DB2 makes it equal to the value of the primary ID. If the list of secondary IDs is blank, it remains blank. No default secondary IDs exist.

Your routine must also set a return code in word 5 of the exit parameter list to allow or deny access (field EXPLARC). By those means you can deny the connection altogether. The code must have one of the values that are shown in Table 66.

Table 66. Required return code in EXPLARC

Value	Meaning
0	Access allowed; continue processing.
12	Access denied; terminate.

Any other value will cause an abend.

**PSPI**

## Processing in sample connection and sign-on routines

The sample routines that are provided by IBM can serve as models for the processing that is required in connection routines and sign-on routines.

**PSPI** **Recommendation:** Consider using the sample routines as a starting point when you write your own routines.

Both the sample connection routine (DSN3SATH) and the sample sign-on routine have similar sections for setup, constants, and storage areas. Both routines set values of the primary ID, the SQL ID, and the secondary IDs in three numbered sections.

*In the sample connection routine (DSN3SATH):* The three sections of the sample connection routine perform the following functions:

#### Section 1

Section 1 provides the same function as in the default connection routine. It determines whether the first character of the input primary ID has a value that is greater than blank (hex 40), and performs the following operations:

- If the first character is greater than hex 40, the value is not changed.
- If the first character is not greater than hex 40, the value is set according to the following rules:
  - If the request is from a TSO foreground address space, the primary ID is set to the logon ID.
  - If the request is not from a TSO foreground address space, the primary ID is set to the job user ID from the JES job control table.
  - If no primary ID is located, Section 2 is bypassed.

#### Section 2

At the beginning of Section 2, you can restore one commented-out instruction, which then truncates the primary authorization ID to 7 characters. (The instruction is identified by comments in the code.)

Section 2 next tests RACF options and makes the following changes in the list of secondary IDs, which is initially blank:

- If RACF is not active, the list remains blank.
- If the list of groups option is not active, but an ACEE exists, the connected group name is copied as the only secondary ID. The source of the ACEE is one of the following:
  - An ACEE that is passed by the caller
  - The address-space-level ACEE
  - The task-level ACEE if the connection is for batch utilities.
- If the list of groups option is active, the list of group names from the ICHPCGRP block is copied into AIDLSEC in the authorization ID list.

#### Section 3

Section 3 performs the following steps:

1. The SQL ID is set equal to the primary ID.
2. If the TSO data set name prefix is a valid primary or secondary ID, the SQL ID is replaced with the TSO data set name prefix. Otherwise, the SQL ID remains set to the primary ID.

*In the sample sign-on routine (DSN3SSGN):* The three sections of the sample sign-on routine perform the following functions:

#### Section 1


Section 1 does not change the primary ID.

## Section 2

Section 2 sets the SQL ID to the value of the primary ID.


## Section 3

Section 3 tests RACF options and makes the following changes in the list of secondary IDs, which is initially blank:


- If RACF is not active, the list remains blank.
- If the list of groups option is active, section 3 attempts to find an existing ACEE from which to copy the authorization ID list.
  - If AIDLACEE contains a valid ACEE, it is used.  
Otherwise, look for a valid ACEE chained from the TCB or from the ASXB or, if no usable ACEE exists, issue RACROUTE to have RACF build an ACEE structure for the primary ID.  
Copy the list of group names from the ACEE structure into the secondary authorization list.
  - If the exit issued RACROUTE to build an ACEE, another RACROUTE macro is issued and the structure is deleted.
- If a list of secondary authorization IDs has not been built, and AIDLSAPM is not zero, the data that is pointed to by AIDLSAPM is copied into AIDLSEC. 

## Performance considerations for connection and sign-on routines


Your sign-on exit routine is part of the critical path for transaction processing in IMS and CICS. Therefore, try to execute as quickly as possible.

 Avoid writing SVC calls like GETMAIN, FREEMAIN, and ATTACH. Also avoid I/O operations to any data set or database. To improve performance, you might be able to delete the list of groups that process in Section 3 of the sample sign-on exit routine.

The sample sign-on exit routine can issue the RACF RACROUTE macro with the default option SMC=YES. If another product issues RACROUTE with SMC=NO, a deadlock might occur.

Your routine can also enhance the performance of later authorization checking. Authorization for dynamic SQL statements is checked first for the CURRENT SQLID, then for the primary authorization ID, and then for the secondary authorization IDs. If you know that a user's privilege most often comes from a secondary authorization ID, then set the CURRENT SQLID to this secondary ID within your exit routine. 

### Related concepts

 General guidelines for writing exit routines (DB2 Administration Guide)

## Debugging connection and sign-on routines

The diagnostic aids can assist you in debugging connection exit routines and sign-on exit routines.



**PSPI** *Subsystem support identify recovery:* The identify ESTAE recovery routine, DSN3IDES, generates the following VRADATA entries. The last entry, key VRAIMO, is generated only if the abend occurred within the connection exit routine.

Table 67. VRADATA entries that are generated by DSN3IDES

VRA keyname	Key hex value	Data length	Content
VRAFP	22	8	Constant 'IDESTRAK'
VRAFP	23	24	<ul style="list-style-type: none"> <li>32-bit recovery tracking flags</li> <li>32-bit integer AGNT block unique identifier</li> <li>AGNT block address</li> <li>AIDL block address</li> <li>Initial primary authorization ID as copied from ASXBUSER</li> </ul>
VRAIMO	7C	10	<ul style="list-style-type: none"> <li>Connection exit load module load point address</li> <li>Connection exit entry point address</li> <li>Offset of failing address in the PSW from the connection exit entry point address</li> </ul>

*Subsystem support sign-on recovery:* The sign-on ESTAE recovery routine DSN3SIES generates the following VRADATA entries. The last entry, key VRAIMO, is generated only if the abend occurred within the sign-on exit routine.

Table 68. VRADATA entries that are generated by DSN3SIES

VRA keyname	Key hex value	Data length	Content
VRAFP	22	8	Constant 'SIESTRAK'
VRAFP	23	20	<ul style="list-style-type: none"> <li>Primary authorization ID (CCBUSER)</li> <li>AGNT block address</li> <li>Identify-level CCB block address</li> <li>Sign-on-level CCB block address</li> </ul>
VRAIMO	7C	10	<ul style="list-style-type: none"> <li>Sign-on exit load module load point address</li> <li>Sign-on exit entry point address</li> <li>Offset of failing address in the PSW from the sign-on exit entry point address</li> </ul>

*Diagnostics for connection exit routines and sign-on exit routines:* The connection (identify) recovery routine and the sign-on recovery routine provide diagnostics for the corresponding exit routines. The diagnostics are produced only when the abend occurs in the exit routine. The following diagnostics are available:

#### Dump title


The component failing module name is "DSN3@ATH" for a connection exit or "DSN3@SGN" for a sign-on exit.

#### z/OS and RETAIN® symptom data

SDWA symptom data fields SDWAC SCT (CSECT/) and SDWAMODN (MOD/) are set to "DSN3@ATH" or "DSN3@SGN", as appropriate.

#### Summary dump additions

The AIDL, if addressable, and the SADL, if present, are included in the

summary dump for the failing allied agent. If the failure occurred in connection or sign-on processing, the exit parameter list (EXPL) is also included. If the failure occurred in the system services address space, the entire SADL storage pool is included in the summary dump. 

#### Related concepts

“Connection routines and sign-on routines” on page 229

#### Related reference

“Processing of connection requests” on page 154


“Processing of sign-on requests” on page 157

“Sample connection and sign-on routines” on page 230

“Exit parameter list for connection and sign-on routines” on page 231

## Session variables in connection and sign-on routines

DB2 supplies default session variables. In addition, the connection exit routine and the sign-on exit routine support up to 10 more session variables. You can define these additional session variables and use them to provide information to applications by using the GETVARIABLE function.


 **The session variable structure:** The connection exit routine and the sign-on exit routine point to the session variable structure (DSNDSVS). DSNDSVS specifies the maximum number of entries in the session array, the actual number of entries in the session array, and a pointer to the session variable array. The default value for the actual number of session variables is zero.

**Defining session variables:** To define session variables, use the session variable array (DSNDSVA) to list up to 10 session variables as name and value pairs. The session variables that you establish in the connection exit routine and the sign-on exit routine are defined in the SESSION schema. The values that the exit routine supplies in the session variable array replace the previous values.

**Example:** The following session variable array lists six session variables.

Table 69. Sample session variable array

Name	Value
default_database	DATAXM
default_driver	PZN4Y7
location	Kyoto
member_of	GROUP_42
filename	report.txt
account_number	A1-X142783

The unqualified names are defined as VARCHAR(128), and the values are defined as VARCHAR(255). The exit routines must provide these values in Unicode CCSID 1208. 

---

## Access control authorization exit routine

You can provide your own access control authorization exit routine by using an exit point that DB2 provides. Alternatively, after you carefully consider several important factors, you might choose to let RACF perform DB2 authorization checking for you.



### Is the access control authorization exit routine right for you?

Using the RACF (Security Server for z/OS) to perform access control is not the best choice for every customer. Consider the following points before choosing RACF to perform access control:

- If you want the database administrators to manage security, integration with DB2 is very important. Using RACF access control provides less integration with DB2. In most of these cases, DB2 authorization provides advantages.
- If you want security administrators to manage security, integration with the security server is more important. In most of these cases, using RACF for access control provides advantages. Furthermore, if you want a security group to define authorization and a centralized security control point, RACF access control is an excellent match.

If you change from DB2 authorization to RACF access control, you must change to RACF methods for some authorization techniques, and you must understand how DB2 and RACF work together. Expect to make the following changes when you implement RACF access control:

- Plan to use RACF facilities (such as groups and patterns) more.
- Plan to use patterns instead of individual item access profiles and permissions.
- Plan to use DB2 roles, RACF groups, or both, instead of secondary authorization IDs, which are not implemented in RACF. OWNER generally must be a valid group or a DB2 role.
- Plan to use DB2 roles for BINDAGENT processing. BINDAGENT based on secondary authorizations IDs is not implemented in RACF.
- Understand how SET CURRENT SQLID works with RACF. SET CURRENT SQLID can set a qualifier, but does not change authorization.
- Know that authorizations are not dropped when objects are dropped or renamed.
- Be aware of the relationship between objects and revoked privileges. Packages are not invalidated when authorizations are revoked. Views are not dropped when authorizations are revoked.

### How the access control authorization routine works

Your routine specifies whether the authorization checking should all be done by RACF only, or by both RACF and DB2. (Also, the routine can be called and still let all checking be performed by DB2.)

When DB2 invokes the routine, it passes three possible functions to the routine:

- Initialization (DB2 startup)
- Authorization check
- Termination (DB2 shutdown)

The bulk of the work in the routine is for authorization checking. When DB2 must determine the authorization for a privilege, it invokes your routine. The routine determines the authorization for the privilege and then indicates to DB2 whether the privilege is authorized or not authorized, or whether DB2 should do its own authorization check, instead.

When you write an access control authorization routine, use the general guidelines for writing exit routines, with the following exceptions to the environment description:

- The routine executes in non-cross-memory mode during initialization and termination (XAPLFUNC of 1 or 3).
- During authorization checking, the routine can execute under a TCB or SRB in cross-memory or non-cross-memory mode.

## Bypass of the access control authorization routine

In the following situations, the access control authorization routine is not called to check authorization:

- The authorization ID that DB2 uses to determine access has installation SYSADM or installation SYSOPR authority (where installation SYSOPR authority is sufficient to authorize the request). This authorization check is made strictly within DB2. For example, if the execute privilege is being checked on a package, DB2 performs the check on the plan owner that this package is in. If the plan owner has installation SYSADM, the routine is not called.
- DB2 security has been disabled. (You can disable DB2 security by specifying NO on the USE PROTECTION field of installation panel DSNTIPP).
- Authorization has been cached from a prior check.
- In a prior invocation of the exit routine, the routine indicated that it should not be called again.
- GRANT statements.

The routine executes in the *ssnm*DBM1 address space of DB2.



### Related concepts

General guidelines for writing exit routines (DB2 Administration Guide)

“Access control external to DB2” on page 6

Overview (RACF Access Control Module Guide)

### Related reference

“Parameter list for access control authorization routines” on page 249

## Specifying the access control authorization routine

Your access control authorization routine must have a CSECT name and an entry point of DSNX@XAC. The load module name or alias name must also be DSNX@XAC. A default routine with this name and entry point exists in library *prefix*.SDSNLOAD.



To use your routine instead, place it in the *prefix*.SDSNEXIT library. Use installation job DSNTIJEX to assemble and link-edit the routine and to place it in the *prefix*.SDSNEXIT library. If you use any other library, you might need to change the STEPLIB or JOBLIB concatenations in the DB2 start-up procedures.

The source code for the default routine is in *prefix.SDSNSAMP* as *DSNXSXAC*. You can use it to write your own exit routine. To assemble it, you must use the High Level Assembler.

RACF provides a sample exit routine *DSNXRXAC*, which is shipped with DB2. It can be found in *prefix.SDSNSAMP*. 

## The default access control authorization routine

The default exit routine returns a code to the DB2 authorization module. The code indicates that a user-defined access control authorization exit routine is not available. DB2 then performs normal authorization checking and does not attempt to invoke this exit routine again.

## When access control authorization routine is taken

DB2 can take the access control authorization routine when it starts up, shuts down, or performs an authorization check on a privilege.

 The access control authorization routine is taken in the following three instances:

### At DB2 startup

This exit routine is taken when DB2 starts to allow the external authorization checking application to perform any required setup prior to authorization checking. For example, loading authorization profiles into storage is a required setup task. DB2 uses the reason code that the exit routine sets during startup to determine how to handle exception situations.

### When an authorization check is to be performed on a privilege

This exit routine is taken when DB2 accesses security tables in the catalog to check authorization on a privilege. The exit routine is taken only if none of the prior invocations have indicated that the exit routine must not be called again.

### At DB2 shutdown

This exit routine is taken when DB2 is stopping, to let the external authorization checking application perform its cleanup before DB2 stops.




## Considerations for the access control authorization routine

You need to take additional factors into consideration when you use the access control authorization exit routine.


### When DB2 cannot provide an ACEE

Sometimes DB2 cannot provide an ACEE. This happens, for example, when you do not use external security in CICS and CICS does not pass an ACEE to the CICS attachment facility.

 When DB2 does not have an ACEE, it passes zeros in the *XAPLACEE* field. If this happens, your routine can return a 4 in the *EXPLRC1* field, and let DB2 handle the authorization check.

DB2 does not pass the ACEE address for IMS transactions. The ACEE address is passed for CICS transactions, if available.

DB2 does pass the ACEE address when it is available for DB2 commands that are issued from a logged on z/OS console. DB2 does not pass the ACEE address for DB2 commands that are issued from a console that is not logged on, or for the START DB2 command, or commands issued automatically during DB2 startup.

An ACEE is available to DB2 for an IMS transaction if IMS is configured to use either APPC/OTMA security full or the IMS Build Security Environment exit (DFSBSEX0). You need to code DFSBSEX0 to return RC4 in register 15, which will instruct IMS to create the ACEE in the dependent region. 

### Authorization IDs and ACEEs

XAPL has two authorization ID fields, XAPLUPRM (the primary authorization ID) and XAPLUCHK (the authorization ID that DB2 uses to perform the authorization). These two fields might have different values.

 The ACEE passed in XAPLACEE is that of the primary authorization ID, XAPLUPRM.

The implications of the XAPLUPRM and XAPLUCHK relationship need to be clearly understood. XAPLUCHK, the authorization ID that DB2 uses to perform authorization may be the primary authorization ID (XAPLUPRM), a secondary authorization ID, or another authorization ID such as a package owner.

If the RACF access control module is used, the following rules apply:


- RACF uses the ACEE of the primary authorization ID (XAPLUPRM) to perform authorization.
- Secondary authorization IDs are not implemented in RACF. DB2 roles or RACF groups should be used instead.

**Examples:** The following examples show how the rules apply:

- A package may be bound successfully by using the privileges of the binder (XAPLUPRM). Then only the EXECUTE privilege on the package is needed to execute it. If at some point this package is marked invalid (for instance, if a table it depends upon is dropped and recreated), the next execution of it will cause an AUTOBIND, which will usually fail. In this case, AUTOBIND checks the runner for the necessary authorization, but the runner does not have the required privileges for a successful rebind. However, if the owner of the package is a DB2 role, and the role has the necessary authorization, AUTOBIND will succeed.
- If the OWNER on the BIND command is based on secondary authorization IDs, which are not supported by RACF. RACF groups should be used instead.
- SET CURRENT SQLID can set a qualifier, but it cannot change authorization.
- The DYNAMICRULES settings have a limited effect on which authorization ID is checked. Only the primary authorization ID and secondary IDs that are valid RACF groups for this user are considered. For dynamic statements with the DYNAMICRULES(BIND) option to work, for example, the package owner must be the primary authorization ID or one of the RACF groups of the user who executes the statements.


However, the DYNAMICRULES settings will have the desired effect on which authorization is checked if the authorization is based on a DB2 role. For

example, the dynamic statements with the DYNAMICRULES(BIND) option will work if a DB2 role is the owner of a plan or package or the definer of a stored procedure.


- User-defined function and stored procedure authorizations involve several authorization IDs, such as implementer, definer, invoker, and so forth. Only the primary authorization ID and secondary IDs that are DB2 roles or RACF groups are considered. 

### Invalid and inoperative packages

In DB2, when a privilege that is required by a package is revoked, the package is invalidated. DB2 can automatically rebind an invalidated package if proper privileges are granted.



 However, if you use an authorization access control routine, it cannot tell DB2 that a privilege is revoked. Therefore, DB2 cannot know to invalidate the package.

If the revoked privilege is the EXECUTE privilege on a user-defined function, DB2 marks the package inoperative, instead of invalid; you will need to manually rebind the inoperative package.

If a privilege that the package depends on is revoked, and if you want to invalidate the package or make it inoperative, you must use the SQL GRANT statement to grant the revoked privilege and then use the SQL REVOKE statement to revoke it. 



### Automatic rebind with DB2 roles

If you execute a package that is marked invalid, DB2 will attempt to rebind it.

 If the package contains static SQL statements, DB2 will check the owner for the required authorization for a successful rebind. If RACF access control is used and if the owner of the plan or package is a DB2 role, DB2 will be able to complete the rebind. 

### DB2 roles for the DYNAMICRULES(BIND) Option

The DYNAMICRULES(BIND) option provides the flexibility for you to specify the owner of a plan or a package that DB2 checks for the required authorization for dynamic SQL statements. Because RACF does not support secondary IDs, you can use DB2 roles to exploit this flexibility.

 To use DB2 roles with the DYNAMICRULES(BIND) option, the owner of the plan or package must be a DB2 role. Similarly, for the define and invoke behavior of the DYNAMICRULES BIND options, the definer or invoker must be a DB2 role. In order to make the owner of the plan, package, or stored procedure a DB2 role, you need to create the plan, package, or stored procedure in a trusted context that is defined with the ROLE AS OBJECT OWNER AND QUALIFIER clause. 

### Using DB2 roles for BINDAGENT

You can bind plans and packages on the behalf of the owner by using the RACF BINDAGENT privilege through a DB2 role.



**PSPI** RACF provides support for BINDAGENT through DB2 roles. To use BINDAGENT, you must specify a role, instead of a secondary ID, as the owner of a plan or package and perform the BIND task within a trusted context. Suppose you want role ROLEOWNER to own package COLLECTION01.PACKAGE01, but will have role ROLEBINDAGENT perform the BIND on behalf of role ROLEOWNER.

To have ROLEBINDAGENT perform the BIND on behalf of ROLEOWNER:

1. Create role ROLEOWNER and role ROLEBINDAGENT. Make sure that the ROLEOWNER role is the owner of the package and that the binder is associated with the ROLEBINDAGENT role and will bind the package.
2. Create trusted context CTX1 with the WITH ROLE AS OBJECT OWNER AND QUALIFIER clause. Specify ROLEBINDAGENT as the default role and set JOB=BINDPKG (which is the bind job name) and SYSTEM AUTHID=UBINDER (which is the binder's userid).
3. Create a RACF profile V91A.ROLEOWNER.BINDAGENT to control BINDAGENT access

4. Permit role ROLEBINDAGENT access to profile V91A.ROLEOWNER.BINDAGENT by issuing a RACF **PERMIT** command:

```
PERMIT V91A.ROLEOWNER.BINDAGENT ID(*) +
      WHEN(CRITERIA(SQLROLE('ROLEBINDAGENT')))) CL(MDSNSM)
```

5. Set up appropriate RACF profiles and give role ROLEOWNER the BINDADD and CREATE IN privileges on the package collection:

```
PERMIT V91A.BINDADD ID(*) CL(MDSNTB) +
      WHEN(CRITERIA(SQLROLE('ROLEOWNER'))))

PERMIT V91A.COLLECTION01.CREATEIN ID(*) CL(MDSNTB) +
      WHEN(CRITERIA(SQLROLE('ROLEOWNER'))))
```

6. Permit role ROLEOWNER all the required privileges for executing SQL statements in the application as shown in the following example:
7. Have UBINDER submit bind job BINDPKG that will run in trusted context CTX1 with role ROLEBINDAGENT and perform the BIND on behalf of role ROLEOWNER:

```
BIND PACKAGE(COLLECTION01) MEMBER(PACKAGE01) ACTION(REP) OWNER(ROLEOWNER)

RACF performs the BINDAGENT check on binder UBINDER, its role
ROLEBINDAGENT, and its RACF groups. It then perform all the remaining
checks on role ROLEOWNER and allows the BIND command to complete.
```

**PSPI**


## View authorization

DB2 passes specific base table information to an access control authorization exit (ACAE) routine. This information helps the routine to effectively control data access through views.

**PSPI** For the DELETE and INSERT privileges, DB2 passes the schema and name of the base table in the XAPLBSCM and XAPLBSNAM fields, along with the information about the view itself. For the UPDATE privilege, DB2 additionally passes the name of the base table column in the XAPLBCOL field that is being updated.



For any view in a nested stack, DB2 passes the base table information in addition to that of the view itself. All the intermediate views between the base table and the view that is being processed are ignored.

In the cases when the view is not updatable, the view information will be repeated in the XAPLBSCM, XAPLBSNAM, and XAPLBCOL fields. For example, if the view is specified with the Instead of Trigger, the base table of the view is not being updated using the view because all processing is based on the content of the trigger package. The view information is repeated in the base table fields to facilitate any view authorization check.

When a view is created, DB2 checks whether the owner of the view has the INSERT, UPDATE and DELETE privileges on the underlying base table. DB2 performs this check to determine what privileges should be granted to the view owner. This processing occurs whether or not an ACAE routine, like the RACF access control module, is in effect. If an ACAE routine is in effect, the result of the DB2 authorization check does not impact the creation of the view or the privileges that the view owner gets on the view. In the case when the view is created based on another view, the base view information will be repeated in the XAPLBSCM, XAPLBSNAM, and XAPLBCOL fields. 



### Behavior of EXPLAIN STMTCACHE with the access control authorization routine

The behavior of EXPLAIN STMTCACHE changes because in some instances the primary authorization ID replaces the statement authorization ID.

 Dynamic statements are cached by using the primary authorization ID that runs the plan or package regardless of the DYNAMICRULES value. Therefore, if the access control authorization routine is used for security, the EXPLAIN STMTCACHE statement must be issued with the same primary authorization ID as that for inserting the dynamic statements into the cache. 


### Dropping views


A view is dropped when the privilege that is required to create it is revoked.

 Revoking the privilege on a view is not communicated to DB2 by the authorization checking routine. If you want DB2 to drop the view when the privilege is revoked, you must issue the SQL REVOKE statement. 

### Caching of EXECUTE on plans, packages, and routines


The results of authorization checks on the EXECUTE privilege for plans are not cached when those checks are performed by the authorization access control exit routine. The results of authorization checks on the EXECUTE privilege for packages and routines are cached if package and routine authorization caching is enabled on your system.


 If authorization checks on the EXECUTE privilege for packages and routines are performed by the authorization access control exit routine, the role in effect or the primary authorization ID is cached. DB2 authorization can cache roles or primary authorization IDs for handling packages and routines. DB2 checks and caches a role if it is in effect and authorized. If a role is not in effect or authorized, DB2 checks and caches the primary authorization ID.

If this privilege is revoked in the exit routine, the cached information is not updated to reflect the revoke. You must use the GRANT statement and the REVOKE statement to update the cached information. 

## Caching of dynamic SQL statements



Dynamic statements can be cached when they have passed the authorization checks if the dynamic statement caching is enabled on your system.

 If authorization checks for dynamic statements are performed by the authorization access control exit routine, the role in effect or the primary authorization ID is cached. DB2 authorization can cache roles or primary authorization IDs for handling dynamic statements. DB2 checks and caches a role if it is in effect and authorized. If a role is not in effect or authorized, DB2 checks and caches the primary authorization ID.

If the privileges that this statement requires are revoked from the authorization ID that is cached with the statement, this cached statement must be invalidated. If the privilege is revoked in the exit routine this does not happen, and you must use the SQL statements GRANT and REVOKE to refresh the cache. 


## Resolution of user-defined functions


The create timestamp for a user-defined function must be older than the bind timestamp for the package or plan in which the user-defined function is invoked. If DB2 authorization checking is in effect, and DB2 performs an automatic rebind on a plan or package that invokes a user-defined function, any user-defined functions that are created after the original BIND or REBIND of the invoking plan or package are not candidates for execution.

 If you use an access control authorization exit routine, some user-defined functions that were not candidates for execution before the original BIND or REBIND of the invoking plan or package might become candidates for execution during the automatic rebind of the invoking plan or package. If a user-defined function is invoked during an automatic rebind, and that user-defined function is invoked from a trigger body and receives a transition table, the form of the invoked function that DB2 uses for function selection includes only the columns of the transition table that existed at the time of the original BIND or REBIND of the package or plan for the invoking program. 

## Creating materialized query tables

When a materialized query table is created, a CRTVUAUTT authorization check is performed. The CRTVUAUTT check is used to determine whether the creator of a materialized query table can provide the required SELECT privileges on base tables to the owner of the materialized query table.

 If the owner of the materialized query table has the required privileges, the CRTVUAUTT authorization check proves redundant. However, the check is performed before the owner of the materialized query table's privileges are determined. Therefore, if the materialized query table owner holds the necessary privileges and the creator of the materialized query table does not, the CRTVUAUTT check can produce unwanted error messages.

For an ACA exit routine to suppress unwanted error messages during the creation of materialized query tables, XAPLFSUP is turned on. 

## Parameter list for access control authorization routines

The parameter list of access control authorization routines contains pointers to other information, such as the work area and the authorization ID list.

**PSPI** The following diagram shows how the parameter list points to other information.

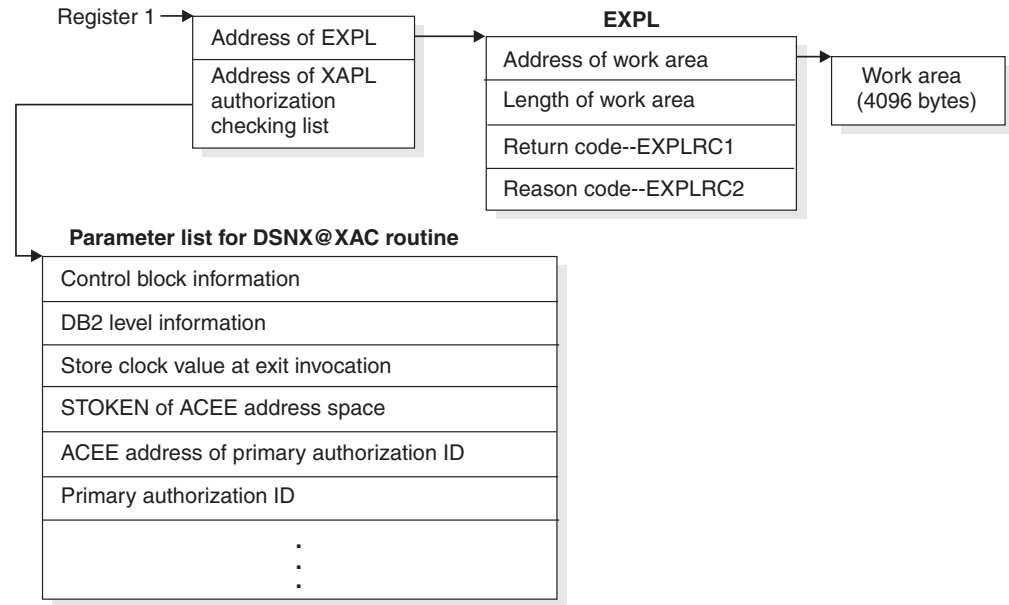


Figure 14. How an authorization routine's parameter list points to other information

The work area (4096 bytes) is obtained once during the startup of DB2 and only released when DB2 is shut down. The work area is shared by all invocations to the exit routine.

At invocation, registers are set, and the authorization checking routine uses the standard exit parameter list (EXPL). The following is a list of the exit-specific parameters, described by macro DSNDXAPL. Field names indicated by an asterisk (\*) apply to initialization, termination, and authorization checking. Other fields apply to authorization checking only.

Table 70. Parameter list for access control authorization routines

Name	Hex offset	Data type	Input or output	Description
XAPLCBID*	0	Character, 2-bytes	Input	Control block identifier; value X'216A'.
XAPLLEN *	2	Signed, 2-byte integer	Input	Length of XAPL; value X'100' (decimal 256).
XAPLEYE *	4	Character, 4 bytes	Input	Control block eye catcher; value "XAPL".
XAPLLVL *	8	Character, 8 bytes	Input	DB2 version and level; for example, "VxRxMx ".
XAPLSTCK *	10	Character, 8 bytes	Input	The store clock value when the exit is invoked. Use this to correlate information to this specific invocation.

Table 70. Parameter list for access control authorization routines (continued)

Name	Hex offset	Data type	Input or output	Description
XAPLSTKN *	18	Character, 8 bytes	Input	STOKEN of the address space in which XAPLACEE resides. Binary zeroes indicate that XAPLACEE is in the home address space.
XAPLACEE *	20	Address, 4 bytes	Input	ACEE address: <ul style="list-style-type: none"> <li>Of the DB2 address space (<i>ssnmDBM1</i>) when XAPLFUNC is 1 or 3.</li> <li>Of the primary authorization ID associated with this agent when XAPLFUNC is 2.</li> </ul> There may be cases where an ACEE address is not available for an agent. In such cases this field contains binary zeroes.
XAPLUPRM *	24	Character, 8 bytes	Input	One of the following IDs: <ul style="list-style-type: none"> <li>When XAPLFUNC is 1 or 3, it contains the User ID of the DB2 address space (<i>ssnmDBM1</i>)</li> <li>When XAPLFUNC is 2, it contains the primary authorization ID associated with the agent</li> </ul>
XAPLFUNC *	2C	Signed, 2-byte integer	Input	Function to be performed by exit routine: <div> <div>1</div> <div>Initialization</div> </div> <div> <div>2</div> <div>Authorization Check</div> </div> <div> <div>3</div> <div>Termination</div> </div>
XAPLGPAT *	2E	Character, 4 bytes	Input	DB2 group attachment name for data sharing. The DB2 subsystem name if not data sharing.
XAPLUCKT	32	Character, 1 byte	Input	Type of the authorization ID on which DB2 performs the check: <div> <div>' '</div> <div>An authorization ID</div> </div> <div> <div>L</div> <div>A role</div> </div>
XAPLONRT	33	Character, 1 byte	Input	Type of the authorization ID that owns the object in XAPLOWNR: <div> <div>' '</div> <div>An authorization ID</div> </div> <div> <div>L</div> <div>A role</div> </div>
XAPLSDEF	34	Character, 1 byte	Input	System-defined object: <div> <div>S</div> <div>A system-defined routine or package</div> </div> <div> <div>' '</div> <div>Not a system-defined object</div> </div>
XAPLRSV1	35	Character, 3 bytes		Reserved
XAPLPRIV	38	Signed, 2-byte integer	Input	DB2 privilege being checked. Security administrator (SECADM) authority and secure object creation (CREATE_SECURE_OBJECT) privilege required for row and column access control

Table 70. Parameter list for access control authorization routines (continued)

Name	Hex offset	Data type	Input or output	Description
XAPLTYPE	3A	Character, 1	Input	DB2 object type:
				<b>B</b> Buffer pool
				<b>C</b> Collection
				<b>D</b> Database
				<b>E</b> Distinct typeDistinct type
				<b>F</b> User-defined functionUser-defined function
				<b>J</b> JAR
				<b>K</b> Package
				<b>L</b> Role
				<b>M</b> Schema
				<b>N</b> Trusted context
				<b>O</b> Stored procedure
				<b>P</b> Application plan
				<b>Q</b> Sequence
				<b>R</b> Table space
				<b>S</b> Storage group
				<b>T</b> Table
				<b>U</b> System privilege
				<b>V</b> View

Table 70. Parameter list for access control authorization routines (continued)

Name	Hex offset	Data type	Input or output	Description
XAPLFLG1	3B	Character, 1	Input	<p>The highest-order bit, bit 8, (XAPLCHKS) is on if the secondary IDs associated with this authorization ID (XAPLUCHK) are included in DB2's authorization check. If it is off, only this authorization ID is checked.</p> <p>Bit 7 (XAPLUTB) is on if this is a table or view privilege (SELECT, INSERT, and so on) and if SYSCTRL, SQLADM, System DBADM, ACCESSCTRL, DATAACCESS, or SECADM is not sufficient authority to perform the specified operation on a table or view. SYSCTRL, SQLADM, System DBADM, ACCESSCTRL, DATAACCESS, or SECADM does not have the privilege of accessing user data unless the privilege is specifically granted to it.</p> <p>Bit 6 (XAPLAUTO) is on if this is an AUTOBIND.</p> <p>Bit 5 (XAPLCRVW) is on if the installation parameter DBADM CREATE AUTH is set to YES.</p> <p>Bit 4 (XAPLRDWR) is on if the privilege is a write privilege. If the privilege is a read-only privilege, bit 4 is off.</p> <p>Bit 3 (XAPLFSUP) is on to suppress error messages from the CRTVUAUTT authorization check during the creation of a materialized query table. These error messages are caused by intermediate checks that do not affect the final result.</p> <p>Bit 2 (XAPLRAOO) is on if this operation is in a trusted context that is defined with the ROLE AS OBJECT OWNER clause.</p> <p>Bit 1 (XAPLIMPD) is on if authorization checking involves an implicitly created database.</p>
XAPLUCHK	3C	Address, 4 bytes	Input	<p>Address to the authorization ID on which DB2 performs the check. It could be the primary, secondary, or some other ID. This is a VARCHAR(128) field.</p>



Table 70. Parameter list for access control authorization routines (continued)

Name	Hex offset	Data type	Input or output	Description
XAPLOBJN	40	Address, 4 bytes	Input	<p>Address to the unqualified name of the object with which the privilege is associated. This is a VARCHAR(128) field. It is one of the following names:</p> <p><b>Name</b>    <b>Length</b></p> <p><b>Application plan</b> 8</p> <p><b>Buffer pool</b> 8</p> <p><b>Collection</b> VARCHAR(128)</p> <p><b>Database</b> 8</p> <p><b>Distinct type</b> VARCHAR(128)</p> <p><b>JAR</b>    VARCHAR(128)</p> <p><b>Package</b> VARCHAR(128)</p> <p><b>Role</b>    VARCHAR(128)</p> <p><b>Schema</b> VARCHAR(128)</p> <p><b>Sequence</b> VARCHAR(128)</p> <p><b>Storage group</b> VARCHAR(128)</p> <p><b>Table</b>    VARCHAR(128)</p> <p><b>Table space</b> 8</p> <p><b>Trusted context</b> VARCHAR(128)</p> <p><b>User-defined function</b> VARCHAR(128)</p> <p><b>View</b>    VARCHAR(128)</p> <p>For special system privileges (SYSADM, SYSCTRL, and so on) this field might contain binary zeroes.</p>
XAPLOWNQ	44	Address, 4 bytes	Input	<p>Address of the object owner (creator) or object qualifier. The contents of this parameter depends on either the privilege being checked or the object. This is a VARCHAR(128) field.</p> <p>If this field is not applicable, it contains binary zeros.</p>
XAPLREL1	48	Address, 4 bytes	Input	<p>Address of other related information 1. The contents of this parameter depend on either the privilege being checked or the object. This is a VARCHAR(128) field.</p> <p>If this field is not applicable, it contains binary zeros.</p>

Table 70. Parameter list for access control authorization routines (continued)

Name	Hex offset	Data type	Input or output	Description						
XAPLREL2	4C	Address, 4 bytes	Input	Address of other related information 2. The contents of this parameter depends on the privilege being checked. This is a VARCHAR(128) field.  If this field is not applicable, it contains binary zeros.						
XAPLDBSP	50	Address, 4 bytes	Input	Address of database information. This information is passed for CREATE VIEW and CREATE ALIAS.  If this field is not applicable, it contains binary zeros.						
XAPLOWNR	54	Address, 4 bytes	Input	Address of the object owner. This is a VARCHAR(128) field.  If this field is not applicable, it contains binary zeros.						
XAPLROLE	58	Address, 4 bytes	Input	Address of the user's role when operating in a trusted context. If this field is not applicable, it contains binary zeros.						
XAPLOONM	5C	Address, 4 byte	Input	Address of other object name						
XAPLOOON	60	Address, 4 byte	Input	Address of other object owner						
XAPLBSCM	64	Address, 4 byte	Input	Address of base table qualifier of a view or repeated view qualifier						
XAPLBNAM	68	Address, 4 byte	Input	Address of base table name of a view or repeated view name						
XAPLBCOL	6C	Address, 4 byte	Input	Address of base table column name of a view or repeated view column name						
XAPLRV2	70	Character, 49 bytes		Reserved.						
XAPLOOTP	A1	Character, 1 byte	Input	Other object type or the owner of the base table of a view						
XAPLOOOT	A2	Character, 1 byte	Input	Other object owner type or the owner type of the base table of a view						
XAPLRV3	A3	Character, 1 byte		Reserved						
XAPLXBTS	A4	Timestamp, 10 bytes	Input	The function resolution timestamp. Authorizations received prior to this timestamp are valid.  Applicable to functions and procedures.						
XAPLONWT	AE	Character, 1 byte	Output	Information required by DB2 from the exit routine for the UPDATE and REFERENCES table privileges:  <table><thead><tr><th>Value</th><th>Explanation</th></tr></thead><tbody><tr><td>' '</td><td>Requester has privilege on the entire table</td></tr><tr><td>*</td><td>Requester has privilege on just this column</td></tr></tbody></table>	Value	Explanation	' '	Requester has privilege on the entire table	*	Requester has privilege on just this column
Value	Explanation									
' '	Requester has privilege on the entire table									
*	Requester has privilege on just this column									

Table 70. Parameter list for access control authorization routines (continued)

Name	Hex offset	Data type	Input or output	Description
XAPLFLG2	AF	Character, 1 byte	Input	Bit 8 (the highest-order bit) is on if an object is associated with the row and column access control (XAPLSOBJ)
				Bit 7 is on if the SEPARATE SECURITY system parameter is set to YES (XAPLSPSC)
				Bit 6 is on when a catalog table (XAPLSCTB) can be accessed only by the SECADM authority.
				Bit 5 (XAPLACAC) is on when authorization checking is done for statements that involve the package authorization, routine authorization, or dynamic statement cache.
				The remaining 4 bits are reserved
XAPLDIAG	B0	Character, 80 bytes	Output	Information returned by the exit routine to help diagnose problems.

The following table includes database information for determining authorization for creating a view. The address to this parameter list is in XAPLREL2.

Table 71. Parameter list for access control authorization routines—database information

Name	Hex offset	Data type	Input or output	Description
XAPLDBNP	0	Address	Input	Address of information for the next database. X'00000000' indicates no next database exists.
XAPLDBNM	4	Character, 8 bytes	Input	Database name.

Table 71. Parameter list for access control authorization routines—database information (continued)

Name	Hex offset	Data type	Input or output	Description
XAPLDBDA	C	Character, 1 byte	Output	<p>Required by DB2 from the exit routine for CREATE VIEW.</p> <p>A value of Y and EXPLRC1=0 indicate that the user ID in field XAPLUCHK has database administrator authority on the database in field XAPLDBNM.</p> <p>When the exit checks if XAPLUCHK can create a view for another authorization ID, it first checks for SYSADM or SYSCTRL authority. If the check is successful, no more checking is necessary because SYSCTRL authority (for non-user tables) or SYSADM authority satisfies the requirement that the view owner has the SELECT privilege for all tables and views that the view might be based on. This is indicated by a blank value and EXPLRC1=0.</p> <p>If the authorization ID does not have SYSADM or SYSCTRL authority, the exit checks if the view creator has DBADM on each database of the tables that the view is based on because the DBADM authority on the database of the base table satisfies the requirement that the view owner has the SELECT privilege for all base tables in that database.</p>
XAPLDBIM	D	Character, 1 bytes	Input	A value of 'Y' indicates that the database is implicitly created.
XAPLRV5	E	Character, 2 bytes	none	Reserved.

XAPLOWNQ, XAPLREL1 and XAPLREL2 might further qualify the object or may provide additional information that can be used in determining authorization for certain privileges. The following is a list of the privileges and the contents of XAPLOWNQ, XAPLREL1 and XAPLREL2.

Table 72. Related information for certain privileges

Privilege	Object type (XAPLTYPE)	XAPLOWNQ	XAPLREL1	XAPLREL2	XAPLOWNR
0263 (USAGE)	E	Address of schema name	Address of distinct type owner	Contains binary zeroes	Address of distinct type owner
0064 (EXECUTE) 0265 (START) 0266 (STOP) 0267 (DISPLAY)	F	Address of schema name	Address of user-defined function owner	Contains binary zeroes	Address of user-defined function owner
0263 (USAGE)	J	Address of schema name	Address of JAR owner	Contains binary zeroes	Address of JAR owner
0064 (EXECUTE)	K	Address of collection ID	Contains binary zeroes	Contains binary zeroes	Contains binary zeroes

Table 72. Related information for certain privileges (continued)

Privilege	Object type (XAPLTYPE)	XAPLOWNQ	XAPLREL1	XAPLREL2	XAPLOWNR
0065 (BIND)	K	Address of collection ID	Address of package owner	Contains binary zeroes	Address of package owner
0073 (DROP)	K	Address of collection ID	Contains binary zeroes	Address of version ID	Contains binary zeroes
0097 (COMMENT)	K	Address of collection ID	Address of package owner	Contains binary zeroes	Address of package owner
0225 (COPY ON PKG)	K	Address of collection ID	Address of package owner	Contains binary zeroes	Address of package owner
0228 (ALLPKAUT)	K	Address of collection ID	Contains binary zeroes	Contains binary zeroes	Contains binary zeroes
0229 (SUBPKAUT)	K	Address of collection ID	Contains binary zeroes	Contains binary zeroes	Contains binary zeroes
0252 (ALTERIN) 0097 (COMMENT) 0252 (DROPIN)	M	Address of schema name	Address of object owner	Contains binary zeroes	Address of object owner
0064 (EXECUTE) 0265 (START) 0266 (STOP) 0267 (DISPLAY)	O	Address of schema name	Address of procedure owner	Contains binary zeroes	Address of procedure owner
0065 (BIND)	P	Address of plan owner	Contains binary zeroes	Contains binary zeroes	Address of plan owner
0097 (COMMENT)	P	Address of plan owner	Contains binary zeroes	Contains binary zeroes	Address of plan owner
0061 (ALTER) 0263 (USAGE)	Q	Address of schema name	Address of sequence name	Contains binary zeroes	Contains binary zeroes
0061 (ALTER)	R	Address of database name	Contains binary zeroes	Contains binary zeroes	Contains binary zeroes
0073 (DROP)	R	Address of database name	Contains binary zeroes	Contains binary zeroes	Contains binary zeroes
0087 (USE)	R	Address of database name	Contains binary zeroes	Contains binary zeroes	Contains binary zeroes
0053 (UPDATE)   0054 (REFERENCES)	T	Address of table schema	Address of column name, if applicable	Address of database name	Address of table owner

Table 72. Related information for certain privileges (continued)

Privilege	Object type (XAPLTYPE)	XAPLOWNQ	XAPLREL1	XAPLREL2	XAPLOWNR
0022 (CATMAINT CONVERT) 0050 (SELECT) 0051 (INSERT) 0052 (DELETE) 0055 (TRIGGER) 0056 (CREATE INDEX) 0061 (ALTER) 0073 (DROP) 0075 (LOAD) 0076 (CHANGE NAME QUALIFIER) 0097 (COMMENT) 0098 (LOCK) 0233 (ANY TABLE PRIVILEGE) 0251 (RENAME) 0275 (REFRESH)	T	Address of table schema	Contains binary zeroes	Address of database name	Address of table owner
0020 (DROP ALIAS) 0104 (DROP SYNONYM)	T	Address of table schema	Contains binary zeroes	Contains binary zeroes	Contains binary zeroes
0103 (ALTER INDEX) 0105 (DROP INDEX) 0274 (COMMENT ON INDEX) 0283 (RENAME INDEX)	T	Address of table schema	Contains binary zeroes	Address of database name	Address of index owner
0227 (BIND AGENT)	U	Address of package owner	Contains binary zeroes	Contains binary zeroes	Address of package owner
0015 (CREATE ALIAS)	U	Contains binary zeroes	Contains binary zeroes	Address of database name, if the alias is on a table	Contains binary zeroes
0053 (UPDATE)	V	Address of view schema	Address of column name, if applicable	Address of the database name of the view's base table, if applicable	Address of view owner
0051 (INSERT) 0052 (DELETE)	V	Address of view schema	Contains binary zeroes	Address of the database name of the view's base table, if applicable	Address of view owner
0050 (SELECT) 0073 (DROP) 0097 (COMMENT) 0233 (ANY TABLE PRIVILEGE)	V	Address of view schema	Contains binary zeroes	Contains binary zeroes	Address of view owner
0055 (TRIGGER)	V	Address of view schema	Contains binary zeroes	Contains binary zeroes	Address of view owner
0061 (ALTER)	V	Address of view schema	Contains binary zeroes	Contains binary zeroes	Address of view owner

The following is a list of data types and field lengths.

Table 73. Data types and field lengths

Resource name or other	Type	Length
Database name	Character	8
Table name qualifier	Character	VARCHAR(128)
Object name qualifier	Character	VARCHAR(128)
Column name	Character	VARCHAR(128)
Collection ID	Character	VARCHAR(128)
Plan owner	Character	VARCHAR(128)
Package owner	Character	VARCHAR(128)
Package version ID	Character	VARCHAR(64)
Schema name	Character	VARCHAR(128)
Distinct typeowner	Character	VARCHAR(128)
JAR owner	Character	VARCHAR(128)
User-defined function owner	Character	VARCHAR(128)
Procedure owner	Character	VARCHAR(128)
View name qualifier	Character	VARCHAR(128)
Sequence owner	Character	VARCHAR(128)
Sequence name	Character	VARCHAR(128)



## Expected output for access control authorization routines

Your authorization exit routine is expected to return certain fields when it is called. If an unexpected value is returned in any of these fields, an abend occurs.



The following is a list of output fields for the access control authorization routine. Register 3 points to the field in error, and abend code 00E70009 is issued.

Table 74. Output fields for the access control authorization routine

Field	Required or optional
EXPLRC1	Required
EXPLRC2	Optional
XAPLONWT	Required only for UPDATE and REFERENCES table privileges
XAPLDIAG	Optional



## Handling return codes

You need to place the return codes from the access control authorization routine in the EXPL field named EXPLRC1. The EXPLRC1 value affects DB2 processing.



EXPLRC1 must have one of the following values during initialization.



Table 75. Required values in EXPLRC1 during initialization

Value	Meaning
0	Initialization successful.
12	Unable to service request; don't call exit again.

DB2 does not check EXPLRC1 on return from the exit routine during termination. Make sure that EXPLRC1 has one of the following values during the authorization check.

Table 76. Required values in EXPLRC1 during authorization check

Value	Meaning
0	Access permitted.
4	Unable to determine; perform DB2 authorization checking.
8	Access denied.
12	Unable to service request; don't call exit routine again.

On authorization failures, the return code is included in the IFCID 0140 trace record.



#### Related concepts



General guidelines for writing exit routines (DB2 Administration Guide)

#### Related reference

"Exception processing" on page 261

### Handling reason codes

After initialization, the access control authorization routine returns reason code EXPLRC2. EXPLRC2 determines how DB2 processes return code EXPLRC1 that is returned during authorization checking.



The following is a list of reason codes during initialization.

Table 77. Reason codes during initialization

Value	Meaning
-1	Identifies the default exit routine shipped with DB2. If you replace or modify the default exit, you should not use this value.
16	Indicates to DB2 that it should terminate if the exit routine returns EXPLRC1=12, an invalid EXPLRC1 or abnormally terminates during initialization or authorization checking. <b>When the exit routine sets the reason code to 16, DB2 does an immediate shutdown, without waiting for tasks to end.</b> For long-running tasks, an immediate shutdown can mean that recovery times are long.
Other	Ignored by DB2.

Field EXPLRC2 enables you to put in any code for authorization check. You can use EXPLRC2 to determine why the authorization check in the exit routine failed. On authorization failures, the reason code is included in the IFCID 0140 trace



## Related concepts

 General guidelines for writing exit routines (DB2 Administration Guide)

## Related reference

“Exception processing”

## Exception processing

During initialization or authorization checking, DB2 issues diagnostic message DSNX210I to the operator's console when an error condition occur.

 DB2 issues diagnostic message DSNX210I if one of the following conditions occur:

- The authorization exit returns a return code of 12 or an invalid return code.
- The authorization exit abnormally terminates.

Additional actions that DB2 performs depend on the reason code that the exit returns during initialization. The following is a list of these actions.

*Table 78. How an error condition affects DB2 actions during initialization and authorization checking*

Exit result	Reason code of 16 returned by exit routine during initialization	Reason code other than 16 or -1 returned by exit routine during initialization <sup>1</sup>
Return code 12	<ul style="list-style-type: none"><li>• The task<sup>2</sup> abnormally terminates with reason code 00E70015</li><li>• DB2 terminates</li></ul>	<ul style="list-style-type: none"><li>• The task<sup>2</sup> abnormally terminates with reason code 00E70009</li><li>• DB2 switches to DB2 authorization checking</li></ul>
Invalid return code	<ul style="list-style-type: none"><li>• The task<sup>2</sup> abnormally terminates with reason code 00E70015</li><li>• DB2 terminates</li></ul>	<ul style="list-style-type: none"><li>• The task<sup>2</sup> abnormally terminates with reason code 00E70009</li><li>• DB2 switches to DB2 authorization checking</li></ul>
Abnormal termination during initialization	DB2 terminates	DB2 switches to DB2 authorization checking
Abnormal termination during authorization checking	<p>You can use the subsystem parameter AEXITLIM<sup>3</sup> to control how DB2 and the exit behave.</p> <p><b>Example:</b> If you set AEXITLIM to 10, the exit routine continues to run after the first 10 abnormal terminations. On the eleventh abnormal termination, the exit stops and DB2 terminates.</p>	<p>You can use the subsystem parameter AEXITLIM to control how DB2 and the exit behave.</p> <p><b>Example:</b> If you set AEXITLIM to 10, the exit routine continues to run after the first 10 abnormal terminations. On the eleventh abnormal termination, the exit routine stops and DB2 switches to DB2 authorization checking.</p>

### Note:

1. During initialization, DB2 sets a value of -1 to identify the default exit. The user exit routine should not set the reason code to -1.
2. During initialization, the task is DB2 startup. During authorization checking, the task is the application.
3. AEXITLIM (authorization exit limit) can be updated online.

## Debugging access control authorization routines

You can use IFCID 0314 to provide a trace record of the parameter list on return from the exit routine. You can activate the trace record by turning on trace class 22.

## Determining whether the access control authorization routine is active

You can determine whether the exit routine or DB2 is performing authorization checks.

To determine whether the exit routine or DB2 is performing authorization checks:

1. Start audit trace class 1.
2. Choose a DB2 table on which to issue a SELECT statement and an authorization ID to perform the SELECT. The authorization ID must not have the DB2 SELECT privilege or the external security system SELECT privilege on the table.
3. Use the authorization ID to issue a SELECT statement on the table. The SELECT statement should fail.
4. Format the trace data and examine the return code (QW0140RC) in the IFCID 0140 trace record.
  - QW0140RC = -1 indicates that DB2 performed the authorization check and denied access.
  - QW0140RC = 8 indicates that the external security system performed the authorization check and denied access.

---

## RACF access control module

The RACF access control module allows you to use RACF as an alternative to DB2 authorization checking for DB2 objects, authorities, and utilities.

You can activate the RACF access control module at the DB2 access control authorization exit point (DSNX@XAC), where you can replace the default routine. The RACF access control module is provided as an assembler source module in the DSNRXAC member of DB2.SDSNSAMP.

The RACF access control module (DSNXRXAC) does not provide full support of role on z/OS 1.7.

### Related concepts



Overview (RACF Access Control Module Guide)

---

## Chapter 8. Protecting data through encryption and RACF

You can use the DB2 Secure Socket Layer (SSL) support or built-in data encryption functions to protect your sensitive data. You can also use the security features of RACF, or an equivalent system, to protect your data sets.

---

### Encrypting your data with Secure Socket Layer support

DB2 supports Secure Socket Layer (SSL) protocol by using the z/OS Communications Server IP Application Transparent Transport Layer Security (AT-TLS).

The z/OS Communications Server for TCP/IP (beginning in V1R7 of z/OS) supports the AT-TLS function in the TCP/IP stack for applications that require secure TCP/IP connections. AT-TLS performs TLS on behalf of the application, such as DB2, by invoking the z/OS system SSL in the TCP layer of the TCP/IP stack. The z/OS system SSL supports TLS V1.0, SSL V3.0, and SSL V2.0 protocols.

AT-TLS also uses policies that provide system SSL configurations for connections that use AT-TLS. An application continues to send and receive clear text data over the socket while the transmission is protected by the system SSL.

AT-TLS support is policy-driven and can be deployed transparently underneath many existing sockets applications.

#### Related concepts

“Encrypting your data through DB2 built-in functions” on page 272

“Protecting data sets through RACF” on page 269

### AT-TLS configuration

You need to complete a set of configurations that are required for DB2 to take advantage of AT-TLS support.

You must complete the following configurations of your DB2 to utilize the AT-TLS support:

- PROFILE.TCPIP configuration

You can specify the TTLS or NOTTLS parameter on the TCPCONFIG statement in PROFILE.TCPIP to control whether you want to use the AT-TLS support.

- TCP/IP stack access control configuration

To protect TCP/IP connections, you can configure the RACF EZB.INITSTACK.sysname.tcpname resource in the SERVAUTH class to block all stack access except for the user IDs that are permitted to use the resource.

- Policy configuration

The policy agent provides AT-TLS policy rules to the TCP/IP stack. Each rule defines a set of security conditions that the policy agent compares to the conditions at the connection that it is checking. When the policy agent finds a match, it assigns the connection to the actions that are associated with the rule.

## SSL authentication level

The Secure Socket Layer (SSL) protocol supports server and client authentication during the handshake phase.

The SSL provides server authentication as the minimum level of security. It uses the Server Authentication mechanism to secure communications between a server and its client and allows the client to validate the authenticity of the server.

The SSL provides client authentication as an additional level of authentication and access control. It enables a server to validate the certificates of a client at the server and thus prevents the client from obtaining a secure connection without an installation-approved certificate.

Client authentication is optional and, if used, can provide the following three levels of authentication:

- Level 1 authentication is performed by system SSL. A client passes a digital certificate to a server as part of the SSL handshake. To successfully pass the required authentication, the Certificate Authority (CA) that signs the client certificate must be trusted by the server. That is, the certificate for the CA must be in the key ring that the server uses and designates as trusted.
- Level 2 (addition to level 1) authentication requires that a client certificate be registered with RACF (or other SAF-compliant security products) and mapped to a valid user ID. When AT-TLS receives the client certificate during the SSL handshake, it queries RACF to verify that the certificate maps to a valid user ID before allowing a secure connection to be established. This level of client authentication provides additional access control at the server and ensures that the client is known to have a valid user ID on the server host.
- Level 3 (addition to levels 1 and 2) authentication provides the capability to restrict access to a server based on the user ID associated with a client certificate. A client can access a server only if the client itself is valid to the server, its certificate is valid, and a user ID associated with the certificate is valid. This level of authentication uses the RACF SERVAUTH general resource class to restrict access to the server based on the user ID of the client. If the SERVAUTH general resource class is not active or the SERVAUTH profile for the server is not defined, AT-TLS assumes that this level of authentication is not requested. However, if the SERVAUTH general resource class is active and the server's SERVAUTH profile is defined, a remote secure connection is established only if the user ID that is associated with the client certificate is permitted to the server's SERVAUTH profile. Otherwise, the secure connection is not established and the connection itself is dropped.

### Configuring SSL authentication levels

The Secure Socket Layer (SSL) protocol supports server and client authentication during the handshake phase. You can specify to use either server authentication, client authentication, or both depending on your security need.

To specify whether to use server authentication, client authentication, or both, use the following approaches:

- If you need only a minimum level of security to authenticate the communications between a server and its clients, consider using server authentication. To use server authentication, specify the HandshakeRole Server parameter for the TTLSEnvironmentAction statement in the AT-TLS policy, as shown in the following example:

```

TTLSEnvironmentAction DB2ServerSecureEnvAct
{
    TLSKeyRingParms
    {
        Keyring DB2ServerKeyring
    }
    HandshakeRole Server
}

```

With this configuration, AT-TLS sends the server certificate to the client during the handshake phase of a connection request. The client then validates the server by examining the server certificate that it receives.

- If you need additional security, consider using client authentication. To use client authentication:

1. Specify the HandshakeRole ServerWithClientAuth parameter for the TTLSEnvironmentAction statement in the AT-TLS policy, as shown in the following example:

```

TTLSEnvironmentAction DB2ServerSecureEnvAct
{
    TLSKeyRingParms
    {
        Keyring DB2SERVERKEYRING
    }
    HandshakeRole ServerWithClientAuth
    TTLSEnvironmentAdvancedParms
    {
        ClientAuthType SAFCheck
    }
}

```

2. Determining the level of client authentication by specifying the ClientAuthType parameter for the TTLSEnvironmentAdvancedParms statement in the AT-TLS policy.

Table 79. Client authentication levels

Client Authentication Level	ClientAuthType	Client Certificate	SERVAUTH Class Active and Server SERVAUTH Profile Defined	Certificate Validation
None	PassThru	Optional	N/A	None
None	Full	Optional	N/A	Certificate is validated against key ring, if provided
Level 1	Required (default)	Required	N/A	Certificate is validated against key ring
Level 2	SAFCheck	Required	Optional	Certificate is validated against key ring and must be associated with a user ID in the security product
Level 3	SAFCheck	Required	Required	Certificate is validated against key ring and must be associated to a user ID in the security product and must be permitted to access server's SERVAUTH profile

Depending on the authentication type (ClientAuthType) you specify, AT-TLS may not require the client to present its certificate during the SSL handshake phase.

3. Register the client Certificate Authority (CA) certificate to RACF as trusted by issuing the RACDCERT ADD command, as shown in the following example:

```
RACDCERT ID(USRT001) ADD('USRT001.CLIENT.CRT') TRUST
```

This registers the client CA certificate in dataset 'USRT001.CLIENT.CRT' to the RACF database. The certificate is owned by RACF-defined user USRT001. The client CA certificate is also marked as trusted so that RACF can use it to verify the client certificate when it is presented to the system.

4. Add the client CA certificate to a key ring and map it to a user ID by issuing the RACDCERT CONNECT command, as shown in the following example:

```
RACDCERT ID(SYSDSP) CONNECT(ID(USRT001) LABEL('LABEL00000001'))  
RING(DB2SERVERKEYRING) USAGE(PERSONAL))
```

This adds the client CA certificate to server key ring DB2SERVERKEYRING. The certificate is owned by user USRT001.

DB2 is now ready to accept secure connections from remote clients that use SSL client and server authentication.

## Creating and activating client certificate name filters

A *certificate name filter* enables you to associate many client certificates with one user ID based on the unique user information in the certificate, such as the user's affiliation. You can create one or more certificate name filters to map a large number of client certificates to a limited number of user IDs, which helps you reduce administrative costs.

To create and activate a certificate name filter:

1. Create a certificate name filter by issuing the **RACDCERT MAP** command, as shown in the following example:

```
RACDCERT MAP ID(USRT001) -  
SDNFILTER('O=IBM.L=San Jose.SP=CA.C=US')  
WITHLABEL('IBMers') TRUST
```

This creates a new certificate name filter based on the subject's distinguished name in the certificate. The filter associates user ID USRT001 to any user presenting a certificate with subject name 'O=IBM, L=San Jose, ST=CA, C=US'.

2. Activate the SETROPTS RACLIST processing for the DIGTNMAP class.

Using the **RACDCERT MAP** command to create a certificate name filter automatically generates a mapping profile in the DIGTNMAP class that represents the new filter. Both the DIGTNMAP class and the SETROPTS RACLIST processing for the DIGTNMAP class must be active before you can complete the creation of the new certificate name filter. Issue the following command to activate the SETROPTS RACLIST processing for the DIGTNMAP class:

```
SETROPTS CLASSACT(DIGTNMAP) RACLIST(DIGTNMAP)
```

3. Refresh the DIGTNMAP class.

Once SETROPTS RACLIST processing for the DIGTNMAP class is active, you must refresh the DIGTNMAP class for the certificate name filter to take effect. Issue the following command to refresh the DIGTNMAP class:

```
SETROPTS RACLIST(DIGTNMAP) REFRESH
```

4. Register a client CA certificate to use with the certificate name filter.

During the SSL handshake phase of establishing a secure connection, AT-TLS retrieves certificate information from RACF if client authentication is specified. In order for AT-TLS to retrieve the client CA certificate and private keys from RACF, the client CA certificate must be connected to the server key ring. You can use the new certificate name filter to register the client CA certificate to



RACF, connect to the server key ring, and map to the user ID CERTAUTH as trusted by issuing the following command:

```
RACDCERT CERTAUTH ADD('USRT001.CLIENT.CRT') TRUST
RACDCERT ID(SYSDSP) CONNECT(CERTAUTH LABEL('LABEL000000001'))
RING(DB2SERVERKEYRING) USAGE(CERTAUTH)
```

This registers the client CA certificate to RACF and maps it to the ID CERTAUTH as TRUST. It then adds the certificate to key ring DB2ASERVERKEYRING (owned by ID SYSDSP) and indicates it is used for certificate authority purposes. As a result, when a remote client establishes a secure connection with DB2, AT-TLS is able to authenticate the client from the client CA certificate in RACF. Because the certificate name filter is active, user ID USRT001 is returned by AT-TLS to DB2.

## Configuring the DB2 server for SSL

To implement SSL support for a DB2 server, you need to make sure that the TCP/IP SQL Listener service task of DDF is capable of listening to a secondary secure port for inbound SSL connections.

The TCP/IP Listener accepts regular (non-SSL) connections on the DRDA port, whereas the secure port accepts only SSL connections to provide secure communications with a partner. Clients are assured of getting the SSL protocol connections that they require.

The secure port is used only for accepting secure connections that use the SSL protocol. When the TCP/IP Listener accepts an inbound connection on the secure port, DDF invokes the SIOCTLSSLCTL IOCTL service with TTLSi\_Req\_Type set to TTLS\_QUERY\_ONLY. It also retrieves the following AT-TLS policy information:

- Status of the connection. The status of a connection is either SECURE or NOT SECURE.
- Policy status of the connection. The IOCTL returns one of the following policy status:
  - If the IOCTL returns a policy status of TTLS\_POL\_NO\_POLICY, a matching policy rule is not found for the connection and subsequently the connection status is not secure.
  - If the IOCTL returns a policy status of TTLS\_POL\_NOT\_ENABLED, a matching policy rule is found for the connection but the policy is not configured to allow a secure connection for that client.
  - If the IOCTL returns a policy status of TTLS\_POL\_ENABLED, a matching policy rule is found, and SSL is enabled for the connection.
- Security type for the connection. The security type is either server or server with client authentication (with ClientAuthType = SAFCheck)
- RACF-defined user ID that is associated with a client certificate. If a client certificate is provided by the client and validated by AT-TLS and if a user ID is mapped to the certificate, the user ID is returned. Otherwise, the user ID is not returned.

If a secure port is not properly configured, DDF rejects the inbound connection request on the secure port. You must change the client system to either use the non-secure port, or you can configure the secure port to access DB2 remotely.

To specify a secure port to DB2, use one of the following approaches:

- Specify the TCP/IP port number in the DRDA SECURE PORT field of the Distributed Data Facility Panel 2 (DSNTIP5) during DB2 installation.



The DRDA SECURE PORT field specifies the port number that is to be used for accepting TCP/IP connection requests from remote DRDA clients that want to establish a secure connection using the SSL protocol. The value of the port number is a decimal number between 1 and 65534, and it cannot have the same value as the values of the DRDA PORT and RESYNC PORT fields. Any non-zero port numbers are verified to ensure that they are all unique port numbers. If an error is detected, installation is not allowed to proceed until you resolve the error condition. If the DRDA SECURE PORT field is blank, SSL verification support is disabled, and the DB2 TCP/IP SQL Listener does not accept any inbound SSL connections on the secure port.

- Update the SECPORT parameter of the DDF statement in the BSDS with the change log inventory (DSNJU003) stand-alone utility.

The SECPORT parameter specifies the port number for the DB2 TCP/IP SQL Listener to accept inbound SSL connections. The value of the port number is a decimal number between 0 to 65535, and it cannot have the same value as the values of the PORT and RESPORT parameters. If the value of SECPORT secure port is the same as the value of PORT or RESPORT, DB2 issues an error message. If you specify a value of 0 for the SECPORT parameter, SSL verification support is disabled, and the DB2 TCP/IP SQL Listener does not accept any inbound SSL connections on the secure port.

If the value of SECPORT is disabled, the client can still use the DRDA PORT and use SSL on it, but DB2 does not validate whether the connection uses SSL protocol.

**Data sharing considerations:** For a data sharing environment, each DB2 member with SSL support must specify a secure port. The secure port for each DB2 member of the group should be the same, just as the DRDA PORT for each member should also be the same. If each DB2 member specifies a unique secure port, unpredictable behaviors might occur. For example, Sysplex member workload balancing might not work correctly.

Similarly, for DB2 members that are defined as a subset of the data sharing group, each DB2 member that belongs to the subset needs to configure the secure port. You do not need to define a separate unique secure port for the location alias.

## Configuring the DB2 requester for SSL

A DB2 requester must be able to insist on an SSL-protected connection to certain servers. To ensure SSL-protected connections, you can make communications database (CDB) changes that indicate that SSL-protected connections are required to certain remote locations.

If a secure connection is required, DDF must determine whether an AT-TLS policy rule is defined and whether AT-TLS is enabled for the connection. To obtain this AT-TLS information, DDF invokes SIOCTTLCTL IOCTL with TTLSi\_Req\_Type = TTLS\_QUERY\_ONLY. If the IOCTL returns a policy status of TTLS\_POL\_NO\_POLICY, a matching policy rule is not found for the connection.

If the IOCTL returns a policy status of TTLS\_POL\_NOT\_ENABLED, a policy rule is defined for the connection, but AT-TLS is not enabled, and a secure connection is not established with the remote server. DDF issues a message, and the connection is closed.

If the IOCTL returns a policy status of TTLS\_POL\_ENABLED, a matching policy rule is found, and SSL is enabled for the connection.

To specify a secure connection to DB2, use one of the following approaches:

- Specify 'Y' for the SECURE column in the SYSIBM.LOCATIONS table.
- Specify a desired value for the PORT column in the SYSIBM.LOCATIONS table for SSL connections.

For SSL support, the PORT column must contain the value of the configured secure DRDA port at the remote server. However, if the value of the PORT column is blank and the value of the SECURE column is 'Y', DB2 uses the reserved secure DRDA port (448) as the default.

Some DB2 applications might require SSL protection and accept the performance cost for this level of security. However, some applications might be satisfied with unprotected connections. This flexibility can be accomplished by the use of the LOCATION ALIAS name feature.

Consider a DB2 server that is configured to support both non-secure and secure connections. At the DB2 requester, you can define two rows in the SYSIBM.LOCATIONS table: one row that specifies the location name and the non-secure DRDA port of the server and another row that specifies a different location name and the secure DRDA port of the server and SECURE='Y'. At the DB2 server, you can define a LOCATION ALIAS name to provide alternative names for any DB2 requesters that need to access the server by using the SSL protocol.

---

## Protecting data sets through RACF

To fully protect the data in DB2, you must take steps to ensure that no other process has access to the data sets in which DB2 data resides.

Use RACF, or a similar external security system, to control access to the data sets just as RACF controls access to the DB2 subsystem. This section explains how to create RACF profiles for data sets and allow their use through DB2.

Assume that the RACF groups DB2 and DB2USER, and the RACF user ID DB2OWNER, have been set up for DB2 IDs. Given that setting, the examples that follow show you how to:

- Add RACF groups to control data sets that use the default DB2 qualifiers.
- Create generic profiles for different types of DB2 data sets and permit their use by DB2 started tasks.
- Permit use of the profiles by specific IDs.
- Allow certain IDs to create data sets.

### Related concepts

“Encrypting your data through DB2 built-in functions” on page 272

“Encrypting your data with Secure Socket Layer support” on page 263

## Adding groups to control DB2 data sets

The default high-level qualifier for data sets that contain DB2 databases and recovery logs is DSNCL00. The default high-level qualifier for distribution, target, SMP, and other installation data sets is DSNAL00.

The DB2OWNER user ID can create groups that control those data sets by issuing the following commands:

```
ADDGROUP DSNCL00 SUPGROUP(DB2) OWNER(DB2OWNER)
ADDGROUP DSNAL00 SUPGROUP(DB2) OWNER(DB2OWNER)
```

## Creating generic profiles for data sets

DB2 uses specific names to identify data sets for special purposes.

Suppose that SYSDSP is the RACF user ID for DB2 started tasks in the following examples. DB2OWNER can issue the following commands to create generic profiles for the data sets and give complete control over the data sets to DB2 started tasks:

- For active logs, issue the following commands:  

```
ADDSD 'DSNC100.LOGCOPY*' UACC(NONE)
PERMIT 'DSNC100.LOGCOPY*' ID(SYSDSP) ACCESS(ALTER)
```
- For archive logs, issue the following commands:  

```
ADDSD 'DSNC100.ARCHLOG*' UACC(NONE)
PERMIT 'DSNC100.ARCHLOG*' ID(SYSDSP) ACCESS(ALTER)
```
- For bootstrap data sets, issue the following commands:  

```
ADDSD 'DSNC100.BSDS*' UACC(NONE)
PERMIT 'DSNC100.BSDS*' ID(SYSDSP) ACCESS(ALTER)
```
- For table spaces and index spaces, issue the following commands:  

```
ADDSD 'DSNC100.DSNUBC.*' UACC(NONE)
PERMIT 'DSNC100.DSNUBC.*' ID(SYSDSP) ACCESS(ALTER)
```
- For installation libraries, issue the following command:  

```
ADDSD 'DSNA10.*' UACC(READ)
```

Started tasks do not need control.
- For other general data sets, issue the following commands:  

```
ADDSD 'DSNC100.*' UACC(NONE)
PERMIT 'DSNC100.*' ID(SYSDSP) ACCESS(ALTER)
```

Although all of those commands are not absolutely necessary, the sample shows how you can create generic profiles for different types of data sets. Some parameters, such as universal access, could vary among the types. In the example, installation data sets (DSNA10.\*) are universally available for read access.

If you use generic profiles, specify NO on installation panel DSNTIPP for ARCHIVE LOG RACE, or you might get a z/OS error when DB2 tries to create the archive log data set. If you specify YES, DB2 asks RACF to create a separate profile for each archive log that is created, which means that you cannot use generic profiles for these data sets.

To protect VSAM data sets, use the cluster name. You do not need to protect the data component names, because the cluster name is used for RACF checking.

The VSAM resource that is used to store the administrative scheduler task list must be protected in RACF against unauthorized access. Only the administrative scheduler started task user has the UPDATE authority on VSAM resources.

**Access by stand-alone DB2 utilities:** The following DB2 utilities access objects that are outside of DB2 control:

- DSN1COPY and DSN1PRNT: table space and index space data sets
- DSN1LOGP: active logs, archive logs, and bootstrap data sets
- DSN1CHKR: DB2 directory and catalog table spaces
- Change Log Inventory (DSNJU003) and Print Log Map (DSNJU004): bootstrap data sets

The Change Log Inventory and Print Log Map utilities run as batch jobs that are protected by the USER and PASSWORD options on the JOB statement. To provide a value for the USER option, for example SVCAID, issue the following commands:

- For DSN1COPY:  
PERMIT 'DSNC100.\*' ID(SVCAID) ACCESS(CONTROL)
- For DSN1PRNT:  
PERMIT 'DSNC100.\*' ID(SVCAID) ACCESS(READ)
- For DSN1LOGP:  
PERMIT 'DSNC100.LOGCOPY\*' ID(SVCAID) ACCESS(READ)  
PERMIT 'DSNC100.ARCHLOG\*' ID(SVCAID) ACCESS(READ)  
PERMIT 'DSNC100.BSDS\*' ID(SVCAID) ACCESS(READ)
- For DSN1CHKR:  
PERMIT 'DSNC100.DSNDBDC.\*' ID(SVCAID) ACCESS(READ)
- For Change Log Inventory:  
PERMIT 'DSNC100.BSDS\*' ID(SVCAID) ACCESS(CONTROL)
- For Print Log Map:  
PERMIT 'DSNC100.BSDS\*' ID(SVCAID) ACCESS(READ)

The level of access depends on the intended use, not on the type of data set (VSAM KSDS, VSAM linear, or sequential). For update operations, ACCESS(CONTROL) is required; for read-only operations, ACCESS(READ) is sufficient.

You can use RACF to permit programs, rather than user IDs, to access objects. When you use RACF in this manner, IDs that are not authorized to access the log data sets might be able to do so by running the DSN1LOGP utility. Permit access to database data sets through DSN1PRNT or DSN1COPY.

## Authorizing DB2 IDs to use data set profiles

Authorization IDs with the installation SYSADM or installation SYSOPR authority need access to most DB2 data sets.

The following command adds the two default IDs that have the SYSADM and SYSOPR authorities if no other IDs are named when DB2 is installed:

```
ADDUSER (SYSADM SYSOPR)
```

The next two commands connect those IDs to the groups that control data sets, with the authority to create new RACF database profiles. The ID that has the installation SYSOPR authority (SYSOPR) does not need that authority for the installation data sets.

```
CONNECT (SYSADM SYSOPR) GROUP(DSNC100) AUTHORITY(CREATE) UACC(NONE)
CONNECT (SYSADM) GROUP(DSNA10) AUTHORITY(CREATE) UACC(NONE)
```

The following set of commands gives the IDs complete control over DSNC100 data sets. The system administrator IDs also have complete control over the installation libraries. Additionally, you can give the system programmer IDs the same control.

```
PERMIT 'DSNC100.LOGCOPY*' ID(SYSADM SYSOPR) ACCESS(ALTER)
PERMIT 'DSNC100.ARCHLOG*' ID(SYSADM SYSOPR) ACCESS(ALTER)
PERMIT 'DSNC100.BSDS*' ID(SYSADM SYSOPR) ACCESS(ALTER)
PERMIT 'DSNC100.DSNDBDC.*' ID(SYSADM SYSOPR) ACCESS(ALTER)
PERMIT 'DSNC100.*' ID(SYSADM SYSOPR) ACCESS(ALTER)
PERMIT 'DSNA10.*' ID(SYSADM) ACCESS(ALTER)
```

## Enabling DB2 IDs to create data sets

You can enable DB2 IDs to create data sets by connecting them to the DSNC100 group that has the CREATE authority.

You can issue the following command to connect several IDs to the DSNC100 group:

```
CONNECT (USER1 USER2 USER3 USER4 USER5)  
        GROUP(DSNC100) AUTHORITY(CREATE) UACC(NONE)
```

Those IDs can now explicitly create data sets whose names have DSNC100 as the high-level qualifier. Any such data sets that are created by DB2 or by these RACF user IDs are protected by RACF. Other RACF user IDs are prevented by RACF from creating such data sets.

If no option is supplied for PASSWORD on the ADDUSER command that adds those IDs, the first password for the new IDs is the name of the default group, DB2USER. The first time that the IDs sign on, they all use that password, but they must change the password during their first session.

---

## Encrypting your data through DB2 built-in functions

DB2 provides built-in data encryption and decryption functions that you can use to encrypt sensitive data, such as credit card numbers and medical record numbers.

You can encrypt data at the column or value level. You must install the Integrated Cryptographic Service Facility to use the built-in functions for data encryption.

When you use data encryption, DB2 requires the correct password to retrieve the data in a decrypted format. If an incorrect password is provided, DB2 does not decrypt the data.

The ENCRYPT keyword encrypts data. The DECRYPT\_BIT, DECRYPT\_CHAR, and DECRYPT\_DB keywords decrypt data. These functions work like other built-in functions. To use these functions on data, the column that holds the data must be properly defined.

Built-in encryption functions work for data that is stored within DB2 subsystem and is retrieved from within that same DB2 subsystem. The encryption functions do not work for data that is passed into and out of a DB2 subsystem. This task is handled by DRDA data encryption, and it is separate from built-in data encryption functions.

**Attention:** DB2 cannot decrypt data without the encryption password, and DB2 does not store encryption passwords in an accessible format. If you forget the encryption password, you cannot decrypt the data, and the data might become unusable.

### Related concepts

“Encrypting your data with Secure Socket Layer support” on page 263

“Protecting data sets through RACF” on page 269

## Defining columns for encrypted data

When data is encrypted, it is stored as a binary data string. Therefore, encrypted data should be stored in columns that are defined as VARCHAR FOR BIT DATA.

Columns that hold encrypted data also require additional bytes to hold a header and to reach a multiple of 8 bytes.

Suppose that you have non-encrypted data in a column that is defined as VARCHAR(6). Use the following calculation to determine the column definition for storing the data in encrypted format:

Maximum length of non-encrypted data	6 bytes
Number of bytes to the next multiple of 8	2 bytes
24 bytes for encryption key	24 bytes
	-----
Encrypted data column length	32 bytes

Therefore, define the column for encrypted data as VARCHAR(32) FOR BIT DATA.

If you use a password hint, DB2 requires an additional 32 bytes to store the hint. Suppose that you have non-encrypted data in a column that is defined as VARCHAR(10). Use the following calculation to determine the column definition for storing the data in encrypted format with a password hint:

Maximum length of non-encrypted data	10 bytes
Number of bytes to the next multiple of 8	6 bytes
24 bytes for encryption key	24 bytes
32 bytes for password hint	32 bytes
	-----
Encrypted data column length	72 bytes

Therefore, define the column for encrypted data as VARCHAR(72) FOR BIT DATA.

#### Related tasks

“Defining column-level encryption”

“Defining value-level encryption” on page 275

“Optimizing performance of encrypted data” on page 277

## Defining column-level encryption

For column-level encryption, all encrypted values in a column are encrypted with the same password.

To define column-level encryption:

1. Create the EMP table with the EMPNO column. The EMPNO column must be defined with the VARCHAR data type, must be defined FOR BIT DATA, and must be long enough to hold the encrypted data. The following statement creates the EMP table:

**GUI**

```
CREATE TABLE EMP (EMPNO VARCHAR(32) FOR BIT DATA);
```

**GUI**

2. Set the encryption password. The following statement sets the encryption password to the host variable :hv\_pass:

**GUI**

```
SET ENCRYPTION PASSWORD = :hv_pass;
```

**GUI**

3. Use the ENCRYPT keyword to insert encrypted data into the EMP table by issuing the following statements:

**GUI**

```
INSERT INTO EMP (EMPNO) VALUES(ENCRYPT('47138'));
INSERT INTO EMP (EMPNO) VALUES(ENCRYPT('99514'));
INSERT INTO EMP (EMPNO) VALUES(ENCRYPT('67391'));
```

**GUI**

4. Select the employee ID numbers in decrypted format:

**GUI**

```
SELECT DECRYPT_CHAR(EMPNO) FROM EMP;
```

**GUI**

If you provide the correct password, DB2 returns the employee ID numbers in decrypted format.

**Related tasks**

“Defining columns for encrypted data” on page 272

“Defining value-level encryption” on page 275

“Optimizing performance of encrypted data” on page 277

**Creating views with column-level encryption**

You can create a view that uses column-level encryption and selects decrypted data from a table.

You can define the view with a decryption function in the defining fullselect. If the correct password is provided when the view is queried, DB2 will return decrypted data. Suppose that you want to create a view that contains decrypted employee ID numbers from the EMP table.

To create a view that uses column-level encryption and selects decrypted data:

1. Create a view on the EMP table by using the following statement:

**GUI**

```
CREATE VIEW CLR_EMP (EMPNO) AS SELECT DECRYPT_CHAR(EMPNO) FROM EMP;
```

**GUI**

2. Set the encryption password so that the fullselect in the view definition can retrieve decrypted data.

Use the following statement:

**GUI**

```
SET ENCRYPTION PASSWORD = :hv_pass;
```

**GUI**

3. Select the desired data from the view by using the following statement:

**GUI**

```
SELECT EMPNO FROM CLR_EMP;
```

**GUI**



## Using password hints with column-level encryption

DB2 can store encryption password hints to help with forgotten encryption passwords. Each password hint uses 32 bytes in the encrypted column.

For column-level encryption, the password hint is set with the SET ENCRYPTION PASSWORD statement. The GETHINT function returns the password hint.

**Example:** Use the following statement to set the password hint to the host variable hv\_hint:

**GUIP**

```
SET ENCRYPTION PASSWORD = :hv_pass WITH HINT = :hv_hint;
```

**GUIP**

**Example:** Suppose that the EMPNO column in the EMP table contains encrypted data and that you submitted a password hint when you inserted the data. Suppose that you cannot remember the encryption password for the data. Use the following statement to return the password hint:

**GUIP**

```
SELECT GETHINT (EMPNO) FROM EMP;
```

**GUIP**

## Defining value-level encryption

When you use value-level encryption, each value in a given column can be encrypted with a different password. You set the password for each value by using the ENCRYPT keyword with the password.

The following keywords are used with value-level encryption:

### ENCRYPT

Indicates which data requires encryption. Also, encryption passwords, and optionally password hints, are indicated as part of the ENCRYPT keyword for value-level encryption.

**Recommendation:** Use host variables instead of literal values for all passwords and password hints. If statements contain literal values for passwords and password hints, the security of the encrypted data can be compromised in the DB2 catalog and in a trace report.

### DECRYPT\_BIT, DECRYPT\_CHAR, DECRYPT\_DB

Checks for the correct password and decrypts data when the data is selected.

**Example:** Suppose that a web application collects user information about a customer. This information includes the customer name, which is stored in host variable custname; the credit card number, which is stored in a host variable cardnum; and the password for the card number value, which is stored in a host variable userpswd. The application uses the following statement to insert the customer information:

**GUIP**



```
INSERT INTO CUSTOMER (CCN, NAME)
VALUES(ENCRYPT(:cardnum, :userpswd), :custname);
```

#### GUIP

Before the application displays the credit card number for a customer, the customer must enter the password. The application retrieves the credit card number by using the following statement:

#### GUIP

```
SELECT DECRYPT_CHAR(CCN, :userpswd) FROM CUSTOMER WHERE NAME = :custname;
```

#### GUIP

### Related tasks

“Defining columns for encrypted data” on page 272

“Defining column-level encryption” on page 273

“Optimizing performance of encrypted data” on page 277

### Using password hints with value-level encryption

DB2 can store encryption password hints to help with forgotten encryption passwords. Each password hint uses 32 bytes in the encrypted column.

For value-level encryption, the password hint is set with the ENCRYPT keyword. The GETHINT function returns the password hint.

**Recommendation:** Use host variables instead of literal values for all passwords and password hints. If the statements contain literal values for passwords and password hints, the security of the encrypted data can be compromised in the DB2 catalog and in a trace report.

**Example:** Suppose that you want the application from the previous example to use a hint to help customers remember their passwords. The application stores the hint in the host variable pswdhint. For this example, assume the values 'Tahoe' for userpswd and 'Ski Holiday' for pswdhint. The application uses the following statement to insert the customer information:

#### GUIP

```
INSERT INTO CUSTOMER (CCN, NAME)
VALUES(ENCRYPT(:cardnum, :userpswd, :pswdhint), :custname);
```

#### GUIP

If the customer requests a hint about the password, the following query is used:

#### GUIP

```
SELECT GETHINT(CCN) INTO :pswdhint FROM CUSTOMER WHERE NAME = :custname;
```

#### GUIP

The value for pswdhint is set to 'Ski Holiday' and returned to the customer. Hopefully the customer can remember the password 'Tahoe' from this hint.

## Encrypting non-character values

DB2 supports encryption for numeric and datetime data types indirectly through casting. If non-character data is cast as VARCHAR or CHAR, the data can be encrypted.

**Example:** Suppose that you need to encrypt timestamp data and retrieve it in decrypted format. Perform the following steps:

1. Create a table to store the encrypted values and set the column-level encryption password by using the following statements:

```
CREATE TABLE ETEMP (C1 VARCHAR(124) FOR BIT DATA);  
SET ENCRYPTION PASSWORD :hv_pass;
```

2. Cast, encrypt, and insert the timestamp data by using the following statement:

```
INSERT INTO ETEMP VALUES ENCRYPT(CHAR(CURRENT TIMESTAMP));
```

3. Recast, decrypt, and select the timestamp data by using the following statement:

```
SELECT TIMESTAMP(DECRYPT_CHAR(C1)) FROM ETEMP;
```

## Using predicates for encrypted data

When data is encrypted, only = and <> predicates provide accurate results. Predicates such as >, <, and LIKE return inaccurate results for encrypted data.

**Example:** Suppose that the value 1234 is encrypted as H71G. Also suppose that the value 5678 is encrypted as BF62. If you use a <> predicate to compare these two values in encrypted format, you receive the same result as you will if you compare these two values in decrypted format:

Decrypted:	1234 <> 5678	True
Encrypted:	H71G <> BF62	True

In both case, they are not equal. However, if you use a < predicate to compare these values in encrypted format, you receive a different result than you will if you compare these two values in decrypted format:

Decrypted:	1234 < 5678	True
Encrypted:	H71G < BF62	False

To ensure that predicates such as >, <, and LIKE return accurate results, you must first decrypt the data.

## Optimizing performance of encrypted data

Encryption typically degrades the performance of most SQL statements. Decryption requires extra processing, and encrypted data requires more space in DB2.

If a predicate requires decryption, the predicate is a stage 2 predicate, which can degrade performance. Encrypted data can also impact your database design, which can indirectly impact performance. To minimize performance degradation, use encryption only in cases that require encryption.

**Recommendation:** Encrypt only a few highly sensitive data elements, such credit card numbers and medical record numbers.

Some data values are poor candidates for encryption. For example, boolean values and other small value sets, such as the integers 1 through 10, are poor candidates for encryption. Because few values are possible, these types of data can be easy to guess even when they are encrypted. In most cases, encryption is not a good security option for this type of data.

**Data encryption and indexes:** Creating indexes on encrypted data can improve performance in some cases. Exact matches and joins of encrypted data (if both tables use the same encryption key to encrypt the same data) can use the indexes that you create. Because encrypted data is binary data, range checking of encrypted data requires table space scans. Range checking requires all the row values for a column to be decrypted. Therefore, range checking should be avoided, or at least tuned appropriately.

**Encryption performance scenario:** The following scenario contains a series of examples that demonstrate how to improve performance while working with encrypted data.

**Example:** Suppose that you must store EMPNO in encrypted form in the EMP table and in the EMPPROJ table. To define tables and indexes for the encrypted data, use the following statements:

**GUIP**

```
CREATE TABLE EMP (EMPNO VARCHAR(48) FOR BIT DATA, NAME VARCHAR(48));
CREATE TABLE EMPPROJ(EMPNO VARCHAR(48) FOR BIT DATA, PROJECTNAME VARCHAR(48));
CREATE INDEX IXEMPPRJ ON EMPPROJ(EMPNO);
```

**GUIP**

**Example:** Next, suppose that one employee can work on multiple projects, and that you want to insert employee and project data into the table. To set the encryption password and insert data into the tables, use the following statements:

**GUIP**

```
SET ENCRYPTION PASSWORD = :hv_pass;
SELECT INTO :hv_enc_val FROM FINAL TABLE
  (INSERT INTO EMP VALUES (ENCRYPT('A7513'),'Super Prog'));
INSERT INTO EMPPROJ VALUES (:hv_enc_val,'UDDI Project');
INSERT INTO EMPPROJ VALUES (:hv_enc_val,'DB2 10');
SELECT INTO :hv_enc_val FROM FINAL TABLE
  (INSERT INTO EMP VALUES (ENCRYPT('4NF18'),'Novice Prog'));
INSERT INTO EMPPROJ VALUES (:hv_enc_val,'UDDI Project');
```

**GUIP**

You can improve the performance of INSERT statements by avoiding unnecessary repetition of encryption processing. Note how the host variable hv\_enc\_val is defined in the SELECT INTO statement and then used in subsequent INSERT statements. If you need to insert a large number of rows that contain the same encrypted value, you might find that the repetitive encryption processing degrades performance. However, you can dramatically improve performance by encrypting the data, storing the encrypted data in a host variable, and inserting the host variable.

**Example:** Next, suppose that you want to find the programmers who are working on the UDDI Project. Consider the following pair of SELECT statements:

- **Poor performance:** The following query shows how not to write the query for good performance:

**GUIP**

```
SELECT A.NAME, DECRYPT_CHAR(A.EMPNO) FROM EMP A, EMPPROJECT B
WHERE DECRYPT_CHAR(A.EMPNO) = DECRYPT_CHAR(B.EMPNO) AND
      B.PROJECT = 'UDDI Project';
```

**GUIP** Although the preceding query returns the correct results, it decrypts every EMPNO value in the EMP table and every EMPNO value in the EMPPROJECT table where PROJECT = 'UDDI Project' to perform the join. For large tables, this unnecessary decryption is a significant performance problem.

- **Good performance:** The following query produces the same result as the preceding query, but with significantly better performance. To find the programmers who are working on the UDDI Project, use the following statement:

**GUIP**

```
SELECT A.NAME, DECRYPT_CHAR(A.EMPNO) FROM EMP A, EMPPROJ B
WHERE A.EMPNO = B.EMPNO AND B.PROJECT = 'UDDI Project';
```

**GUIP**

**Example:** Next, suppose that you want to find the projects that the programmer with employee ID A7513 is working on. Consider the following pair of SELECT statements:

- **Poor performance:** The following query requires DB2 to decrypt every EMPNO value in the EMPPROJECT table to perform the join:

**GUIP**

```
SELECT PROJECTNAME FROM EMPPROJ WHERE DECRYPT_CHAR(EMPNO) = 'A7513';
```

**GUIP**

- **Good performance:** The following query encrypts the literal value in the predicate so that DB2 can compare it to encrypted values that are stored in the EMPNO column without decrypting the whole column. To find the projects that the programmer with employee ID A7513 is working on, use the following statement :

**GUIP**

```
SELECT PROJECTNAME FROM EMPPROJ WHERE EMPNO = ENCRYPT('A7513');
```

**GUIP**

### Related tasks

“Defining columns for encrypted data” on page 272

“Defining column-level encryption” on page 273

“Defining value-level encryption” on page 275



---

## Chapter 9. Auditing access to DB2

*Security auditing* allows you to inspect and examine the adequacy and effectiveness of the policies and procedures that you put in place to secure your data.

DB2 provides the ability for you to monitor if your security plan is adequately designed based on your security objectives and determine if your implementation techniques and procedures are effectively carried out to protect your data access and consistency. It enables you to address the following fundamental questions about your data security.

- What sensitive data requires authorized access?
- Who is privileged to access the data?
- Who has actually accessed the data?
- What attempts are made to gain unauthorized access?

The DB2 catalog contains critical authorization and authentication information. This information provides the primary audit trail for the DB2 subsystem. You can retrieve the information from the catalog tables by issuing SQL queries.

Most of the catalog tables describe the DB2 objects, such as tables, views, table spaces, packages, and plans. Other tables, particularly those with the “AUTH” character string in their names, hold records of every granted privilege and authority. Each catalog record of a grant contains the following information:

- Name of the object
- Type of privilege
- IDs that receive the privilege
- IDs that grant the privilege
- Time of the grant

The DB2 audit trace can help you monitor and track all the accesses to your protected data. The audit trace records provide another important trail for the DB2 subsystem. You can use the the audit trace to record the following access information:

- Changes in authorization IDs
- Changes to the structure of data, such as dropping a table
- Changes to data values, such as updating or inserting records
- Access attempts by unauthorized IDs
- Results of GRANT statements and REVOKE statements
- Mapping of Kerberos security tickets to IDs
- Other activities that are of interest to auditors

### Related tasks

“Auditing manager access” on page 13

“Auditing payroll operations and management” on page 16

---

## Determining active security measures

If you are a security auditor, you must know the security measures that are enabled on the DB2 subsystem.

You can determine whether DB2 authorization checking, the audit trace, and data definition control are enabled in the following ways:

**Audit trace**

To see whether the trace is running, display the status of the trace by the command `DISPLAY TRACE(AUDIT)`.

**DB2 authorization checking**

Without changing anything, look at panel DSNTIPP. If the value of the `USE PROTECTION` field is YES, DB2 checks privileges and authorities before permitting any activity.

**Data definition control**

Data definition control is a security measure that provides additional constraints to existing authorization checks. With it, you control how specific plans or collections of packages can use data definition statements. To determine whether data definition control is active, look at option 1 on the DSNTIPZ installation panel.

---

## DB2 audit trace

The DB2 trace facility lets you collect monitoring, auditing, and performance information about your data and environment.

The audit trace enables you to trace different events or categories of events by authorization IDs, object ownerships, and so on. When started, the audit trace records certain types of actions and sends the report to a named destination. The trace reports can indicate who has accessed data.

As with other types of DB2 traces, you can choose the following options for the audit trace:

- Categories of events
- Particular authorization IDs or plan IDs
- Methods to start and stop the audit trace
- Destinations for audit records

You can choose whether to audit the activity on a table by specifying an option of the `CREATE` and `ALTER` statements.

**Related concepts**

“Audit trace records” on page 285

“Audit trace reports” on page 284

**Related tasks**

“Starting the audit trace” on page 286

“Stopping the audit trace” on page 286

## Authorization IDs traced by auditing

An audit traces generally identifies a process by its primary authorization ID. It records the primary ID before and after the invocation of an authorization exit routine. Therefore, you can identify the primary ID that is associated with a data change.

**Exception:** If a primary ID has been translated many times, you might not be able to identify the primary ID that is associated with a change. Suppose that the server does not recognize the translated ID from the requesting site. In this case, you cannot use the primary ID to gather all audit records for a user that accesses remote data.

The AUTHCHG record shows the values of all secondary authorization IDs that are established by an exit routine.

With the audit trace, you can also determine which primary ID is responsible for the action of a secondary ID or a current SQL ID. Suppose that the user with primary ID SMITHJ sets the current SQL ID to TESTGRP to grant privileges over the table TESTGRP.TABLE01 to another user. The DB2 catalog records the grantor of the privileges as TESTGRP. However, the audit trace shows that SMITHJ issued the grant statement.

**Recommendation:** Consider carefully the consequences of altering that ID by using an exit routine because the trace identifies a process by its primary ID. If the primary ID identifies a unique user, individual accountability is possible. However, if several users share the same primary ID, you cannot tell which user issues a particular GRANT statement or runs a particular application plan.

### Audit classes

When you start the trace, you choose the events to audit by specifying one or more audit classes.



The trace records are limited to 5000 bytes; the descriptions that contain long SQL statements might be truncated. The following table describes the available classes and the events that they include.

Table 80. Audit classes and the events that they trace

Audit class	Events that are traced
1	Access attempts that DB2 denies because of inadequate authorization. This class is the default.
2	Explicit GRANT and REVOKE statements and their results. This class does not trace implicit grants and revokes.
3	Traces CREATE, DROP, and ALTER operations against an audited table or a table that is enabled with multilevel security with row-level granularity. For example, it traces the updates to a table created with the AUDIT CHANGES or AUDIT ALL clause. It also traces the deletion of a table as the result of a DROP TABLESPACE or DROP DATABASE statement.
4	Changes to audited tables. Only the first attempt to change a table, within a unit of recovery, is recorded. (If the agent or the transaction issues more than one COMMIT statement, the number of audit records increases accordingly.) The changed data is not recorded; only the attempt to make a change is recorded. If the change is not successful and is rolled back, the audit record remains; it is not deleted. This class includes access by the LOAD utility. Accesses to a dependent table that are caused by attempted deletions from a parent table are also audited. The audit record is written even if the delete rule is RESTRICT, which prevents the deletion from the parent table. The audit record is also written when the rule is CASCADE or SET NULL, which can result in deletions that cascade to the dependent table.
5	All read accesses to tables that are identified with the AUDIT ALL clause. As in class 4, only the first access within a DB2 unit of recovery is recorded. References to a parent table are also audited.



Table 80. Audit classes and the events that they trace (continued)

Audit class	Events that are traced
6	<p>The bind of static and dynamic SQL statements of the following types:</p> <ul style="list-style-type: none"> <li>• INSERT, UPDATE, DELETE, CREATE VIEW, and LOCK TABLE statements for audited tables. Except for the values of host variables, the audit record contains the entire SQL statement.</li> <li>• SELECT statements on tables that are identified with the AUDIT ALL clause. Except for the values of host variables, the audit record contains the entire SQL statement.</li> </ul>
7	<p>Assignment or change of an authorization ID because of the following reasons:</p> <ul style="list-style-type: none"> <li>• Changes through an exit routine (default or user-written)</li> <li>• Changes through a SET CURRENT SQLID statement</li> <li>• An outbound or inbound authorization ID translation</li> <li>• An ID that is being mapped to a RACF ID from a Kerberos security ticket</li> </ul>
8	The start of a utility job, and the end of each phase of the utility
9	Various types of records that are written to IFCID 0146 by the IFI WRITE function
10	CREATE and ALTER TRUSTED CONTEXT statements, establish trusted connection information and switch user information
11	Audit the use of any administrative authority and the successful execution of any authorization ID



## Audit trace reports

If you regularly start the audit trace for all classes, you can generate audit reports based on the data that you accumulate.

Consider producing audit trace reports that focus on the following important security events:

### Use of sensitive data

You should define tables that contain sensitive data, such as employee salary records, with the AUDIT ALL option. You can report use by table and by authorization ID to look for access by unusual IDs, at unusual times, or of unexpected types. You should also record any ALTER or DROP operations that affect the data. Use audit classes 3, 4, and 5.

### Grants of critical privileges

Carefully monitor IDs with special authorities, such as SYSADM and DBADM. Also carefully monitor IDs with privileges over sensitive data, such as an update privilege on records of accounts payable. You can query the DB2 catalog to determine which IDs hold privileges and authorities at a particular time. To determine which IDs received privileges and then had them revoked, use audit class 2 and consult the audit records.

### Unsuccessful access attempts

Investigate all unsuccessful access attempts. Although some access failures are only user errors, others can be attempts to violate security. If you have sensitive data, always use trace audit class 1. You can report by table or by authorization ID.

### **Related concepts**

“DB2 audit trace” on page 282

“Audit trace records”

## **Audit trace records**

An audit trace record contains the information about the authorization ID that initiated the activity that is traced.

In addition, it contains the following information:

- The LOCATION of the ID that initiated the activity (if the access was initiated from a remote location)
- The type of activity and the time that the activity occurred
- The DB2 objects that were affected
- Whether access was denied
- The owner of a particular plan and package
- The database alias (DBALIAS) that was used to access a remote location or a location alias that was accepted from a remote application.

### **Related concepts**

“DB2 audit trace” on page 282

“Audit trace reports” on page 284

### **Related tasks**

“Collecting audit trace records” on page 287

“Formatting audit trace records” on page 287

## **Limitations of the audit trace**

The audit trace has certain limitations, including that it does not automatically record everything.

The audit trace has the following additional limitations:

- The audit trace must be turned on; it is not on by default.
- The trace does not record old data after it is changed.
- If an agent or transaction accesses a table more than once in a single unit of recovery, the audit trace records only the first access.
- The audit trace does not record accesses if you do not start the audit trace for the appropriate class of events.
- Except class 8, the audit trace does not audit certain utilities. For example, the trace audits the first access of a table with the LOAD utility, but it does not audit access by the COPY, RECOVER, and REPAIR utilities. The audit trace does not audit access by stand-alone utilities, such as DSN1CHKR and DSN1PRNT.
- The trace audits only the tables that you specifically choose to audit.
- You cannot audit access to auxiliary tables.
- You cannot audit the catalog tables because you cannot create or alter catalog tables.

This auditing coverage is consistent with the goal of providing a moderate volume of audit data with a low impact on performance. However, when you choose classes of events to audit, consider that you might ask for more data than you are willing to process.

## Starting the audit trace

You can automatically start an audit trace whenever DB2 is started.

You can do so by setting the AUDIT TRACE field on the DSNTIPN installation panel to one of the following options:

- \* (an asterisk) to provide a complete audit trace.
- NO, the default, if you do not want an audit trace to start automatically.
- YES to start a trace automatically for the default class (class 1: access denials) and the default destination (the SMF data set).
- A list of audit trace classes (for example, 1,3,5) to start a trace automatically for those classes. This option uses the default destination.

As with other types of DB2 traces, you can start an audit trace at any time by issuing the START TRACE command. You can choose the audit classes to trace and the destination for trace records. You can also include an identifying comment.

**Example:** The following command starts an audit trace for classes 4 and 6 with distributed activity:

**GUIP**

```
-START TRACE (AUDIT) CLASS (4,6) DEST (GTF) LOCATION (*)  
  COMMENT ('Trace data changes; include text of dynamic DML statements.')
```

**GUIP**

### Related concepts

“DB2 audit trace” on page 282

### Related tasks

“Stopping the audit trace”

## Stopping the audit trace

You can have multiple traces that run at the same time, including more than one audit trace. You can stop a particular trace by issuing the STOP TRACE command with the same options that you use for START TRACE.

**GUIP**

You must include enough options to uniquely identify a particular trace when you issue the command.

**Example:** The following command stops the trace that you started:

```
-STOP TRACE (AUDIT) CLASS (4,6) DEST (GTF)
```

If you did not save the START command, you can determine the trace number and stop the trace by its number. Use DISPLAY TRACE to find the number.

**Example:** DISPLAY TRACE (AUDIT) might return a message like the following output:

TNO	TYPE	CLASS	DEST	QUAL
01	AUDIT	01	SMF	NO
02	AUDIT	04,06	GTF	YES

The message indicates that two audit traces are active. Trace 1 traces events in class 1 and sends records to the SMF data set. Trace 1 can be a trace that starts automatically whenever DB2 starts. Trace 2 traces events in classes 4 and 6 and sends records to GTF.

You can stop either trace by using its identifying number (TNO).

**Example:** To stop trace 1, use the following command:

```
-STOP TRACE AUDIT TNO(1)
```



#### **Related concepts**

“DB2 audit trace” on page 282

#### **Related tasks**

“Starting the audit trace” on page 286

## **Collecting audit trace records**

You can prepare the System Management Facility (SMF) or Generalized Trace Facility (GTF) to accept audit trace records the same way as you prepare performance trace records. The records are of SMF type 102, as are performance trace records.

If you send trace records to SMF (the default), data might be lost in the following circumstances:

- SMF fails while DB2 continues to run.
- An unexpected abend (such as a TSO interrupt) occurs while DB2 is transferring records to SMF.

In those circumstances, SMF records the number of records that are lost. z/OS provides an option to stop the system rather than to lose SMF data.

#### **Related concepts**

“Audit trace records” on page 285

#### **Related tasks**

“Formatting audit trace records”

## **Formatting audit trace records**

You can extract, format, and print DB2 trace records.

You can use any of the following methods to extract, format, and print the trace records:

- DB2 Audit Management Expert for z/OS
- IBM Tivoli zSecure Audit
- IBM Tivoli OMEGAMON XE on z/OS
- Your own application program to access the SMF data
- The instrumentation facility interface (IFI) as an online resource to retrieve audit records.

### Related concepts

“Audit trace records” on page 285

### Related tasks

“Collecting audit trace records” on page 287

## Auditing in a distributed data environment

The DB2 audit trace records any access to your data, whether the request is from a remote location or from your local DB2 subsystem.

The trace record for a remote request reports the authorization ID as the final result of one of the following conditions:

- An outbound translation
- An inbound translation
- Activity of an authorization exit routine

Essentially, the ID on a trace record for a remote request is the same as the ID to which you grant access privileges for your data. Requests from your location to a remote DB2 are audited only if an audit trace is active at the remote location. The output from the trace appears only in the records at that location.

---

## DB2 audit policy

An *audit policy* is a set of criteria that determines the categories to be audited. It helps you configure and control the audit requirements of your security policies and to monitor data access by applications and individual users (authorization IDs or roles), including administrative authorities.

### GUPI

You can create an audit policy by inserting a row in the SYSIBM.SYSAUDITPOLICIES table. The SECADM, ACCESSCTRL, DATAACCESS, system DBADM, SQLADM, SYSCRTL, and SYSADM authorities all have the implicit SELECT privilege on the SYSIBM.SYSAUDITPOLICIES table. The SECADM authority also has implicit INSERT, UPDATE, and DELETE privileges on the SYSIBM.SYSAUDITPOLICIES table.

If you have the required privileges to issue the **START TRACE**, **STOP TRACE**, and **DISPLAY TRACE** commands, you can activate, deactivate, and display an audit policy by issuing those commands with the AUDTPLCY option. The SECADM authority has the implicit privileges to issue the **START TRACE**, **STOP TRACE**, and **DISPLAY TRACE**

commands. **GUPI**

## Audit category

DB2 audit policies are created and stored in the SYSIBM.SYSAUDITPOLICIES table. Each policy is specified with specific audit categories.

### PSPI

DB2 supports the following audit categories:

Table 81. DB2 audit policy categories

Category	Description
CHECKING	Generates IFCID 140 trace records for denied access attempts due to inadequate DB2 authorization and IFCID 83 trace records for RACF authentication failures

Table 81. DB2 audit policy categories (continued)

Category	Description
VALIDATE	Generates IFCID 55, 83, 87, 169, and 319 trace records for new or changed assignments of authorization IDs and IFCID 269 trace records for the establishment of trusted connections or the switch of users in existing trusted connections
OBJMAINT	<p>Generates IFCID 142 trace records when tables are altered or dropped. When an audit policy is defined, it specifies the tables to be audited. The same audit policy can be used to audit different tables in a schema by specifying the table names with the SQL LIKE predicate.</p> <p>Only tables that are defined in the following types of table spaces can be audited:</p> <ul style="list-style-type: none"> <li>• Universal table space (UTS), including UTS that contains implicitly created tables, such as XML tables</li> <li>• Traditional partitioned table space</li> <li>• Segmented table space.</li> </ul> <p>In addition to tables, an audit policy can also be used to audit clone tables and tables that are implicitly created for XML columns.</p> <p>The type of the object to be audited can be specified by using the OBJECTTYPE column. The default OBJECTTYPE column value of blank indicates that all of the supported object types are audited.</p>
EXECUTE	<p>Generates IFCID 143 and 144 trace records for SQL statement and generates IFCID 145 records to trace bind time information about SQL statements that involve audited objects.</p> <p>When an audit policy is defined, it specifies the tables to be audited. The same audit policy can be used to audit different tables in a schema by specifying the table names with the SQL LIKE predicate.</p> <p>Only tables that are defined in the following types of table spaces can be audited:</p> <ul style="list-style-type: none"> <li>• Universal table space (UTS), including UTS that contains implicitly created tables, such as XML tables</li> <li>• Traditional partitioned table space</li> <li>• Segmented table space.</li> </ul> <p>In addition to tables, an audit policy can also be used to audit clone tables and tables that are implicitly created for XML columns.</p> <p>The type of the object to be audited can be specified by using the OBJECTTYPE column. The default OBJECTTYPE column value of blank indicates that all of the supported object types are audited.</p> <p>These trace records are written when the table that is identified by the OBJECTSCHEMA, OBJECTNAME and OBJECTTYPE is accessed during the first operation by each unit of work. If the audit policy is started after the SQL query is started, access to the table will not be audited.</p>
CONTEXT	Generates IFCID 23, 24, and 25 records.
SECMAINT	Generates IFCID 141 trace records for granting and revoking privileges or administrative authorities, IFCID 270 trace records for creating and altering trusted contexts, and IFCID 271 trace records for creating, altering, and dropping row permissions or column masks.

Table 81. DB2 audit policy categories (continued)


Category	Description
SYSADMIN	<p>Generates IFCID 361 trace records when an administrative authority, in the order of installation SYSADM, installation SYSOPR, SYSOPR, SYSCTRL, or SYSADM, satisfies the required privilege for performing an operation</p> <p>If the Access Control Authorization Exit (ACAE) is active, only the operations that are performed by the installation SYSADM and installation SYSOPR authorities are audited.</p>
DBADMIN	<p>Generates IFCID 361 trace records when an administrative authority, in the order of DBMAINT, DBCTRL, DBADM, PACKADM, SQLADM, system DBADM, DATAACCESS, ACCESSCTRL, or SECADM, satisfies the required privilege for performing an operation</p> <p>The database name can be specified for auditing the DBADM, DBCTRL and DBMAINT authorities. If the database name is not specified, all the databases, including implicit databases, are audited.</p> <p>The collection ID can be specified for auditing the PACKADM authority. If the collection ID is specified, all packages in that collection are audited. If the collection ID is not specified, the packages in all collections are audited.</p> <p>If the Access Control Authorization Exit (ACAE) is active, only the operations that are performed by the SECADM authority are audited.</p>

For the SYSADMIN and DBADMIN categories, DB2 checks a set of rules for each operation to determine the required authorization. In general, the rules are checked in the order of installation SYSADM, installation SYSOPR (if applicable), specific privileges required for the operation (i.e., SELECT, UPDATE), database authorities (i.e., DBMAINT, DBCTRL, DBADM), system database authorities (i.e., SQLADM, system DBADM, DATAACCESS, and ACCESSCTRL), and system authorities (i.e., SYSCTRL, SYSADM, and SECADM).

For example, to determine whether a user can alter a table, DB2 checks the required privilege in the following order:

1. Installation SYSADM
2. ALTER table privilege
3. DBADM authority on the database that the table is in
4. System DBADM
5. SYSCTRL
6. SYSADM

If the user has only the ALTER privilege on the table and if the audit policy is activated to audit the SYSADM authority, DB2 does not generate an IFCID 361 audit record on the ALTER operation. If the user also has the SYSADM authority, DB2 still does not generate an IFCID 361 record because the lowest (ALTER) privilege permits the operation.

In general, DB2 always checks the installation SYSADM and installation SYSOPR authorities prior to the lowest (ALTER) privilege. If the user has the installation SYSADM authority and the audit policy is activated to audit the installation SYSADM authority, DB2 generates an IFCID 361 record. 



## Creating and activating audit policies

With the SECADM authority, you can create, display, activate, or inactivate DB2 audit policies.

To create and activate an audit policy:

1. Obtain the SECADM authority if you don't have it. Alternately, you can have the SECADM authority grant you the required privileges to create an audit policy. A user with the SYSOPR authority can activate the policy.
2. Create a new audit policy by issuing the INSERT statement.

You need to specify a name for the new audit policy. An audit policy name is an identifier that is 1 to 128 letters or digits in length, begins with a letter.

You also need to specify proper audit categories in the new audit policy. If you specify the OBJMAINT or EXECUTE category, you must also specify the OBJECTSCHEMA, OBJECTNAME, and OBJECTTYPE columns in the SYSIBM.SYSAUDITPOLICIES table that identify the table to be audited.

For example, if you want to create a new AUDITADMIN1 policy to audit the SYSADM authority, you can specify SYSADMIN as the category:

```
INSERT INTO SYSIBM.SYSAUDITPOLICIES(AUDITPOLICYNAME, SYSADMIN)
VALUES('AUDITADMIN1','S');
```

**GUI**

You can also use the SQL LIKE predicate to audit tables of the same characteristics. For example, you can audit all tables that start with EMP in schema TSCHEMA by issuing the following INSERT statement:

```
INSERT INTO SYSIBM.SYSAUDITPOLICIES
(AUDITPOLICYNAME, OBJECTSCHEMA, OBJECTNAME, OBJECTTYPE, EXECUTE)
VALUES('TEST2','TSCHEMA','E_P%', 'T','C');
```

**GUI**

3. Activate the audit policy by issuing the **START TRACE** command with the AUDTPLCY option.

You need to specify the AUDTPLCY option on the command to enable a specific audit policy:

```
-STA TRACE (AUDIT) DEST (GTF) AUDTPLCY(AUDITADMIN1)
```

**GUI**

This command starts IFCID 361 trace record to audit the use of the SYSADM authority. DB2 also starts an IFCID 362 trace record to trace the audit policy information as defined in the catalog. If multiple audit policies are specified to start at the same time, the IFCID 362 record is cut for every audit policy specified and contains the information about whether the policies successfully started or failed.

Depending on the categories in the audit policy, DB2 starts the associated audit trace records, one for each IFCID that is related to the specified audit category.

DB2 runs against the audit policies that are already defined in the SYSIBM.SYSAUDITPOLICIES table when you issue the **START TRACE** command; it ignores any change you make to a specific audit policy after you start the START TRACE command. If you want DB2 to run against the updated audit



policy, you need to stop and then start the audit policy trace. In addition, you cannot specify the CLASS or IFCID option when you specify the AUDTPLCY option on the **START TRACE** command.

If you prefer the audit policy to be automatically started, you need to set the DB2START column to Y or S in the SYSIBM.SYSAUDITPOLICIES table. The audit policy will be started during DB2 startup. When you specify DB2START='S', only users (authorization IDs or roles) with the SECADM authority can stop the policy. If you set DB2START='S' to an audit policy that is already started, you must stop and restart the policy for the new setting to take effect.

You can automatically start up to 8 audit policies during DB2 startup. If you specify to automatically start multiple audit policies with different DB2START column settings, DB2 will start two traces, one for policies with DB2START = 'Y' and the other for policies with DB2START = 'S'. If you need to stop any audit policy that is automatically started, you must simultaneously stop all the policies that are assigned the same trace number.

4. If necessary, display the audit policy by issuing the **DISPLAY TRACE** command. You need to specify the AUDTPLCY option on the command to show the name

and other details about the AUDITADMIN1 audit policy: **GUIP**  
-DISPLAY TRACE (AUDIT) DETAIL(2) DEST (GTF) AUDTPLCY(AUDITADMIN1)

**GUIP**

The command returns an output like the following: **GUIP**

```
15.49.46          -DIS TRACE(AUDIT) DETAIL(2)
15.49.47   STC00125  DSNW143I - CURRENT TRACE QUALIFICATIONS ARE -
15.49.47   STC00125  DSNW152I - BEGIN TNO 04 QUALIFICATIONS:
NO QUALIFICATIONS
END TNO 04 QUALIFICATIONS
15.49.47   STC00125  DSNW185I - BEGIN TNO 04 AUDIT POLICIES:
ACTIVE AUDIT POLICY: AUDITADMIN1
ACTIVE AUDIT POLICY: AUDITTABLE1
END TNO 04 AUDIT POLICIES
15.49.47   STC00125  DSNW148I - *****END OF DISPLAY TRACE QUALIFICATION
DATA*****
15.49.47   STC00125  DSNW022I - DSNWVCM1 '-DIS TRACE' NORMAL COMPLETION
```

**GUIP**

5. If necessary, disable the audit policy by issuing the **STOP TRACE** command. You need to specify the AUDTPLCY option on the command to stop all the

trace activities that are started by a specific audit policy: **GUIP**  
-STO TRACE (AUDIT) DEST (GTF) AUDTPLCY(AUDITADMIN1)

**GUIP**

Only the **STOP TRACE** command can stop all the trace activities that are started by a specific audit policy; deleting the active policy row from the SYSIBM.SYSAUDITPOLICIES table does not stop the tracing.

## Auditing the use of an administrative authority

You can create and activate an audit policy to audit how a DB2 administrative authority is used.

Suppose that you have the SECADM authority and are responsible for making sure that all security policies, including audit policies, work as designed. You want to audit the use of the SYSADM authority by user SYSADMIN1.

To audit the use of the SYSADM authority by SYSADMIN1:

1. Create audit policy AUDITADMN1 by issuing the following INSERT statement:

**GUI**

```
INSERT INTO SYSIBM.SYSAUDITPOLICIES(AUDITPOLICYNAME, SYSADMIN)
VALUES('AUDITADMN1','S');
```

**GUI**

DB2 checks to make sure that you have the required privilege to issue the INSERT statement. Upon successful verification, it inserts a row in SYSIBM.SYSAUDITPOLICIES to include the new policy.

2. Activate the audit policy by issuing the **START TRACE** command:

**GUI**

```
-STA TRACE (AUDIT) DEST (GTF) AUDTPLCY(AUDITADMN1)
```

**GUI**

**PSPI**

DB2 checks to make sure that you have the required privilege to run the **START TRACE** command. Upon successful verification, it starts an IFCID 361 trace record.

For example, if SYSADM1 issues the ALTER BUFFERPOOL command to alter the attributes for active buffer pools, DB2 records the ALTER activity in the

IFCID 361 trace record. **PSPI**

## Auditing tables without specifying the AUDIT clause

With the SECADM authority, you can set up audit policies and dynamically enable auditing of tables that do not have the AUDIT clause specified.

To audit the activities on table EMPLOYEE.SALARY without having to specify the AUDIT clause:

1. Obtain the SECADM authority if you do not have it. Alternately, you can have the SECADM authority grant you the required privileges to create an audit policy. A user with the SYSOPR authority can activate the policy.
2. Create audit policy TABADT1 by issuing the following INSERT statement:

**GUI**

```
INSERT INTO SYSIBM.SYSAUDITPOLICIES(AUDITPOLICYNAME, OBJECTSCHEMA,
OBJECTNAME, OBJECTTYPE, EXECUTE)
VALUES('TABADT1','EMPLOYEE','SALARY','T','A');
```

**GUI**

DB2 checks to make sure that you have the required privilege to issue the INSERT statement. Upon successful verification, it inserts a row in SYSIBM.SYSAUDITPOLICIES to include the new policy.

3. Activate the audit policy by issuing the **START TRACE** command:

**GUI**

```
-STA TRACE (AUDIT) DEST (GTF) AUDTPLCY(TABADT1);
```

GUPI

PSPI

DB2 checks to make sure that you have the required privilege to run the **START TRACE** command. Upon successful verification, it starts the IFCID 143, 144, and 145 trace records.

For example, if a user issues the SELECT statement to select from the EMPLOYEE.SALARY table, DB2 records the query activity in the IFCID 144

trace record. PSPI

---

## Additional sources of audit information

In addition to the audit trace, DB2 offers other sources of audit information for you to use.

### Additional DB2 traces

DB2 accounting, statistics, and performance traces are also available. You can also use DB2 Performance Expert to print reports of these traces.

### Recovery log

Although the recovery log is not an all-purpose log, it can be useful for auditing. You can print information from the log by using the DSN1LOGP utility. For example, the summary report can show which table spaces have been updated within the range of the log that you scan. The REPORT utility can indicate what log information is available and where it is located.

### Image copies of table spaces

Typical recovery procedures generate image copies of table spaces. You can inspect these copies, or use them with the RECOVER utility to recover a table space to a particular point in time. If you recover to a point in time, you narrow the time period during which a particular change could have been made.

### z/OS console log

The z/OS console log contains messages about exceptional conditions that are encountered during DB2 operation. Inspect this log for symptoms of problems.

---

## Determining ID privileges and authorities

As an auditor, you must be aware of the privileges and authorities that are associated with the IDs or roles in the DB2 subsystem.

You can use the following methods to determine the privileges and authorities that a specific ID or role holds:

- Query the DB2 catalog to determine which IDs or roles hold particular privileges.
- Check on individual IDs that are associated with group IDs or roles. Some authorization IDs that you encounter are probably group IDs, to which many individual IDs can be connected. To see which IDs are connected to a group, obtain a report from RACF or from whatever external security system you are using. These reports can tell you which IDs have the required privileges to use DB2 data sets and other resources.

---

## Auditing specific IDs or roles

As with other types of DB2 traces, you can start an audit trace for a particular plan name, a primary authorization ID, a role, or all of the above.

You might consider having audit traces on at all times for IDs with the SYSADM authority because they have complete access to every table. If you have a network of DB2 subsystems, you might need to trace multiple authorization IDs if the primary authorization IDs are translated several times. For embedded SQL, the audited ID is the primary authorization ID of the plan or package owner. For dynamic SQL, the audited ID is the primary authorization ID.

You can also start an audit trace for a particular role in a trusted context by using the ROLE and XROLE filters. For example, you can issue the following command to write accounting records for threads with a ROLE = abc:

### GUIP

```
-start trace(acctg) dest(smf) role(abc)
```

### GUIP

You can also issue the following command to write accounting records for threads with a ROLE= abc:

### GUIP

```
-start trace(acctg) dest(smf) xrole(abc)
```

### GUIP

In addition, you can use the asterisk (\*) wildcard character (as in "abc\*") or the underscore (\_) wildcard character (as in "a\_c") for more flexibility in audit tracing.

### Related tasks

“Auditing specific tables”

---

## Auditing specific tables

You can issue the CREATE TABLE or ALTER TABLE statement to audit a specific table.

### GUIP

**Example:** DB2 audits the department table whenever the audit trace is on if you create the table with the following statement:

```
CREATE TABLE DSN81010.DEPT
  (DEPTNO   CHAR(3)           NOT NULL,
   DEPTNAME VARCHAR(36)       NOT NULL,
   MGRNO    CHAR(6)           ,
   ADMRDEPT CHAR(3)           NOT NULL,
   LOCATION CHAR(16)          ,
   PRIMARY KEY (DEPTNO)       )
IN DSN8D10A.DSN8S10D
AUDIT CHANGES;
```

Because this statement includes the AUDIT CHANGES option, DB2 audits the table for each access that inserts, updates, or deletes data (trace class 4).

**Example:** To also audit the table for read accesses (class 5), issue the following statement:

```
ALTER TABLE DSN81010.DEPT  
  AUDIT ALL;
```

The statement is effective regardless of whether the table was previously chosen for auditing.

**Example:** To prevent all auditing of the table, issue the following statement:

```
ALTER TABLE DSN81010.DEPT  
  AUDIT NONE;
```

For the CREATE TABLE statement, the default audit option is NONE. For the ALTER TABLE statement, no default option exists. If you do not use the AUDIT clause in an ALTER TABLE statement, the audit option for the table is unchanged.

When CREATE TABLE statements or ALTER TABLE statements affect the audit of a table, you can audit those statements. However, the results of those audits are in audit class 3, not in class 4 or class 5. Use audit class 3 to determine whether auditing was turned off for a table for an interval of time.

If an ALTER TABLE statement turns auditing on or off for a specific table, any packages that use the table are invalidated and must be rebound. If you change the auditing status, the change does not affect packages, or dynamic SQL statements that are currently running. The change is effective only for packages or dynamic SQL statements that begin running after the ALTER TABLE statement has completed.



#### **Related tasks**

“Auditing specific IDs or roles” on page 295

---

## **Ensuring data accuracy and integrity**

DB2 provides many controls that you can apply to data entry and update.

Some of the controls are automatic; some are optional. All of the controls prohibit certain operations and provide error or warning messages if those operations are attempted. You can use these controls as a set auditing techniques to ensure data accuracy and integrity.

The set of techniques is not intended to be exhaustive. Other combinations of techniques are possible. For example, you can use table check constraints or a view with the check option to ensure that data values are members of a certain set. Or you can set up a master table and define referential constraints. You can also enforce the controls through application programs, and restrict the INSERT and UPDATE privileges only to those programs.

#### **Related tasks**

“Ensuring data consistency” on page 298

## **Ensuring data presence and uniqueness**

You can define columns with the NOT NULL clause to ensure that the required data is present. You can also control the type of data by assigning data types and lengths to column data.

For example, you can specify that alphabetic data cannot be entered into a column with one of the numeric data types. You can also specify that the data for a DATE or TIME column must use a specific format.

You must ensure that the data in a column or a set of columns is unique. You can do so by creating a unique index on a column or set of columns.

## Protecting data integrity

Triggers and table check constraints enhance the ability to control data integrity. Triggers are very useful for defining and enforcing rules that involve different states of DB2 data.

For example, a rule can prevent a salary column from more than a ten percent increase. A trigger can enforce this rule and provide the value of the salary before and after the increase for comparison.

Table check constraints designate the values that specific columns of a base table can contain. A check constraint can express simple constraints, such as a required pattern or a specific range, and rules that refer to other columns of the same table.

As an auditor, you can verify that the table definitions express the required constraints on column values as table check constraints. You can also create a view with the check option and insert or update values only through that view.

**GUIP** **Example:** Suppose that, in table T, data in column C1 must be a number between 10 and 20. Suppose also that data in column C2 is an alphanumeric code that must begin with A or B. Create view V1 with the following statement:

```
CREATE VIEW V1 AS
  SELECT * FROM T
    WHERE C1 BETWEEN 10 AND 20
      AND (C2 LIKE 'A%' OR C2 LIKE 'B%')
  WITH CHECK OPTION;
```

Because of the CHECK OPTION, view V1 allows only data that satisfies the WHERE clause. **GUIP**

You cannot use the LOAD utility with a view, but that restriction does not apply to user-written exit routines; you can consider using the following types of user-written routines:

### Validation routines

You can use validation routines to validate data values. Validation routines access an entire row of data, check the current plan name, and return a nonzero code to DB2 to indicate an invalid row.

### Edit routines

Edit routines have the same access as validation routines, and can also change the row that is to be inserted. Auditors typically use edit routines to encrypt data and to substitute codes for lengthy fields. However, edit routines can also validate data and return nonzero codes.

### Field procedures

Field procedures access data that is intended for a single column; they apply only to short-string columns. However, they accept input

parameters, so generalized procedures are possible. A column that is defined with a field procedure can be compared only to another column that uses the same procedure.

## Tracking data changes

Triggers offer an efficient means of maintaining an audit trail. You can define a trigger to activate in response to certain DELETE, INSERT, or UPDATE statements that change data.

You can qualify a trigger by providing a list of column names when you define the trigger. The qualified trigger is activated only when one of the named columns is changed. A trigger that performs validation for changes that are made in an UPDATE operation must access column values both before and after the update. Transition variables (available only to row triggers) contain the column values of the row change that activated the trigger. The old column values and the column values from after the triggering operation are both available.

## Checking for lost and incomplete transactions

You can use the database balancing technique to alert you about lost and incomplete transactions. For each set of data, database balancing determines whether the opening balance and control totals equal the closing balance and control totals of processed transactions.

DB2 has no automatic mechanism to calculate control totals and column balances and compare them with transaction counts and field totals. Therefore, to use database balancing, you must design these mechanisms into the application program.

**Example:** Use your application program to maintain a control table. The control table contains information to balance the control totals and field balances for update transactions against a user's view. The control table might contain these columns:

- View name
- Authorization ID
- Number of logical rows in the view (not the same as the number of physical rows in the table)
- Number of insert transactions and update transactions
- Opening balances
- Totals of insert transaction amounts and update transaction amounts
- Relevant audit trail information such as date, time, workstation ID, and job name

The program updates the transaction counts and amounts in the control table each time it completes an insert or update to the view. To maintain coordination during recovery, the program commits the work only after it updates the control table. After the application processes all transactions, the application writes a report that verifies the control total and balancing information.

---

## Ensuring data consistency

When you control data entry, you perform only part of a complete security and auditing policy. You must also verify the results when data is accessed and changed. In addition, you need to make sure that your data is consistent.



### Related concepts

“Ensuring data accuracy and integrity” on page 296

## Using referential integrity for data consistency

Referential integrity ensures that data is consistent across tables. When you define primary and foreign keys, DB2 automatically enforces referential integrity.

As a result, every value of a foreign key in a dependent table must be a value of a primary key in the appropriate parent table. However, DB2 does not enforce informational referential constraints across subsystems.

**Recommendation:** Use referential integrity to ensure that a column allows only specific values. Set up a master table of allowable values, and define its primary key. Define foreign keys in other tables that must have matching values in their columns. In most cases, you should use the SET NULL delete rule.

### Related tasks

“Using locks for data consistency”

“Checking data consistency” on page 300

## Using locks for data consistency

Locks can ensure that data remains consistent even when multiple users try to access the same data at the same time. You can use locks to ensure that only one user is privileged to change data at a given time and that no user is privileged to access uncommitted data.

If you use repeatable read (RR), read stability (RS), or cursor stability (CS) as your isolation level, DB2 automatically controls access to data by using locks. However, if you use uncommitted read (UR) as your isolation level, users can access uncommitted data and introduce inconsistent data. As an auditor, you must know the applications that use UR isolation and that can introduce inconsistent data or create security risks.

 For static SQL, you can determine the plans and packages that use UR isolation by querying the catalog.

**Example:** For static SQL statements, use the following query to determine which plans use UR isolation:

```
SELECT DISTINCT Y.PLNAME
  FROM SYSIBM.SYSPLAN X, SYSIBM.SYSSTMT Y
 WHERE (X.NAME = Y.PLNAME AND X.ISOLATION = 'U')
        OR Y.ISOLATION = 'U'
 ORDER BY Y.PLNAME;
```

**Example:** For static SQL statements, use the following query to determine which packages use UR isolation:

```
SELECT DISTINCT Y.COLLOID, Y.NAME, Y.VERSION
  FROM SYSIBM.SYSPACKAGE X, SYSIBM.SYSPACKSTMT Y
 WHERE (X.LOCATION = Y.LOCATION AND
        X.COLLOID = Y.COLLOID AND
        X.NAME = Y.NAME AND
        X.VERSION = Y.VERSION AND
        X.ISOLATION = 'U')
        OR Y.ISOLATION = 'U'
 ORDER BY Y.COLLOID, Y.NAME, Y.VERSION;
```



For dynamic SQL statements, turn on performance trace class 3 to determine which plans and packages use UR isolation. 

**Consistency between systems:** When an application program writes data to both DB2 and IMS, or to both DB2 and CICS, the subsystems prevent concurrent use of data until the program declares a point of consistency.

#### Related tasks

“Using referential integrity for data consistency” on page 299

“Checking data consistency”

## Checking data consistency

Whenever an operation changes the contents of a data page or an index page, DB2 verifies that the modifications do not produce inconsistent data.

You can run the DSN1CHKR utility to verify the integrity of the DB2 catalog and the directory table spaces. You can also run this utility to scan the specified table space for broken links, damaged hash chains, or orphan entries.

You can use a variety of other SQL queries, commands, and utilities to check data consistency.

#### Related tasks

“Using referential integrity for data consistency” on page 299

“Using locks for data consistency” on page 299

### Checking data consistency with SQL queries

If you suspect that a table contains inconsistent data, you can submit an SQL query to search for a specific type of error.

**Example:** Consider the view that is created by the following statement as an example:

```
CREATE VIEW V1 AS
  SELECT * FROM T
    WHERE C1 BETWEEN 10 AND 20
      AND (C2 LIKE 'A%' OR C2 LIKE 'B%')
  WITH CHECK OPTION;
```

The view allows an insert or update to table T1 only if the value in column C1 is between 10 and 20 and if the value in C2 begins with A or B. To check that the control has not been bypassed, issue the following statement:

```
SELECT * FROM T1
  WHERE NOT (C1 BETWEEN 10 AND 20
    AND (C2 LIKE 'A
```

If the control has not been bypassed, DB2 returns no rows and thereby confirms that the contents of the view are valid. You can also use SQL statements to get information from the DB2 catalog about referential constraints that exist.

### Checking data consistency with the CHECK utilities

You can use the CHECK DATA, CHECK INDEX, and CHECK LOB online utilities to ensure data consistency.

#### CHECK DATA

The CHECK DATA utility checks referential constraints (but not

informational referential constraints). It determines whether each foreign key value in each row is a value of the primary key in the appropriate parent table.

The CHECK DATA utility also checks table check constraints and checks the consistency between a base table space and any associated LOB or XML table spaces. It determines whether each value in a row is within the range that was specified for that column when the table was created.

The CHECK DATA utility also performs consistency checks on XML table spaces and related NodeID indexes. It verifies the consistency of XML documents that are stored in an XML table space and validates the documents against one or more XML schemas that are specified in the XML type modifier.

#### **CHECK INDEX**

The CHECK INDEX utility checks the consistency of indexes with the data to which the indexes point. It determines whether each index pointer points to a data row with the same value as the index key. If an index key points to a LOB, the CHECK INDEX utility determines whether the index key points to the correct LOB. If an index key points to an XML, the CHECK INDEX utility determines whether the index key points to the correct XML.

#### **CHECK LOB**

The CHECK LOB utility checks the consistency of a LOB table space. It determines whether any LOBs in the LOB table space are invalid.

### **Checking data consistency with the DISPLAY DATABASE command**

If you allow a table to be loaded without enforcing referential constraints on its foreign key columns, the table might contain data that violates the constraints. In this case, DB2 places the table space that contains the table in the CHECK-pending status.

You can determine the table spaces with the CHECK-pending status by using the DISPLAY DATABASE command with the RESTRICT option. You can also use the DISPLAY DATABASE command to display table spaces with invalid LOBs.

### **Checking data consistency with the REPORT utility**

You can use the REPORT utility with the TABLESPACESET keyword to retrieve certain information about data consistency.

- Table spaces that contain a set of tables interconnected by referential constraints
- LOB or XML table spaces that are associated with base tables
- Base table column and partition numbers that are associated with each LOB or XML table space.

### **Checking data consistency with the operation log**

You can use the operation log to verify that DB2 is operated reliably and to reveal unauthorized operations and overrides. The operation log consists of an automated log of DB2 operator commands, such as those for starting and stopping the subsystem, and DB2 abends.

The operation log records the following information:

- Command or condition type
- Date and time when the command was issued
- Authorization ID that issued the command

- Database connection code

You can obtain this information from the system log (SYSLOG), the SMF data set, or the automated job scheduling system. To obtain the information, use SMF reporting, job-scheduler reporting, or a user-developed program. As a good practice, review the log report daily and keep a history file for comparison. Because abnormal DB2 termination can indicate integrity problems, implement an immediate notification procedure to alert the appropriate personnel (DBA, systems supervisor, and so on) of abnormal DB2 terminations.

### **Checking data consistency with internal integrity reports**

You can generate internal integrity reports for application programs and utilities.

For application programs, you can record any DB2 return codes that indicate possible data integrity problems, such as inconsistency between index and table information, physical errors on database disk, and so on. All programs must check the SQLCODE or the SQLSTATE for the return code that is issued after an SQL statement is run. DB2 records, on SMF, the occurrence (but not the cause) of physical disk errors and application program abends. The program can retrieve and report this information; the system log (SYSLOG) and the DB2 job output also have this information. However, in some cases, only the program can provide enough detail to identify the exact nature of problem.

You can incorporate these integrity reports into application programs, or you can use them separately as part of an interface. The integrity report records the incident in a history file and writes a message to the operator's console, a database administrator's TSO terminal, or a dedicated printer for certain codes. The recorded information includes the following:

- Date
- Time
- Authorization ID
- Terminal ID or job name
- Application
- Affected view or affected table
- Error code
- Error description

When a DB2 utility reorganizes or reconstructs data in the database, it produces statistics to verify record counts and to report errors. The LOAD and REORG utilities produce data record counts and index counts to verify that no records were lost. In addition to that, keep a history log of any DB2 utility that updates data, particularly REPAIR. Regularly produce and review these reports, which you can obtain through SMF customized reporting or a user-developed program.

---

## Information resources for DB2 for z/OS and related products

Many information resources are available to help you use DB2 for z/OS and many related products. A large amount of technical information about IBM products is now available online in information centers or on library websites.

**Disclaimer:** Any web addresses that are included here are accurate at the time this information is being published. However, web addresses sometimes change. If you visit a web address that is listed here but that is no longer valid, you can try to find the current web address for the product information that you are looking for at either of the following sites:

- <http://www.ibm.com/support/publications/us/library/index.shtml>, which lists the IBM information centers that are available for various IBM products
- <http://www.ibm.com/shop/publications/order>, which is the IBM Publications Center, where you can download online PDF books or order printed books for various IBM products

### DB2 for z/OS product information

The primary place to find and use information about DB2 for z/OS is the Information Management Software for z/OS Solutions Information Center (<http://publib.boulder.ibm.com/infocenter/imzic>), which also contains information about IMS, QMF, and many DB2 and IMS Tools products. This information center is also available as an installable information center that can run on a local system or on an intranet server. You can order the Information Management for z/OS Solutions Information Center DVD (SK5T-7377) for a low cost from the IBM Publications Center (<http://www.ibm.com/shop/publications/order>).

The majority of the DB2 for z/OS information in this information center is also available in the books that are identified in the following table. You can access these books at the DB2 for z/OS library website (<http://www.ibm.com/software/data/db2/zos/library.html>) or at the IBM Publications Center (<http://www.ibm.com/shop/publications/order>).

Table 82. DB2 10 for z/OS book titles

Title	Publication number	Available in information center	Available in PDF	Available in printed format
<i>DB2 10 for z/OS Administration Guide</i>	SC19-2968	X	X	
<i>DB2 10 for z/OS Application Programming &amp; SQL Guide</i>	SC19-2969	X	X	
<i>DB2 10 for z/OS Application Programming Guide and Reference for Java</i>	SC19-2970	X	X	
<i>DB2 10 for z/OS Codes</i>	GC19-2971	X	X	
<i>DB2 10 for z/OS Command Reference</i>	SC19-2972	X	X	
<i>DB2 10 for z/OS Data Sharing: Planning and Administration</i>	SC19-2973	X	X	
<i>DB2 10 for z/OS Diagnosis Guide and Reference</i>	LY37-3220		X	X

Table 82. DB2 10 for z/OS book titles (continued)

Title	Publication number	Available in information center	Available in PDF	Available in printed format
<i>DB2 10 for z/OS Installation and Migration Guide</i>	GC19-2974	X	X	
<i>DB2 10 for z/OS Internationalization Guide</i>	SC19-2975	X	X	
<i>DB2 10 for z/OS Introduction to DB2</i>	SC19-2976	X	X	
<i>DB2 10 for z/OS Licensed Program Specifications</i>	GC19-2977		X	X
<i>DB2 10 for z/OS Managing Performance</i>	SC19-2978	X	X	
<i>DB2 10 for z/OS Messages</i>	GC19-2979	X	X	
<i>DB2 10 for z/OS ODBC Guide and Reference</i>	SC19-2980	X	X	
<i>DB2 10 for z/OS Program Directory</i>	GI10-8829		X	X
<i>DB2 10 for z/OS pureXML Guide</i>	SC19-2981	X	X	
<i>DB2 10 for z/OS RACF Access Control Module Guide</i>	SC19-2982	X	X	
<i>DB2 10 for z/OS SQL Reference</i>	SC19-2983	X	X	
<i>DB2 10 for z/OS Utility Guide and Reference</i>	SC19-2984	X	X	
<i>DB2 10 for z/OS What's New?</i>	GC19-2985	X	X	
<i>IRLM Messages and Codes for IMS and DB2 for z/OS</i>	GC19-2666	X	X	

**Note:**

1. *DB2 10 for z/OS Diagnosis Guide and Reference* is available in PDF format on the DB2 10 for z/OS Licensed Library Collection kit, LK5T-7390. You can order this Licensed Library Collection kit on the IBM Publications Center site (<http://www.ibm.com/e-business/linkweb/publications/servlet/pbi.wss>). This book is also available in online format in DB2 data set DSNA10.SDSNIVPD(DSNDR).

## Information resources for related products

In the following table, related product names are listed in alphabetic order, and the associated web addresses of product information centers or library web pages are indicated.

Table 83. Related product information resource locations

Related product	Information resources
C/C++ for z/OS	Library website: <a href="http://www.ibm.com/software/awdtools/czos/library/">http://www.ibm.com/software/awdtools/czos/library/</a>  This product is now called z/OS XL C/C++.
CICS Transaction Server for z/OS	Information center: <a href="http://publib.boulder.ibm.com/infocenter/cicsts/v3r1/index.jsp">http://publib.boulder.ibm.com/infocenter/cicsts/v3r1/index.jsp</a>
COBOL	Information center: <a href="http://publib.boulder.ibm.com/infocenter/pdthelp/v1r1/index.jsp">http://publib.boulder.ibm.com/infocenter/pdthelp/v1r1/index.jsp</a>  This product is now called Enterprise COBOL for z/OS.
DB2 Connect	Information center: <a href="http://publib.boulder.ibm.com/infocenter/db2luw/v9/index.jsp">http://publib.boulder.ibm.com/infocenter/db2luw/v9/index.jsp</a>  This resource is for DB2 Connect 9.
DB2 Database for Linux, UNIX, and Windows	Information center: <a href="http://publib.boulder.ibm.com/infocenter/db2luw/v9/index.jsp">http://publib.boulder.ibm.com/infocenter/db2luw/v9/index.jsp</a>  This resource is for DB2 9 for Linux, UNIX, and Windows.

Table 83. Related product information resource locations (continued)

Related product	Information resources
DB2 Query Management Facility™	Information center: <a href="http://publib.boulder.ibm.com/infocenter/imzic">http://publib.boulder.ibm.com/infocenter/imzic</a>
DB2 Server for VSE & VM	Product website: <a href="http://www.ibm.com/software/data/db2/vse-vm/">http://www.ibm.com/software/data/db2/vse-vm/</a>
DB2 Tools	<p>One of the following locations:</p> <ul style="list-style-type: none"> <li>• Information center: <a href="http://publib.boulder.ibm.com/infocenter/imzic">http://publib.boulder.ibm.com/infocenter/imzic</a></li> <li>• Library website: <a href="http://www.ibm.com/software/data/db2imstools/library.html">http://www.ibm.com/software/data/db2imstools/library.html</a></li> </ul> <p>These resources include information about the following products and others:</p> <ul style="list-style-type: none"> <li>• DB2 Administration Tool</li> <li>• DB2 Automation Tool</li> <li>• DB2 Log Analysis Tool</li> <li>• DB2 Object Restore Tool</li> <li>• DB2 Query Management Facility</li> <li>• DB2 SQL Performance Analyzer</li> </ul>
DB2 Universal Database™ for iSeries®	Information center: <a href="http://www.ibm.com/systems/i/infocenter/">http://www.ibm.com/systems/i/infocenter/</a>
Debug Tool for z/OS	Information center: <a href="http://publib.boulder.ibm.com/infocenter/pdthelp/v1r1/index.jsp">http://publib.boulder.ibm.com/infocenter/pdthelp/v1r1/index.jsp</a>
Enterprise COBOL for z/OS	Information center: <a href="http://publib.boulder.ibm.com/infocenter/pdthelp/v1r1/index.jsp">http://publib.boulder.ibm.com/infocenter/pdthelp/v1r1/index.jsp</a>
Enterprise PL/I for z/OS	Information center: <a href="http://publib.boulder.ibm.com/infocenter/pdthelp/v1r1/index.jsp">http://publib.boulder.ibm.com/infocenter/pdthelp/v1r1/index.jsp</a>
InfoSphere® Replication Server for z/OS	<p>Information center: <a href="http://publib.boulder.ibm.com/infocenter/dzichelp/v2r2/topic/com.ibm.swg.im.iis.db.prod.repl.nav.doc/dochome/iiryrcnav_dochome.html">http://publib.boulder.ibm.com/infocenter/dzichelp/v2r2/topic/com.ibm.swg.im.iis.db.prod.repl.nav.doc/dochome/iiryrcnav_dochome.html</a></p> <p>This product was also known as DB2 DataPropagator, DB2 Information Integrator Replication Edition for z/OS, and WebSphere Replication Server for z/OS.</p>
IMS	Information center: <a href="http://publib.boulder.ibm.com/infocenter/imzic">http://publib.boulder.ibm.com/infocenter/imzic</a>
IMS Tools	<p>One of the following locations:</p> <ul style="list-style-type: none"> <li>• Information center: <a href="http://publib.boulder.ibm.com/infocenter/imzic">http://publib.boulder.ibm.com/infocenter/imzic</a></li> <li>• Library website: <a href="http://www.ibm.com/software/data/db2imstools/library.html">http://www.ibm.com/software/data/db2imstools/library.html</a></li> </ul> <p>These resources have information about the following products and others:</p> <ul style="list-style-type: none"> <li>• IMS Batch Terminal Simulator for z/OS</li> <li>• IMS Connect</li> <li>• IMS HALDB Conversion and Maintenance Aid</li> <li>• IMS High Performance Utility products</li> <li>• IMS DataPropagator</li> <li>• IMS Online Reorganization Facility</li> <li>• IMS Performance Analyzer</li> </ul>
Integrated Data Management products	<p>Information center: <a href="http://publib.boulder.ibm.com/infocenter/idm/v2r2/index.jsp">http://publib.boulder.ibm.com/infocenter/idm/v2r2/index.jsp</a></p> <p>This information center has information about the following products and others:</p> <ul style="list-style-type: none"> <li>• IBM Data Studio</li> <li>• InfoSphere Data Architect</li> <li>• InfoSphere Warehouse</li> <li>• Optim™ Database Administrator</li> <li>• Optim Development Studio</li> <li>• Optim Query Tuner</li> </ul>

Table 83. Related product information resource locations (continued)

Related product	Information resources
PL/I	Information center: <a href="http://publib.boulder.ibm.com/infocenter/pdthelp/v1r1/index.jsp">http://publib.boulder.ibm.com/infocenter/pdthelp/v1r1/index.jsp</a>  This product is now called Enterprise PL/I for z/OS.
System z <sup>®</sup>	<a href="http://publib.boulder.ibm.com/infocenter/eserver/v1r2/index.jsp">http://publib.boulder.ibm.com/infocenter/eserver/v1r2/index.jsp</a>
Tivoli OMEGAMON XE for DB2 Performance Expert on z/OS	Information center: <a href="http://publib.boulder.ibm.com/infocenter/tivihelp/v15r1/topic/com.ibm.omegamon.xe_db2.doc/ko2welcome_pe.htm">http://publib.boulder.ibm.com/infocenter/tivihelp/v15r1/topic/com.ibm.omegamon.xe_db2.doc/ko2welcome_pe.htm</a>  In earlier releases, this product was called DB2 Performance Expert for z/OS.
WebSphere Application Server	Information center: <a href="http://publib.boulder.ibm.com/infocenter/wasinfo/v6r0/index.jsp">http://publib.boulder.ibm.com/infocenter/wasinfo/v6r0/index.jsp</a>
WebSphere Message Broker with Rules and Formatter Extension	Information center: <a href="http://publib.boulder.ibm.com/infocenter/wmbhelp/v6r0m0/index.jsp">http://publib.boulder.ibm.com/infocenter/wmbhelp/v6r0m0/index.jsp</a>  The product is also known as WebSphere MQ Integrator Broker.
WebSphere MQ	Information center: <a href="http://publib.boulder.ibm.com/infocenter/wmqv6/v6r0/index.jsp">http://publib.boulder.ibm.com/infocenter/wmqv6/v6r0/index.jsp</a>  The resource includes information about MQSeries <sup>®</sup> .
z/Architecture <sup>®</sup>	Library Center site: <a href="http://www.ibm.com/servers/eserver/zseries/zos/bkserv/">http://www.ibm.com/servers/eserver/zseries/zos/bkserv/</a>

Table 83. Related product information resource locations (continued)

Related product	Information resources
z/OS	<p>Library Center site: <a href="http://www.ibm.com/servers/eserver/zseries/zos/bkserv/">http://www.ibm.com/servers/eserver/zseries/zos/bkserv/</a></p> <p>This resource includes information about the following z/OS elements and components:</p> <ul style="list-style-type: none"> <li>• Character Data Representation Architecture</li> <li>• Device Support Facilities</li> <li>• DFSORT</li> <li>• Fortran</li> <li>• High Level Assembler</li> <li>• NetView<sup>®</sup></li> <li>• SMP/E for z/OS</li> <li>• SNA</li> <li>• TCP/IP</li> <li>• TotalStorage Enterprise Storage Server<sup>®</sup></li> <li>• VTAM</li> <li>• z/OS C/C++</li> <li>• z/OS Communications Server</li> <li>• z/OS DCE</li> <li>• z/OS DFSMS</li> <li>• z/OS DFSMS Access Method Services</li> <li>• z/OS DFSMSdss</li> <li>• z/OS DFSMSHsm</li> <li>• z/OS DFSMSdftp</li> <li>• z/OS ICSF</li> <li>• z/OS ISPF</li> <li>• z/OS JES3</li> <li>• z/OS Language Environment<sup>®</sup></li> <li>• z/OS Managed System Infrastructure</li> <li>• z/OS MVS</li> <li>• z/OS MVS JCL</li> <li>• z/OS Parallel Sysplex<sup>®</sup></li> <li>• z/OS RMF<sup>™</sup></li> <li>• z/OS Security Server</li> <li>• z/OS UNIX System Services</li> </ul>
z/OS XL C/C++	<p><a href="http://www.ibm.com/software/awdtools/czos/library/">http://www.ibm.com/software/awdtools/czos/library/</a></p>

The following information resources from IBM are not necessarily specific to a single product:

- The DB2 for z/OS Information Roadmap; available at: <http://www.ibm.com/software/data/db2/zos/roadmap.html>
- DB2 Redbooks<sup>®</sup> and Redbooks about related products; available at: <http://www.ibm.com/redbooks>
- IBM Educational resources:
  - Information about IBM educational offerings is available on the web at: <http://www.ibm.com/software/sw-training/>



- A collection of glossaries of IBM terms in multiple languages is available on the IBM Terminology website at: <http://www.ibm.com/software/globalization/terminology/index.jsp>
- National Language Support information; available at the IBM Publications Center at: <http://www.elink.ibm.link.ibm.com/public/applications/publications/cgibin/pbi.cgi>
- *SQL Reference for Cross-Platform Development*; available at the following developerWorks® site: <http://www.ibm.com/developerworks/db2/library/techarticle/0206sqlref/0206sqlref.html>

The following information resources are not published by IBM but can be useful to users of DB2 for z/OS and related products:

- Database design topics:
  - *DB2 for z/OS and OS/390® Development for Performance Volume I*, by Gabrielle Wiorkowski, Gabrielle & Associates, ISBN 0-96684-605-2
  - *DB2 for z/OS and OS/390 Development for Performance Volume II*, by Gabrielle Wiorkowski, Gabrielle & Associates, ISBN 0-96684-606-0
  - *Handbook of Relational Database Design*, by C. Fleming and B. Von Halle, Addison Wesley, ISBN 0-20111-434-8
- Distributed Relational Database Architecture™ (DRDA) specifications; <http://www.opengroup.org>
- Domain Name System: *DNS and BIND*, Third Edition, Paul Albitz and Cricket Liu, O'Reilly, ISBN 0-59600-158-4
- Microsoft Open Database Connectivity (ODBC) information; <http://msdn.microsoft.com/library/>
- Unicode information; <http://www.unicode.org>

---

## How to obtain DB2 information

You can access the official information about the DB2 product in a number of ways.

- “DB2 on the web”
- “DB2 product information”
- “DB2 education” on page 310
- “How to order the DB2 library” on page 310

### DB2 on the web

Stay current with the latest information about DB2 by visiting the DB2 home page on the web:

<http://www.ibm.com/software/db2zos>

On the DB2 home page, you can find links to a wide variety of information resources about DB2. You can read news items that keep you informed about the latest enhancements to the product. Product announcements, press releases, fact sheets, and technical articles help you plan and implement your database management strategy.

### DB2 product information

The official DB2 for z/OS information is available in various formats and delivery methods. IBM provides mid-version updates to the information in the information center and in softcopy updates that are available on the web and on CD-ROM.

#### Information Management Software for z/OS Solutions Information Center

DB2 product information is viewable in the information center, which is the primary delivery vehicle for information about DB2 for z/OS, IMS, QMF, and related tools. This information center enables you to search across related product information in multiple languages for data management solutions for the z/OS environment and print individual topics or sets of related topics. You can also access, download, and print PDFs of the publications that are associated with the information center topics. Product technical information is provided in a format that offers more options and tools for accessing, integrating, and customizing information resources. The information center is based on Eclipse open source technology.

The Information Management Software for z/OS Solutions Information Center is viewable at the following website:

<http://publib.boulder.ibm.com/infocenter/imzic>

#### CD-ROMs and DVD

Books for DB2 are available on a CD-ROM that is included with your product shipment:

- DB2 10 for z/OS Licensed Library Collection, LK5T-7390, in English

The CD-ROM contains the collection of books for DB2 10 for z/OS in PDF format. Periodically, IBM refreshes the books on subsequent editions of this CD-ROM.

The books for DB2 for z/OS are also available on the following DVD collection kit, which contains online books for many IBM products:

- IBM z/OS Software Products DVD Collection, SK3T-4271, in English

#### **PDF format**

Many of the DB2 books are available in PDF (Portable Document Format) for viewing or printing from CD-ROM or the DB2 home page on the web or from the information center. Download the PDF books to your intranet for distribution throughout your enterprise.

#### **DB2 education**

IBM Education and Training offers a wide variety of classroom courses to help you quickly and efficiently gain DB2 expertise. IBM schedules classes in cities all over the world. You can find class information, by country, at the IBM Learning Services website:

<http://www.ibm.com/services/learning>

IBM also offers classes at your location, at a time that suits your needs. IBM can customize courses to meet your exact requirements. For more information, including the current local schedule, contact your IBM representative.

#### **How to order the DB2 library**

To order books, visit the IBM Publication Center on the web:

<http://www.ibm.com/shop/publications/order>

From the IBM Publication Center, you can go to the Publication Notification System (PNS). PNS users receive electronic notifications of updated publications in their profiles. You have the option of ordering the updates by using the publications direct ordering application or any other IBM publication ordering channel. The PNS application does not send automatic shipments of publications. You will receive updated publications and a bill for them if you respond to the electronic notification.

You can also order DB2 publications and CD-ROMs from your IBM representative or the IBM branch office that serves your locality. If your location is within the United States or Canada, you can place your order by calling one of the toll-free numbers:

- In the U.S., call 1-800-879-2755.
- In Canada, call 1-800-426-4968.

To order additional copies of licensed publications, specify the SOFTWARE option. To order additional publications or CD-ROMs, specify the PUBLICATIONS option. Be prepared to give your customer number, the product number, and either the feature codes or order numbers that you want.

---

## How to use the DB2 library

Titles of books in the library begin with DB2 10 for z/OS. However, references from one book in the library to another are shortened and do not include the product name, version, and release. Instead, they point directly to the section that holds the information. The primary place to find and use information about DB2 for z/OS is the Information Management Software for z/OS Solutions Information Center (<http://publib.boulder.ibm.com/infocenter/imzic>).

If you are new to DB2 for z/OS, *Introduction to DB2 for z/OS* provides a comprehensive introduction to DB2 10 for z/OS. Topics included in this book explain the basic concepts that are associated with relational database management systems in general, and with DB2 for z/OS in particular.

The most rewarding task associated with a database management system is asking questions of it and getting answers, the task called *end use*. Other tasks are also necessary—defining the parameters of the system, putting the data in place, and so on. The tasks that are associated with DB2 are grouped into the following major categories.

### Installation

If you are involved with installing DB2, you will need to use a variety of resources, such as:

- *DB2 Program Directory*
- *DB2 Installation and Migration Guide*
- *DB2 Administration Guide*
- *DB2 Application Programming Guide and Reference for Java*
- *DB2 Codes*
- *DB2 Internationalization Guide*
- *DB2 Messages*
- *DB2 Managing Performance*
- *DB2 RACF Access Control Module Guide*
- *DB2 Utility Guide and Reference*

If you will be using data sharing capabilities you also need *DB2 Data Sharing: Planning and Administration*, which describes installation considerations for data sharing.

If you will be installing and configuring DB2 ODBC, you will need *DB2 ODBC Guide and Reference*.

If you are installing IBM Spatial Support for DB2 for z/OS, you will need *IBM Spatial Support for DB2 for z/OS User's Guide and Reference*.

If you are installing IBM OmniFind® Text Search Server for DB2 for z/OS, you will need *IBM OmniFind Text Search Server for DB2 for z/OS Installation, Administration, and Reference*.

## End use

End users issue SQL statements to retrieve data. They can also insert, update, or delete data, with SQL statements. They might need an introduction to SQL, detailed instructions for using SPUI, and an alphabetized reference to the types of SQL statements. This information is found in *DB2 Application Programming and SQL Guide*, and *DB2 SQL Reference*.

End users can also issue SQL statements through the DB2 Query Management Facility (QMF) or some other program, and the library for that licensed program might provide all the instruction or reference material they need.

## Application programming

Some users access DB2 without knowing it, using programs that contain SQL statements. DB2 application programmers write those programs. Because they write SQL statements, they need the same resources that end users do.

Application programmers also need instructions for many other topics:

- How to transfer data between DB2 and a host program—written in Java, C, or COBOL, for example
- How to prepare to compile a program that embeds SQL statements
- How to process data from two systems simultaneously, for example, DB2 and IMS or DB2 and CICS
- How to write distributed applications across operating systems
- How to write applications that use Open Database Connectivity (ODBC) to access DB2 servers
- How to write applications that use JDBC and SQLJ with the Java programming language to access DB2 servers
- How to write applications to store XML data on DB2 servers and retrieve XML data from DB2 servers.

The material needed for writing a host program containing SQL is in *DB2 Application Programming and SQL Guide*.

The material needed for writing applications that use JDBC and SQLJ to access DB2 servers is in *DB2 Application Programming Guide and Reference for Java*. The material needed for writing applications that use DB2 CLI or ODBC to access DB2 servers is in *DB2 ODBC Guide and Reference*. The material needed for working with XML data in DB2 is in *DB2 pureXML Guide*. For handling errors, see *DB2 Messages and DB2 Codes*.

Information about writing applications across operating systems can be found in *IBM DB2 SQL Reference for Cross-Platform Development*.

## System and database administration

*Administration* covers almost everything else. *DB2 Administration Guide* divides some of those tasks among the following sections:

- Designing a database: Discusses the decisions that must be made when designing a database and tells how to implement the design by creating and altering DB2 objects, loading data, and adjusting to changes.

- Security and auditing: Describes ways of controlling access to the DB2 system and to data within DB2, to audit aspects of DB2 usage, and to answer other security and auditing concerns.
- Operation and recovery: Describes the steps in normal day-to-day operation and discusses the steps one should take to prepare for recovery in the event of some failure.

*DB2 Managing Performance* explains how to monitor the performance of the DB2 system and its parts. It also lists things that can be done to make some parts run faster.

If you will be using the RACF access control module for DB2 authorization checking, you will need *DB2 RACF Access Control Module Guide*.

If you are involved with DB2 only to design the database, or plan operational procedures, you need *DB2 Administration Guide*. If you also want to carry out your own plans by creating DB2 objects, granting privileges, running utility jobs, and so on, you also need:

- *DB2 SQL Reference*, which describes the SQL statements you use to create, alter, and drop objects and grant and revoke privileges
- *DB2 Utility Guide and Reference*, which explains how to run utilities
- *DB2 Command Reference*, which explains how to run commands

If you will be using data sharing, you need *DB2 Data Sharing: Planning and Administration*, which describes how to plan for and implement data sharing.

Additional information about system and database administration can be found in *DB2 Messages* and *DB2 Codes*, which list messages and codes issued by DB2, with explanations and suggested responses.

## Diagnosis

Diagnostics detect and describe errors in the DB2 program. They might also recommend or apply a remedy. The documentation for this task is in *DB2 Diagnosis Guide and Reference*, *DB2 Messages*, and *DB2 Codes*.



---

## 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 or send inquiries, in writing, to:

Intellectual Property Licensing  
Legal and Intellectual Property Law  
IBM Japan, Ltd.  
1623-14, Shimotsuruma, Yamato-shi  
Kanagawa 242-8502 Japan

**The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:**

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM websites are provided for convenience only and do not in any manner serve as an endorsement of those websites. The materials at those websites are not part of the materials for this IBM product and use of those websites is at your own risk.



IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation  
J46A/G4  
555 Bailey Avenue  
San Jose, CA 95141-1003  
U.S.A.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement, or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

#### COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. The sample programs are provided "AS IS", without warranty of any kind. IBM shall not be liable for any damages arising out of your use of the sample programs.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

---



## Programming Interface Information

This information is intended to help you to plan for and administer DB2 10 for z/OS. This information also documents General-use Programming Interface and Associated Guidance Information and Product-sensitive Programming Interface and Associated Guidance Information provided by DB2 10 for z/OS.

### General-use Programming Interface and Associated Guidance Information

General-use Programming Interfaces allow the customer to write programs that obtain the services of DB2 10 for z/OS.

General-use Programming Interface and Associated Guidance Information is identified where it occurs by the following markings:

 General-use Programming Interface and Associated Guidance Information...  


### Product-sensitive Programming Interface and Associated Guidance Information

Product-sensitive Programming Interfaces allow the customer installation to perform tasks such as diagnosing, modifying, monitoring, repairing, tailoring, or tuning of this IBM software product. Use of such interfaces creates dependencies on the detailed design or implementation of the IBM software product. Product-sensitive Programming Interfaces should be used only for these specialized purposes. Because of their dependencies on detailed design and implementation, it is to be expected that programs written to such interfaces may need to be changed in order to run with new product releases or versions, or as a result of service.

Product-sensitive Programming Interface and Associated Guidance Information is identified where it occurs by the following markings:

 Product-sensitive Programming Interface and Associated Guidance Information...  


---

## Trademarks

IBM, the IBM logo, and [ibm.com](http://www.ibm.com)<sup>®</sup> are trademarks or registered marks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the web at <http://www.ibm.com/legal/copytrade.shtml>.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

---

## Glossary

**abend** See abnormal end of task.

**abend reason code**

A 4-byte hexadecimal code that uniquely identifies a problem with DB2.

**abnormal end of task (abend)**

Termination of a task, job, or subsystem because of an error condition that recovery facilities cannot resolve during execution.

**access method services**

The facility that is used to define, alter, delete, print, and reproduce VSAM key-sequenced data sets.

**access path**

The path that is used to locate data that is specified in SQL statements. An access path can be indexed or sequential.

**access path stability**

A characteristic of an access path that defines reliability for dynamic or static queries. Access paths are not regenerated unless there is a schema change or manual intervention.

**active log**

The portion of the DB2 log to which log records are written as they are generated. The active log always contains the most recent log records. See also archive log.

**address space**

A range of virtual storage pages that is identified by a number (ASID) and a collection of segment and page tables that map the virtual pages to real pages of the computer's memory.

**address space connection**

The result of connecting an allied address space to DB2. See also allied address space and task control block.

**address space identifier (ASID)**

A unique system-assigned identifier for an address space.

**AFTER trigger**

A trigger that is specified to be activated after a defined trigger event (an insert, update, or delete operation on the table that is specified in a trigger definition).

Contrast with BEFORE trigger and INSTEAD OF trigger.

**agent** In DB2, the structure that associates all processes that are involved in a DB2 unit of work. See also allied agent and system agent.

**aggregate function**

An operation that derives its result by using values from one or more rows. Contrast with scalar function.

**alias**

An alternative name that can be used in SQL statements to refer to a table or view in the same or a remote DB2 subsystem. An alias can be qualified with a schema qualifier and can thereby be referenced by other users. Contrast with synonym.

**allied address space**

An area of storage that is external to DB2 and that is connected to DB2. An allied address space can request DB2 services. See also address space.

**allied agent**

An agent that represents work requests that originate in allied address spaces. See also system agent.

**allied thread**

A thread that originates at the local DB2 subsystem and that can access data at a remote DB2 subsystem.

**allocated cursor**

A cursor that is defined for a stored procedure result set by using the SQL ALLOCATE CURSOR statement.

**ambiguous cursor**

A database cursor for which DB2 cannot determine whether it is used for update or read-only purposes.

**APAR** See authorized program analysis report.

**APF** See authorized program facility.

**API** See application programming interface.

**APPL** A VTAM network definition statement that is used to define DB2 to VTAM as an application program that uses SNA LU 6.2 protocols.

**application**

A program or set of programs that performs a task; for example, a payroll application.

**application period**

A pair of columns with application-maintained values that indicates the period of time when a row is valid. See also application-period temporal table.

**application-period temporal table**

A table that includes an application period. See also application period and bitemporal table.

**application plan**

The control structure that is produced during the bind process. DB2 uses the application plan to process SQL statements that it encounters during statement execution.

**application process**

The unit to which resources and locks are allocated. An application process involves the execution of one or more programs.

**application programming interface (API)**

A functional interface that is supplied by the operating system or by a separately ordered licensed program that allows an application program that is written in a high-level language to use specific data or functions of the operating system or licensed program.

**application requester**

The component on a remote system that generates DRDA requests for data on behalf of an application.

**application server**

The target of a request from a remote application. In the DB2 environment, the application server function is provided by the distributed data facility and is used to access DB2 data from remote applications.

**archive log**

The portion of the DB2 log that contains log records that have been copied from the active log. See also active log.

**ASCII** An encoding scheme that is used to represent strings in many environments, typically on personal computers and workstations. Contrast with EBCDIC and Unicode.

**ASID** See address space identifier.

**attachment facility**

An interface between DB2 and TSO, IMS, CICS, or batch address spaces. An attachment facility allows application programs to access DB2.

**attribute**

A characteristic of an entity. For example, in database design, the phone number of an employee is an attribute of that employee.

**authorization ID**

A string that can be verified for connection to DB2 and to which a set of privileges is allowed. An authorization ID can represent an individual or an organizational group.

**authorized program analysis report (APAR)**

A report of a problem that is caused by a suspected defect in a current release of an IBM supplied program.

**authorized program facility (APF)**

A facility that allows an installation to identify system or user programs that can use sensitive system functions.

**automatic bind**

(More correctly *automatic rebind*.) A process by which SQL statements are bound automatically (without a user issuing a BIND command) when an application process begins execution and the bound application plan or package it requires is not valid.

**automatic query rewrite**

A process that examines an SQL statement that refers to one or more base tables or materialized query tables, and, if appropriate, rewrites the query so that it performs better.

**auxiliary index**

An index on an auxiliary table in which each index entry refers to a LOB or XML document.

**auxiliary table**

A table that contains columns outside the actual table in which they are defined. Auxiliary tables can contain either LOB or XML data.

**backout**

The process of undoing uncommitted changes that an application process made.

A backout is often performed in the event of a failure on the part of an application process, or as a result of a deadlock situation.

#### **backward log recovery**

The final phase of restart processing during which DB2 scans the log in a backward direction to apply UNDO log records for all aborted changes.

#### **base table**

A table that is created by the SQL CREATE TABLE statement and that holds persistent data. Contrast with clone table, materialized query table, result table, temporary table, and transition table.

#### **base table space**

A table space that contains base tables.

#### **basic row format**

A row format in which values for columns are stored in the row in the order in which the columns are defined by the CREATE TABLE statement. Contrast with reordered row format.

#### **basic sequential access method (BSAM)**

An access method for storing or retrieving data blocks in a continuous sequence, using either a sequential-access or a direct-access device.

#### **BEFORE trigger**

A trigger that is specified to be activated before a defined trigger event (an insert, an update, or a delete operation on the table that is specified in a trigger definition). Contrast with AFTER trigger and INSTEAD OF trigger.

#### **begin column**

In a system period or an application period, the column that indicates the beginning of the period. See also period.

#### **binary large object (BLOB)**

A binary string data type that contains a sequence of bytes that can range in size from 0 bytes to 2 GB, less 1 byte. This string does not have an associated code page and character set. BLOBs can contain, for example, image, audio, or video data. In general, BLOB values are used whenever a binary string might exceed the limits of the VARBINARY type.

#### **binary string**

A sequence of bytes that is not associated with a CCSID. Binary string data type can be further classified as BINARY, VARBINARY, or BLOB.

#### **binary XML format**

A representation of XML data that uses binary values, an approach that facilitates more efficient storage and exchange.

#### **bind**

A process by which a usable control structure with SQL statements is generated; the structure is often called an access plan, an application plan, or a package. During this bind process, access paths to the data are selected, and some authorization checking is performed. See also automatic bind.

#### **bit data**

- Data with character type CHAR or VARCHAR that is defined with the FOR BIT DATA clause. Note that using BINARY or VARBINARY rather than FOR BIT DATA is highly recommended.
- Data with character type CHAR or VARCHAR that is defined with the FOR BIT DATA clause.
- A form of character data. Binary data is generally more highly recommended than character-for-bit data.

#### **bitemporal table**

A table that is both a system-period temporal table and an application-period temporal table. See also application-period temporal table and system-period temporal table.

**BLOB** See binary large object.

#### **block fetch**

A capability in which DB2 can retrieve, or fetch, a large set of rows together. Using block fetch can significantly reduce the number of messages that are being sent across the network. Block fetch applies only to non-rowset cursors that do not update data.

#### **bootstrap data set (BSDS)**

A VSAM data set that contains name and status information for DB2 and RBA range specifications for all active and archive log data sets. The BSDS also contains passwords for the DB2 directory



and catalog, and lists of conditional restart and checkpoint records.

**BSAM**

See basic sequential access method.

**BSDS** See bootstrap data set.

**buffer pool**

An area of memory into which data pages are read, modified, and held during processing.

**built-in data type**

A data type that IBM supplies. Among the built-in data types for DB2 for z/OS are string, numeric, XML, ROWID, and datetime. Contrast with distinct type.

**built-in function**

A function that is generated by DB2 and that is in the SYSIBM schema. Contrast with user-defined function. See also function, cast function, external function, sourced function, and SQL function.

**business dimension**

A category of data, such as products or time periods, that an organization might want to analyze.

**cache structure**

A coupling facility structure that stores data that can be available to all members of a Sysplex. A DB2 data sharing group uses cache structures as group buffer pools.

**CAF** See call attachment facility.

**call attachment facility (CAF)**

A DB2 attachment facility for application programs that run in TSO or z/OS batch. The CAF is an alternative to the DSN command processor and provides greater control over the execution environment. Contrast with Resource Recovery Services attachment facility.

**call-level interface (CLI)**

A callable application programming interface (API) for database access, which is an alternative to using embedded SQL.

**cascade delete**

A process by which DB2 enforces referential constraints by deleting all descendent rows of a deleted parent row.

**CASE expression**

An expression that is selected based on the evaluation of one or more conditions.

**cast function**

A function that is used to convert instances of a (source) data type into instances of a different (target) data type.

**castout**

The DB2 process of writing changed pages from a group buffer pool to disk.

**castout owner**

The DB2 member that is responsible for casting out a particular page set or partition.

**catalog**

In DB2, a collection of tables that contains descriptions of objects such as tables, views, and indexes.

**catalog table**

Any table in the DB2 catalog.

**CCSID**

See coded character set identifier.

**CDB**

See communications database.

**CDRA**

See Character Data Representation Architecture.

**CEC**

See central processor complex.

**central electronic complex (CEC)**

See central processor complex.

**central processor complex (CPC)**

A physical collection of hardware that consists of main storage, one or more central processors, timers, and channels.

**central processor (CP)**

The part of the computer that contains the sequencing and processing facilities for instruction execution, initial program load, and other machine operations.

**CFRM** See coupling facility resource management.

**CFRM policy**

The allocation rules for a coupling facility structure that are declared by a z/OS administrator.

**character conversion**

The process of changing characters from one encoding scheme to another.

**Character Data Representation Architecture (CDRA)**

An architecture that is used to achieve

consistent representation, processing, and interchange of string data.

**character large object (CLOB)**

A character string data type that contains a sequence of bytes that represent characters (single-byte, multibyte, or both) that can range in size from 0 bytes to 2 GB, less 1 byte. In general, CLOB values are used whenever a character string might exceed the limits of the VARCHAR type.

**character set**

A defined set of characters.

**character string**

A sequence of bytes that represent bit data, single-byte characters, or a mixture of single-byte and multibyte characters. Character data can be further classified as CHARACTER, VARCHAR, or CLOB.

**check constraint**

A user-defined constraint that specifies the values that specific columns of a base table can contain.

**check integrity**

The condition that exists when each row in a table conforms to the check constraints that are defined on that table.

**check pending**

A state of a table space or partition that prevents its use by some utilities and by some SQL statements because of rows that violate referential constraints, check constraints, or both.

**checkpoint**

A point at which DB2 records status information on the DB2 log; the recovery process uses this information if DB2 abnormally terminates.

**child lock**

For explicit hierarchical locking, a lock that is held on either a table, page, row, or a large object (LOB). Each child lock has a parent lock. See also parent lock.

**CI** See control interval.

**CICS** Represents (in this information): CICS Transaction Server for z/OS: Customer Information Control System Transaction Server for z/OS.

**CICS attachment facility**

A facility that provides a multithread

connection to DB2 to allow applications that run in the CICS environment to execute DB2 statements.

**claim** A notification to DB2 that an object is being accessed. Claims prevent drains from occurring until the claim is released, which usually occurs at a commit point. Contrast with drain.

**claim class**

A specific type of object access that can be one of the following isolation levels:

- Cursor stability (CS)
- Repeatable read (RR)
- Write

**class of service**

A VTAM term for a list of routes through a network, arranged in an order of preference for their use.

**clause** In SQL, a distinct part of a statement, such as a SELECT clause or a WHERE clause.

**CLI** See call-level interface.

**client** See requester.

**CLOB** See character large object.

**clone object**

An object that is associated with a clone table, including the clone table itself and check constraints, indexes, and BEFORE triggers on the clone table.

**clone table**

A table that is structurally identical to a base table. The base and clone table each have separate underlying VSAM data sets, which are identified by their data set instance numbers. Contrast with base table.

**closed application**

An application that requires exclusive use of certain statements on certain DB2 objects, so that the objects are managed solely through the external interface of that application.

**clustering index**

An index that determines how rows are physically ordered (*clustered*) in a table space. If a clustering index on a partitioned table is not a partitioning index, the rows are ordered in cluster sequence within each data partition instead of spanning partitions.



| **CM** See conversion mode.

| **CM\*** See conversion mode\*.

**C++ member**

A data object or function in a structure, union, or class.

**C++ member function**

An operator or function that is declared as a member of a class. A member function has access to the private and protected data members and to the member functions of objects in its class. Member functions are also called methods.

**C++ object**

A region of storage. An object is created when a variable is defined or a new function is invoked.

An instance of a class.

**coded character set**

A set of unambiguous rules that establish a character set and the one-to-one relationships between the characters of the set and their coded representations.

**coded character set identifier (CCSID)**

A 16-bit number that uniquely identifies a coded representation of graphic characters. It designates an encoding scheme identifier and one or more pairs that consist of a character set identifier and an associated code page identifier.

**code page**

A set of assignments of characters to code points. Within a code page, each code point has only one specific meaning. In EBCDIC, for example, the character *A* is assigned code point X'C1', and character *B* is assigned code point X'C2'.

**code point**

In CDRA, a unique bit pattern that represents a character in a code page.

**code unit**

The fundamental binary width in a computer architecture that is used for representing character data, such as 7 bits, 8 bits, 16 bits, or 32 bits. Depending on the character encoding form that is used, each code point in a coded character set can be represented by one or more code units.

**coexistence**

During migration, the period of time in which two releases exist in the same data sharing group.

**cold start**

A process by which DB2 restarts without processing any log records. Contrast with warm start.

**collection**

A group of packages that have the same qualifier.

**column**

The vertical component of a table. A column has a name and a particular data type (for example, character, decimal, or integer).

**column function**

See aggregate function.

**"come from" checking**

An LU 6.2 security option that defines a list of authorization IDs that are allowed to connect to DB2 from a partner LU.

**command**

A DB2 operator command or a DSN subcommand. A command is distinct from an SQL statement.

**command prefix**

A 1- to 8-character command identifier. The command prefix distinguishes the command as belonging to an application or subsystem rather than to z/OS.

**command recognition character (CRC)**

A character that permits a z/OS console operator or an IMS subsystem user to route DB2 commands to specific DB2 subsystems.

**command scope**

The scope of command operation in a data sharing group.

**commit**

The operation that ends a unit of work by releasing locks so that the database changes that are made by that unit of work can be perceived by other processes. Contrast with rollback.

**commit point**

A point in time when data is considered consistent.

**common service area (CSA)**

| In z/OS, a part of the common area that

contains data areas that are addressable by all address spaces. Most DB2 use is in the extended CSA, which is above the 16-MB line.

**communications database (CDB)**

A set of tables in the DB2 catalog that are used to establish conversations with remote database management systems.

**comparison operator**

A token (such as =, >, or <) that is used to specify a relationship between two values.

**compatibility mode**

See conversion mode.

**compatibility mode\* (CM\*)**

See conversion mode\*.

**composite key**

An ordered set of key columns or expressions of the same table.

**compression dictionary**

The dictionary that controls the process of compression and decompression. This dictionary is created from the data in the table space or table space partition.

**concurrency**

The shared use of resources by more than one application process at the same time.

**conditional restart**

A DB2 restart that is directed by a user-defined conditional restart control record (CRCR).

**connection**

In SNA, the existence of a communication path between two partner LUs that allows information to be exchanged (for example, two DB2 subsystems that are connected and communicating by way of a conversation).

**connection context**

In SQLJ, a Java object that represents a connection to a data source.

**connection declaration clause**

In SQLJ, a statement that declares a connection to a data source.

**connection handle**

The data object containing information that is associated with a connection that DB2 ODBC manages. This includes general status information, transaction status, and diagnostic information.

**connection ID**

An identifier that is supplied by the attachment facility and that is associated with a specific address space connection.

**consistency token**

A timestamp that is used to generate the version identifier for an application. See also version.

**constant**

A language element that specifies an unchanging value. Constants are classified as string constants or numeric constants. Contrast with variable.

**constraint**

A rule that limits the values that can be inserted, deleted, or updated in a table. See referential constraint, check constraint, and unique constraint.

**context**

An application's logical connection to the data source and associated DB2 ODBC connection information that allows the application to direct its operations to a data source. A DB2 ODBC context represents a DB2 thread.

**contracting conversion**

A process that occurs when the length of a converted string is smaller than that of the source string. For example, this process occurs when an EBCDIC mixed-data string that contains DBCS characters is converted to ASCII mixed data; the converted string is shorter because the shift codes are removed.

**control interval (CI)**

- A unit of information that VSAM transfers between virtual and auxiliary storage.
- In a key-sequenced data set or file, the set of records that an entry in the sequence-set index record points to.

**conversation**

Communication, which is based on LU 6.2 or Advanced Program-to-Program Communication (APPC), between an application and a remote transaction program over an SNA logical unit-to-logical unit (LU-LU) session that allows communication while processing a transaction.

<b>conversion mode* (CM*)</b>	A stage of the version-to-version migration process that applies to a DB2 subsystem or data sharing group that was in enabling-new-function mode (ENFM), enabling-new-function mode* (ENFM*), or new-function mode (NFM) at one time. Fallback to a prior version is not supported. When in conversion mode*, a DB2 data sharing group cannot coexist with members that are still at the prior version level. Contrast with conversion mode, enabling-new-function mode, enabling-new-function mode*, and new-function mode.	<b>copy version</b>	A point-in-time FlashCopy copy that is managed by HSM. Each copy pool has a version parameter that specifies the number of copy versions to be maintained on disk.
Previously known as compatibility mode* (CM*).		<b>correlated columns</b>	A relationship between the value of one column and the value of another column.
<b>conversion mode (CM)</b>	The first stage of the version-to-version migration process. In a DB2 data sharing group, members in conversion mode can coexist with members that are still at the prior version level. Fallback to the prior version is also supported. When in conversion mode, the DB2 subsystem cannot use most new functions of the new version. Contrast with conversion mode*, enabling-new-function mode, enabling-new-function mode*, and new-function mode.	<b>correlated subquery</b>	A subquery (part of a WHERE or HAVING clause) that is applied to a row or group of rows of a table or view that is named in an outer subselect statement.
Previously known as compatibility mode (CM).		<b>correlation ID</b>	An identifier that is associated with a specific thread. In TSO, it is either an authorization ID or the job name.
<b>coordinator</b>	The system component that coordinates the commit or rollback of a unit of work that includes work that is done on one or more other systems.	<b>correlation name</b>	An identifier that is specified and used within a single SQL statement as the exposed name for objects such as a table, view, table function reference, nested table expression, or result of a data change statement. Correlation names are useful in an SQL statement to allow two distinct references to the same base table and to allow an alternative name to be used to represent an object.
<b>coprocessor</b>	See SQL statement coprocessor.	<b>cost category</b>	A category into which DB2 places cost estimates for SQL statements at the time the statement is bound. The cost category is externalized in the COST_CATEGORY column of the DSN_STATEMENT_TABLE when a statement is explained.
<b>copy pool</b>	A collection of names of storage groups that are processed collectively for fast replication operations.	<b>coupling facility</b>	A special PR/SM logical partition (LPAR) that runs the coupling facility control program and provides high-speed caching, list processing, and locking functions in a Parallel Sysplex.
<b>copy target</b>	A named set of SMS storage groups that are to be used as containers for copy pool volume copies. A copy target is an SMS construct that lets you define which storage groups are to be used as containers for volumes that are copied by using FlashCopy functions.	<b>coupling facility resource management (CFRM)</b>	A component of z/OS that provides the services to manage coupling facility resources in a Parallel Sysplex. This management includes the enforcement of CFRM policies to ensure that the coupling facility and structure requirements are satisfied.
		<b>CP</b>	See central processor.

**CPC** See central processor complex.

**CRC** See command recognition character.

**created temporary table**

A persistent table that holds temporary data and is defined with the SQL statement CREATE GLOBAL TEMPORARY TABLE. Information about created temporary tables is stored in the DB2 catalog and can be shared across application processes. Contrast with declared temporary table. See also temporary table.

**cross-system coupling facility (XCF)**

A component of z/OS that provides functions to support cooperation between authorized programs that run within a Sysplex.

**cross-system extended services (XES)**

A set of z/OS services that allow multiple instances of an application or subsystem, running on different systems in a Sysplex environment, to implement high-performance, high-availability data sharing by using a coupling facility.

**CS** See cursor stability.

**CSA** See common service area.

**CT** See cursor table.

**current data**

Data within a host structure that is current with (identical to) the data within the base table.

**current status rebuild**

The second phase of restart processing during which the status of the subsystem is reconstructed from information on the log.

**cursor** A control structure that an application program uses to point to a single row or multiple rows within some ordered set of rows of a result table. A cursor can be used to retrieve, update, or delete rows from a result table.

**cursor sensitivity**

The degree to which database updates are visible to the subsequent FETCH statements in a cursor.

**cursor stability (CS)**

The isolation level that provides maximum concurrency without the ability to read uncommitted data. With cursor

stability, a unit of work holds locks only on its uncommitted changes and on the current row of each of its cursors. See also read stability, repeatable read, and uncommitted read.

**cursor table (CT)**

The internal representation of a cursor.

**cycle** A set of tables that can be ordered so that each table is a descendent of the one before it, and the first table is a descendent of the last table. A self-referencing table is a cycle with a single member. See also referential cycle.

**database**

A collection of tables, or a collection of table spaces and index spaces.

**database access thread (DBAT)**

A thread that accesses data at the local subsystem on behalf of a remote subsystem.

**database administrator (DBA)**

An individual who is responsible for designing, developing, operating, safeguarding, maintaining, and using a database.

**database alias**

The name of the target server if it is different from the location name. The database alias is used to provide the name of the database server as it is known to the network.

**database descriptor (DBD)**

An internal representation of a DB2 database definition, which reflects the data definition that is in the DB2 catalog. The objects that are defined in a database descriptor are table spaces, tables, indexes, index spaces, relationships, check constraints, and triggers. A DBD also contains information about accessing tables in the database.

**database exception status**

In a data sharing environment, an indication that something is wrong with a database.

**database identifier (DBID)**

An internal identifier of the database.

**database management system (DBMS)**

A software system that controls the

creation, organization, and modification of a database and the access to the data that is stored within it.

**database request module (DBRM)**

A data set member that is created by the DB2 precompiler and that contains information about SQL statements. DBRMs are used in the bind process.

**database server**

The target of a request from a local application or a remote intermediate database server.

**data currency**

The state in which the data that is retrieved into a host variable in a program is a copy of the data in the base table.

| **data-dependent pagination**

| The process used when applications need  
| to access part of a DB2 result set that is  
| based on a logical key value.

**data dictionary**

A repository of information about an organization's application programs, databases, logical data models, users, and authorizations.

**data partition**

A VSAM data set that is contained within a partitioned table space.

**data-partitioned secondary index (DPSI)**

A secondary index that is partitioned according to the underlying data. Contrast with nonpartitioned secondary index.

| **data set instance number**

| A number that indicates the data set that  
| contains the data for an object.

**data sharing**

A function of DB2 for z/OS that enables applications on different DB2 subsystems to read from and write to the same data concurrently.

**data sharing group**

A collection of one or more DB2 subsystems that directly access and change the same data while maintaining data integrity.

**data sharing member**

A DB2 subsystem that is assigned by XCF services to a data sharing group.

**data source**

A local or remote relational or non-relational data manager that is capable of supporting data access via an ODBC driver that supports the ODBC APIs. In the case of DB2 for z/OS, the data sources are always relational database managers.

**data type**

An attribute of columns, constants, variables, parameters, special registers, and the results of functions and expressions.

**data warehouse**

A system that provides critical business information to an organization. The data warehouse system cleanses the data for accuracy and currency, and then presents the data to decision makers so that they can interpret and use it effectively and efficiently.

**DBA** See database administrator.

**DBAT** See database access thread.

**DB2 catalog**

A collection of tables that are maintained by DB2 and contain descriptions of DB2 objects, such as tables, views, and indexes.

**DBCLOB**

See double-byte character large object.

**DB2 command**

An instruction to the DB2 subsystem that a user enters to start or stop DB2, to display information on current users, to start or stop databases, to display information on the status of databases, and so on.

**DBCS** See double-byte character set.

**DBD** See database descriptor.

**DB2I** See DB2 Interactive.

**DBID** See database identifier.

**DB2 Interactive (DB2I)**

An interactive service within DB2 that facilitates the execution of SQL statements, DB2 (operator) commands, and programmer commands, and the invocation of utilities.

**DBMS**

See database management system.



**DBRM**

See database request module.

**DB2 thread**

The database manager structure that describes an application's connection, traces its progress, processes resource functions, and delimits its accessibility to the database manager resources and services. Most DB2 for z/OS functions execute under a thread structure.

**DCLGEN**

See declarations generator.

**DDF** See distributed data facility.

**deadlock**

Unresolved contention for the use of a resource, such as a table or an index.

**declarations generator (DCLGEN)**

A subcomponent of DB2 that generates SQL table declarations and COBOL, C, or PL/I data structure declarations that conform to the table. The declarations are generated from DB2 system catalog information.

**declared temporary table**

A non-persistent table that holds temporary data and is defined with the SQL statement DECLARE GLOBAL TEMPORARY TABLE. Information about declared temporary tables is not stored in the DB2 catalog and can be used only by the application process that issued the DECLARE statement. Contrast with created temporary table. See also temporary table.

**default value**

A predetermined value, attribute, or option that is assumed when no other value is specified. A default value can be defined for column data in DB2 tables by specifying the DEFAULT keyword in an SQL statement that changes data (such as INSERT, UPDATE, and MERGE).

**deferred embedded SQL**

SQL statements that are neither fully static nor fully dynamic. These statements are embedded within an application and are prepared during the execution of the application.

**deferred write**

The process of asynchronously writing changed data pages to disk.

**degree of parallelism**

The number of concurrently executed operations that are initiated to process a query.

**delete hole**

The location on which a cursor is positioned when a row in a result table is refetched and the row no longer exists on the base table. See also update hole.

**delete rule**

The rule that tells DB2 what to do to a dependent row when a parent row is deleted. Delete rules include CASCADE, RESTRICT, SET NULL, or NO ACTION.

**delete trigger**

A trigger that is defined with the triggering delete SQL operation.

**delimited identifier**

A sequence of one or more characters enclosed by escape characters, such as quotation marks ("").

**delimiter token**

A string constant, a delimited identifier, an operator symbol, or any of the special characters that are shown in DB2 syntax diagrams.

**denormalization**

The intentional duplication of columns in multiple tables to increase data redundancy. Denormalization is sometimes necessary to minimize performance problems. Contrast with normalization.

**dependent**

An object (row, table, or table space) that has at least one parent. The object is also said to be a dependent (row, table, or table space) of its parent. See also parent row, parent table, and parent table space.

**dependent row**

A row that contains a foreign key that matches the value of a primary key in the parent row.

**dependent table**

A table that is a dependent in at least one referential constraint.

**descendent**

An object that is a dependent of an object or is the dependent of a descendent of an object.

**descendent row**

A row that is dependent on another row, or a row that is a descendent of a dependent row.

**descendent table**

A table that is a dependent of another table, or a table that is a descendent of a dependent table.

**deterministic function**

A user-defined function whose result is dependent on the values of the input arguments. That is, successive invocations with the same input values produce the same answer. Sometimes referred to as a *not-variant* function. Contrast with nondeterministic function (sometimes called a *variant function*).

**dimension**

A data category such as time, products, or markets. The elements of a dimension are referred to as members. See also dimension table.

**dimension table**

The representation of a dimension in a star schema. Each row in a dimension table represents all of the attributes for a particular member of the dimension. See also dimension, star schema, and star join.

**directory**

The DB2 system database that contains internal objects such as database descriptors and skeleton cursor tables.

**disk** A direct-access storage device that records data magnetically.

**distinct type**

A user-defined data type that is represented as an existing type (its source type), but is considered to be a separate and incompatible type for semantic purposes.

**distributed data**

Data that resides on a DBMS other than the local system.

**distributed data facility (DDF)**

A set of DB2 components through which DB2 communicates with another relational database management system.

**Distributed Relational Database Architecture (DRDA)**

A connection protocol for distributed relational database processing that is used

by IBM relational database products. DRDA includes protocols for communication between an application and a remote relational database management system, and for communication between relational database management systems. See also DRDA access.

**DNS** See domain name server.

**DOCID**

See document ID.

**document ID**

A value that uniquely identifies a row that contains an XML column. This value is stored with the row and never changes.

**domain**

The set of valid values for an attribute.

**domain name**

The name by which TCP/IP applications refer to a TCP/IP host within a TCP/IP network.

**domain name server (DNS)**

A special TCP/IP network server that manages a distributed directory that is used to map TCP/IP host names to IP addresses.

**double-byte character large object (DBCLOB)**

A graphic string data type in which a sequence of bytes represent double-byte characters that range in size from 0 bytes to 2 GB, less 1 byte. In general, DBCLOB values are used whenever a double-byte character string might exceed the limits of the VARCHAR type.

**double-byte character set (DBCS)**

A set of characters, which are used by national languages such as Japanese and Chinese, that have more symbols than can be represented by a single byte. Each character is 2 bytes in length. Contrast with single-byte character set and multibyte character set.

**double-precision floating point number**

A 64-bit approximate representation of a real number.

**DPSI** See data-partitioned secondary index.

**drain** The act of acquiring a locked resource by quiescing access to that object. Contrast with claim.

**drain lock**

A lock on a claim class that prevents a claim from occurring.

**DRDA**

See Distributed Relational Database Architecture.

**DRDA access**

An open method of accessing distributed data that you can use to connect to another database server to execute packages that were previously bound at the server location.

**DSN**

- The default DB2 subsystem name.
- The name of the TSO command processor of DB2.
- The first three characters of DB2 module and macro names.

**dynamic cursor**

A named control structure that an application program uses to change the size of the result table and the order of its rows after the cursor is opened. Contrast with static cursor.

**dynamic dump**

A dump that is issued during the execution of a program, usually under the control of that program.

**dynamic SQL**

SQL statements that are prepared and executed at run time. In dynamic SQL, the SQL statement is contained as a character string in a host variable or as a constant, and it is not precompiled.

**EA-enabled table space**

A table space or index space that is enabled for extended addressability and that contains individual partitions (or pieces, for LOB table spaces) that are greater than 4 GB.

**EB**

See exabyte.

**EBCDIC**

Extended binary coded decimal interchange code. An encoding scheme that is used to represent character data in the z/OS, VM, VSE, and iSeries environments. Contrast with ASCII and Unicode.

**embedded SQL**

SQL statements that are coded within an application program. See static SQL.

**enabling-new-function mode\* (ENFM\*)**

A transitional stage of the version-to-version migration process that applies to a DB2 subsystem or data sharing group that was in new-function mode (NFM) at one time. When in enabling-new-function mode\*, a DB2 subsystem or data sharing group is preparing to use the new functions of the new version but cannot yet use them. A data sharing group that is in enabling-new-function mode\* cannot coexist with members that are still at the prior version level. Fallback to a prior version is not supported. Contrast with conversion mode, conversion mode\*, enabling-new-function mode, and new-function mode.

**enabling-new-function mode (ENFM)**

A transitional stage of the version-to-version migration process during which the DB2 subsystem or data sharing group is preparing to use the new functions of the new version. When in enabling-new-function mode, a DB2 data sharing group cannot coexist with members that are still at the prior version level. Fallback to a prior version is not supported, and most new functions of the new version are not available for use in enabling-new-function mode. Contrast with conversion mode, conversion mode\*, enabling-new-function mode\*, and new-function mode.

**enclave**

In Language Environment, an independent collection of routines, one of which is designated as the main routine. An enclave is similar to a program or run unit. See also WLM enclave.

**encoding scheme**

A set of rules to represent character data (ASCII, EBCDIC, or Unicode).

**end column**

In a system period or an application period, the column that indicates the end of the period. See also period.

**ENFM** See enabling-new-function mode.



**ENFM\***

See enabling-new-function mode\*.

**entity** A person, object, or concept about which information is stored. In a relational database, entities are represented as tables. A database includes information about the entities in an organization or business, and their relationships to each other.

**enumerated list**

A set of DB2 objects that are defined with a LISTDEF utility control statement in which pattern-matching characters (\*, %;, \_ or ?) are not used.

**environment**

A collection of names of logical and physical resources that are used to support the performance of a function.

**environment handle**

A handle that identifies the global context for database access. All data that is pertinent to all objects in the environment is associated with this handle.

**equijoin**

A join operation in which the join-condition has the form *expression = expression*. See also join, full outer join, inner join, left outer join, outer join, and right outer join.

**error page range**

A range of pages that are considered to be physically damaged. DB2 does not allow users to access any pages that fall within this range.

**escape character**

The symbol, a double quotation (") for example, that is used to enclose an SQL delimited identifier.

**exabyte**

A unit of measure for processor, real and virtual storage capacities, and channel volume that has a value of 1 152 921 504 606 846 976 bytes or 2<sup>60</sup>.

**exception**

An SQL operation that involves the EXCEPT set operator, which combines two result tables. The result of an exception operation consists of all of the rows that are in only one of the result tables.

**exception table**

A table that holds rows that violate referential constraints or check constraints that the CHECK DATA utility finds.

**exclusive lock**

A lock that prevents concurrently executing application processes from reading or changing data. Contrast with share lock.

**executable statement**

An SQL statement that can be embedded in an application program, dynamically prepared and executed, or issued interactively.

**execution context**

In SQLJ, a Java object that can be used to control the execution of SQL statements.

**exit routine**

A user-written (or IBM-provided default) program that receives control from DB2 to perform specific functions. Exit routines run as extensions of DB2.

**expanding conversion**

A process that occurs when the length of a converted string is greater than that of the source string. For example, this process occurs when an ASCII mixed-data string that contains DBCS characters is converted to an EBCDIC mixed-data string; the converted string is longer because shift codes are added.

**explicit hierarchical locking**

Locking that is used to make the parent-child relationship between resources known to IRLM. This kind of locking avoids global locking overhead when no inter-DB2 interest exists on a resource.

**explicit privilege**

A privilege that has a name and is held as the result of an SQL GRANT statement and revoked as the result of an SQL REVOKE statement. For example, the SELECT privilege.

**exposed name**

A correlation name or a table or view name for which a correlation name is not specified.

**expression**

An operand or a collection of operators and operands that yields a single value.

**Extended Recovery Facility (XRF)**

A facility that minimizes the effect of failures in z/OS, VTAM, the host processor, or high-availability applications during sessions between high-availability applications and designated terminals. This facility provides an alternative subsystem to take over sessions from the failing subsystem.

**Extensible Markup Language (XML)**

A standard metalanguage for defining markup languages that is a subset of Standardized General Markup Language (SGML).

**external function**

A function that has its functional logic implemented in a programming language application that resides outside the database, in the file system of the database server. The association of the function with the external code application is specified by the EXTERNAL clause in the CREATE FUNCTION statement. External functions can be classified as external scalar functions and external table functions. Contrast with sourced function, built-in function, and SQL function.

**external procedure**

A procedure that has its procedural logic implemented in an external programming language application. The association of the procedure with the external application is specified by a CREATE PROCEDURE statement with a LANGUAGE clause that has a value other than SQL and an EXTERNAL clause that implicitly or explicitly specifies the name of the external application. Contrast with external SQL procedure and native SQL procedure.

**external routine**

A user-defined function or stored procedure that is based on code that is written in an external programming language.

**external SQL procedure**

An SQL procedure that is processed using a generated C program that is a representation of the procedure. When an external SQL procedure is called, the C program representation of the procedure is executed in a stored procedures address

space. Contrast with external procedure and native SQL procedure.

**failed member state**

A state of a member of a data sharing group in which the member's task, address space, or z/OS system terminates before the state changes from active to quiesced.

**fallback**

The process of returning to a previous release of DB2 after attempting or completing migration to a current release. Fallback is supported only from a subsystem that is in conversion mode.

**false global lock contention**

A contention indication from the coupling facility that occurs when multiple lock names are hashed to the same indicator and when no real contention exists.

**fan set**

A direct physical access path to data, which is provided by an index, hash, or link; a fan set is the means by which DB2 supports the ordering of data.

**federated database**

The combination of a DB2 server (in Linux, UNIX, and Windows environments) and multiple data sources to which the server sends queries. In a federated database system, a client application can use a single SQL statement to join data that is distributed across multiple database management systems and can view the data as if it were local.

**fetch orientation**

The specification of the desired placement of the cursor as part of a FETCH statement. The specification can be before or after the rows of the result table (with BEFORE or AFTER). The specification can also have either a single-row fetch orientation (for example, NEXT, LAST, or ABSOLUTE *n*) or a rowset fetch orientation (for example, NEXT ROWSET, LAST ROWSET, or ROWSET STARTING AT ABSOLUTE *n*).

**field procedure**

A user-written exit routine that is designed to receive a single value and transform (encode or decode) it in any way the user can specify.

| **file reference variable**  
| A host variable that is declared with one  
| of the derived data types (BLOB\_FILE,  
| CLOB\_FILE, DBCLOB\_FILE); file  
| reference variables direct the reading of a  
| LOB from a file or the writing of a LOB  
| into a file.

**filter factor**  
A number between zero and one that  
estimates the proportion of rows in a  
table for which a predicate is true.

| **fixed-length string**  
| A character, graphic, or binary string  
| whose length is specified and cannot be  
| changed. Contrast with varying-length  
| string.

| **FlashCopy**  
| A function on the IBM Enterprise Storage  
| Server that can, in conjunction with the  
| BACKUP SYSTEM utility, create a  
| point-in-time copy of data while an  
| application is running.

**foreign key**  
A column or set of columns in a  
dependent table of a constraint  
relationship. The key must have the same  
number of columns, with the same  
descriptions, as the primary key of the  
parent table. Each foreign key value must  
either match a parent key value in the  
related parent table or be null.

**forest** An ordered set of subtrees of XML nodes.

**forward log recovery**  
The third phase of restart processing  
during which DB2 processes the log in a  
forward direction to apply all REDO log  
records.

**free space**  
The total amount of unused space in a  
page; that is, the space that is not used to  
store records or control information is free  
space.

**full outer join**  
The result of a join operation that  
includes the matched rows of both tables  
that are being joined and preserves the  
unmatched rows of both tables. See also  
join, equijoin, inner join, left outer join,  
outer join, and right outer join.

| **fullselect**  
| A subselect, a fullselect in parentheses, or

a number of both that are combined by  
set operators. Fullselect specifies a result  
table. If a set operator is not used, the  
result of the fullselect is the result of the  
specified subselect or fullselect.

**fully escaped mapping**  
A mapping from an SQL identifier to an  
XML name when the SQL identifier is a  
column name.

**function**  
A mapping, which is embodied as a  
program (the function body) that can be  
invoked by means of zero or more input  
values (arguments) to a single value (the  
result). See also aggregate function and  
scalar function.

Functions can be user-defined, built-in, or  
generated by DB2. (See also built-in  
function, cast function, external function,  
sourced function, SQL function, and  
user-defined function.)

**function definer**  
The authorization ID of the owner of the  
schema of the function that is specified in  
the CREATE FUNCTION statement.

**function package**  
A package that results from binding the  
DBRM for a function program.

**function package owner**  
The authorization ID of the user who  
binds the function program's DBRM into  
a function package.

**function signature**  
The logical concatenation of a fully  
qualified function name with the data  
types of all of its parameters.

**GB** Gigabyte. A value of (1 073 741 824 bytes).

**GBP** See group buffer pool.

**GBP-dependent**  
The status of a page set or page set  
partition that is dependent on the group  
buffer pool. Either read/write interest is  
active among DB2 subsystems for this  
page set, or the page set has changed  
pages in the group buffer pool that have  
not yet been cast out to disk.

**generalized trace facility (GTF)**  
A z/OS service program that records  
significant system events such as I/O

interrupts, SVC interrupts, program interrupts, or external interrupts.

**generated column**

A column for which the database manager assigns the value. An example of a generated column is an identity column, row change timestamp column, or row-begin column. See also generated expression column.

**generated expression column**

A generated column that is defined using an expression. See also generated column.

**generic resource name**

A name that VTAM uses to represent several application programs that provide the same function in order to handle session distribution and balancing in a Sysplex environment.

**getpage**

An operation in which DB2 accesses a data page.

**global lock**

A lock that provides concurrency control within and among DB2 subsystems. The scope of the lock is across all DB2 subsystems of a data sharing group.

**global lock contention**

Conflicts on locking requests between different DB2 members of a data sharing group when those members are trying to serialize shared resources.

**governor**

See resource limit facility.

**graphic string**

A sequence of DBCS characters. Graphic data can be further classified as GRAPHIC, VARGRAPHIC, or DBCLOB.

**GRECP**

See group buffer pool recovery pending.

**gross lock**

The *shared*, *update*, or *exclusive* mode locks on a table, partition, or table space.

**group buffer pool duplexing**

The ability to write data to two instances of a group buffer pool structure: a primary group buffer pool and a secondary group buffer pool. z/OS publications refer to these instances as the “old” (for primary) and “new” (for secondary) structures.

**group buffer pool (GBP)**

A coupling facility cache structure that is used by a data sharing group to cache data and to ensure that the data is consistent for all members.

**group buffer pool recovery pending (GRECP)**

The state that exists after the buffer pool for a data sharing group is lost. When a page set is in this state, changes that are recorded in the log must be applied to the affected page set before the page set can be used.

**group level**

The release level of a data sharing group, which is established when the first member migrates to a new release.

**group name**

The z/OS XCF identifier for a data sharing group.

**group restart**

A restart of at least one member of a data sharing group after the loss of either locks or the shared communications area.

**GTF** See generalized trace facility.

**handle**

In DB2 ODBC, a variable that refers to a data structure and associated resources. See also statement handle, connection handle, and environment handle.

**hash access**

Access to a table using the hash value of a key that is defined by the *organization-clause* of a CREATE TABLE statement or ALTER TABLE statement.

**hash overflow index**

A DB2 index used to track data rows that do not fit into the fixed hash space, and therefore, reside in the hash overflow space. DB2 accesses the hash overflow index to fetch rows from the hash overflow area.

**help panel**

A screen of information that presents tutorial text to assist a user at the workstation or terminal.

**heuristic damage**

The inconsistency in data between one or more participants that results when a heuristic decision to resolve an indoubt

LUW at one or more participants differs from the decision that is recorded at the coordinator.

#### **heuristic decision**

A decision that forces indoubt resolution at a participant by means other than automatic resynchronization between coordinator and participant.

#### **histogram statistics**

A way of summarizing data distribution. This technique divides up the range of possible values in a data set into intervals, such that each interval contains approximately the same percentage of the values. A set of statistics are collected for each interval.

#### **historical row**

A row in a history table. See also history table.

#### **history table**

A table that is used by the database manager to store historical versions of the rows from the associated system-period temporal table. See also historical row and system-period temporal table.

#### **hole**

A row of the result table that cannot be accessed because of a delete or an update that has been performed on the row. See also delete hole and update hole.

#### **home address space**

The area of storage that z/OS currently recognizes as *dispatched*.

#### **host**

The set of programs and resources that are available on a given TCP/IP instance.

#### **host expression**

A Java variable or expression that is referenced by SQL clauses in an SQLJ application program.

#### **host identifier**

A name that is declared in the host program.

#### **host language**

A programming language in which you can embed SQL statements.

#### **host program**

An application program that is written in a host language and that contains embedded SQL statements.

#### **host structure**

In an application program, a structure that is referenced by embedded SQL statements.

#### **host variable**

In an application program written in a host language, an application variable that is referenced by embedded SQL statements.

#### **host variable array**

An array of elements, each of which corresponds to a value for a column. The dimension of the array determines the maximum number of rows for which the array can be used.

#### **IBM System z9 Integrated Processor (zIIP)**

A specialized processor that can be used for some DB2 functions.

#### **IDCAMS**

An IBM program that is used to process access method services commands. It can be invoked as a job or jobstep, from a TSO terminal, or from within a user's application program.

#### **IDCAMS LISTCAT**

A facility for obtaining information that is contained in the access method services catalog.

#### **identity column**

A generated column that is defined with the AS IDENTITY clause. An identity column provides a way for the database manager to automatically generate a numeric value for each row that is inserted into a table. A table can have no more than one identity column.

**IFCID** See instrumentation facility component identifier.

**IFI** See instrumentation facility interface.

#### **IFI call**

An invocation of the instrumentation facility interface (IFI) by means of one of its defined functions.

#### **image copy**

An exact reproduction of all or part of a table space. DB2 provides utility programs to make full image copies (to copy the entire table space) or incremental image copies (to copy only those pages that have been modified since the last image copy).



**IMS attachment facility**

A DB2 subcomponent that uses z/OS subsystem interface (SSI) protocols and cross-memory linkage to process requests from IMS to DB2 and to coordinate resource commitment.

**in-abort**

A status of a unit of recovery. If DB2 fails after a unit of recovery begins to be rolled back, but before the process is completed, DB2 continues to back out the changes during restart.

**in-commit**

A status of a unit of recovery. If DB2 fails after beginning its phase 2 commit processing, it "knows," when restarted, that changes made to data are consistent. Such units of recovery are termed *in-commit*.

**independent**

An object (row, table, or table space) that is neither a parent nor a dependent of another object.

**index** A set of pointers that are logically ordered by the values of a key. Indexes can provide faster access to data and can enforce uniqueness on the rows in a table.

**index-controlled partitioning**

A type of partitioning in which partition boundaries for a partitioned table are controlled by values that are specified on the CREATE INDEX statement. Partition limits are saved in the LIMITKEY column of the SYSIBM.SYSINDEXPART catalog table.

**index key**

The set of columns in a table that is used to determine the order of index entries.

**index partition**

A VSAM data set that is contained within a partitioning index space.

**index space**

A page set that is used to store the entries of one index.

**indicator column**

A 4-byte value that is stored in a base table in place of a LOB column.

**indicator variable**

A variable that is used to represent the null value in an application program. If

the value for the selected column is null, a negative value is placed in the indicator variable.

**indoubt**

A status of a unit of recovery. If DB2 fails after it has finished its phase 1 commit processing and before it has started phase 2, only the commit coordinator knows if an individual unit of recovery is to be committed or rolled back. At restart, if DB2 lacks the information it needs to make this decision, the status of the unit of recovery is *indoubt* until DB2 obtains this information from the coordinator. More than one unit of recovery can be *indoubt* at restart.

**indoubt resolution**

The process of resolving the status of an *indoubt* logical unit of work to either the committed or the rollback state.

**inflight**

A status of a unit of recovery. If DB2 fails before its unit of recovery completes phase 1 of the commit process, it merely backs out the updates of its unit of recovery at restart. These units of recovery are termed *inflight*.

**inheritance**

The passing downstream of class resources or attributes from a parent class in the class hierarchy to a child class.

**initialization file**

For DB2 ODBC applications, a file containing values that can be set to adjust the performance of the database manager.

**inline copy**

A copy that is produced by the LOAD or REORG utility. The data set that the inline copy produces is logically equivalent to a full image copy that is produced by running the COPY utility with read-only access (SHRLEVEL REFERENCE).

**inline SQL PL**

A subset of SQL procedural language that can be used in SQL functions and dynamic compound statements. See also SQL procedural language.

**inner join**

The result of a join operation that includes only the matched rows of both tables that are being joined. See also join,

equijoin, full outer join, left outer join, outer join, and right outer join.

**inoperative package**

In DB2 Version 9.1 for z/OS and earlier, a package that cannot be used because one or more user-defined functions or procedures that the package depends on were dropped. Such a package must be explicitly rebound. Contrast with invalid package.

**insensitive cursor**

A cursor that is not sensitive to inserts, updates, or deletes that are made to the underlying rows of a result table after the result table has been materialized.

**insert trigger**

A trigger that is defined with the triggering SQL operation, an insert.

**install** The process of preparing a DB2 subsystem to operate as a z/OS subsystem.

**INSTEAD OF trigger**

A trigger that is associated with a single view and is activated by an insert, update, or delete operation on the view and that can define how to propagate the insert, update, or delete operation on the view to the underlying tables of the view. Contrast with BEFORE trigger and AFTER trigger.

**instrumentation facility component identifier (IFCID)**

A value that names and identifies a trace record of an event that can be traced. As a parameter on the START TRACE and MODIFY TRACE commands, it specifies that the corresponding event is to be traced.

**instrumentation facility interface (IFI)**

A programming interface that enables programs to obtain online trace data about DB2, to submit DB2 commands, and to pass data to DB2.

**Interactive System Productivity Facility (ISPF)**

An IBM licensed program that provides interactive dialog services in a z/OS environment.

**inter-DB2 R/W interest**

A property of data in a table space, index, or partition that has been opened by more than one member of a data sharing group

and that has been opened for writing by at least one of those members.

**intermediate database server**

The target of a request from a local application or a remote application requester that is forwarded to another database server.

**internal resource lock manager (IRLM)**

A z/OS subsystem that DB2 uses to control communication and database locking.

**intersection**

An SQL operation that involves the INTERSECT set operator, which combines two result tables. The result of an intersection operation consists of all of the rows that are in both result tables.

**invalid package**

In DB2 Version 9.1 for z/OS and earlier, a package that depends on an object (other than a user-defined function) that is dropped. Such a package is implicitly rebound on invocation. Contrast with inoperative package.

**IP address**

A value that uniquely identifies a TCP/IP host.

**IRLM** See internal resource lock manager.

**isolation level**

The degree to which a unit of work is isolated from the updating operations of other units of work. See also cursor stability, read stability, repeatable read, and uncommitted read.

**ISPF** See Interactive System Productivity Facility.

**iterator**

In SQLJ, an object that contains the result set of a query. An iterator is equivalent to a cursor in other host languages.

**iterator declaration clause**

In SQLJ, a statement that generates an iterator declaration class. An iterator is an object of an iterator declaration class.

**JAR** See Java Archive.

**Java Archive (JAR)**

A file format that is used for aggregating many files into a single file.

**JDBC** A Sun Microsystems database application



programming interface (API) for Java that allows programs to access database management systems by using callable SQL.

**join** A relational operation that allows retrieval of data from two or more tables based on matching column values. See also equijoin, full outer join, inner join, left outer join, outer join, and right outer join.

**KB** Kilobyte. A value of 1024 bytes.

**Kerberos**

A network authentication protocol that is designed to provide strong authentication for client/server applications by using secret-key cryptography.

**Kerberos ticket**

A transparent application mechanism that transmits the identity of an initiating principal to its target. A simple ticket contains the principal's identity, a session key, a timestamp, and other information, which is sealed using the target's secret key.

**key** A column, an ordered collection of columns, or an expression that is identified in the description of a table, index, or referential constraint. The same column or expression can be part of more than one key.

**key-sequenced data set (KSDS)**

A VSAM file or data set whose records are loaded in key sequence and controlled by an index.

**KSDS** See key-sequenced data set.

**large object (LOB)**

A sequence of bytes representing bit data, single-byte characters, double-byte characters, or a mixture of single- and double-byte characters. A LOB can be up to 2 GB minus 1 byte in length. See also binary large object, character large object, and double-byte character large object.

**last agent optimization**

An optimized commit flow for either presumed-nothing or presumed-abort protocols in which the last agent, or final participant, becomes the commit coordinator. This flow saves at least one message.

**latch** A DB2 mechanism for controlling concurrent events or the use of system resources.

**LCID** See log control interval definition.

**LDS** See linear data set.

**leaf page**

An index page that contains pairs of keys and RIDs and that points to actual data. Contrast with nonleaf page.

**left outer join**

The result of a join operation that includes the matched rows of both tables that are being joined, and that preserves the unmatched rows of the first table. See also join, equijoin, full outer join, inner join, outer join, and right outer join.

**limit key**

The highest value of the index key for a partition.

**linear data set (LDS)**

A VSAM data set that contains data but no control information. A linear data set can be accessed as a byte-addressable string in virtual storage.

**linkage editor**

A computer program for creating load modules from one or more object modules or load modules by resolving cross references among the modules and, if necessary, adjusting addresses.

**link-edit**

The action of creating a loadable computer program using a linkage editor.

**list**

A type of object, which DB2 utilities can process, that identifies multiple table spaces, multiple index spaces, or both. A list is defined with the LISTDEF utility control statement.

**list structure**

A coupling facility structure that lets data be shared and manipulated as elements of a queue.

**L-lock** See logical lock.

**load module**

A program unit that is suitable for loading into main storage for execution. The output of a linkage editor.

**LOB** See large object.

**LOB locator**

A mechanism that allows an application program to manipulate a large object value in the database system. A LOB locator is a fullword integer value that represents a single LOB value. An application program retrieves a LOB locator into a host variable and can then apply SQL operations to the associated LOB value using the locator.

**LOB lock**

A lock on a LOB value.

**LOB table space**

A table space that contains all the data for a particular LOB column in the related base table.

**local** A way of referring to any object that the local DB2 subsystem maintains. A *local table*, for example, is a table that is maintained by the local DB2 subsystem. Contrast with remote.

**locale** The definition of a subset of a user's environment that combines a CCSID and characters that are defined for a specific language and country.

**local lock**

A lock that provides intra-DB2 concurrency control, but not inter-DB2 concurrency control; that is, its scope is a single DB2.

**local subsystem**

The unique relational DBMS to which the user or application program is directly connected (in the case of DB2, by one of the DB2 attachment facilities).

**location**

The unique name of a database server. An application uses the location name to access a DB2 database server. A database alias can be used to override the location name when accessing a remote server.

**location alias**

Another name by which a database server identifies itself in the network. Applications can use this name to access a DB2 database server.

**lock** A means of controlling concurrent events or access to data. DB2 locking is performed by the IRLM.

**lock duration**

The interval over which a DB2 lock is held.

**lock escalation**

The promotion of a lock from a row, page, or LOB lock to a table space lock because the number of page locks that are concurrently held on a given resource exceeds a preset limit.

**locking**

The process by which the integrity of data is ensured. Locking prevents concurrent users from accessing inconsistent data. See also claim, drain, and latch.

**lock mode**

A representation for the type of access that concurrently running programs can have to a resource that a DB2 lock is holding.

**lock object**

The resource that is controlled by a DB2 lock.

**lock promotion**

The process of changing the size or mode of a DB2 lock to a higher, more restrictive level.

**lock size**

The amount of data that is controlled by a DB2 lock on table data; the value can be a row, a page, a LOB, a partition, a table, or a table space.

**lock structure**

A coupling facility data structure that is composed of a series of lock entries to support shared and exclusive locking for logical resources.

**log**

A collection of records that describe the events that occur during DB2 execution and that indicate their sequence. The information thus recorded is used for recovery in the event of a failure during DB2 execution.

**log control interval definition**

A suffix of the physical log record that tells how record segments are placed in the physical control interval.

**logical claim**

A claim on a logical partition of a nonpartitioning index.

**logical index partition**

The set of all keys that reference the same data partition.

**logical lock (L-lock)**

The lock type that transactions use to control intra- and inter-DB2 data concurrency between transactions. Contrast with physical lock (P-lock).

**logically complete**

A state in which the concurrent copy process is finished with the initialization of the target objects that are being copied. The target objects are available for update.

**logical page list (LPL)**

A list of pages that are in error and that cannot be referenced by applications until the pages are recovered. The page is in *logical error* because the actual media (coupling facility or disk) might not contain any errors. Usually a connection to the media has been lost.

**logical partition**

A set of key or RID pairs in a nonpartitioning index that are associated with a particular partition.

**logical recovery pending (LRECP)**

The state in which the data and the index keys that reference the data are inconsistent.

**logical unit (LU)**

An access point through which an application program accesses the SNA network in order to communicate with another application program. See also LU name.

**logical unit of work**

The processing that a program performs between synchronization points.

**logical unit of work identifier (LUWID)**

A name that uniquely identifies a thread within a network. This name consists of a fully-qualified LU network name, an LUW instance number, and an LUW sequence number.

**log initialization**

The first phase of restart processing during which DB2 attempts to locate the current end of the log.

**log record header (LRH)**

A prefix, in every log record, that contains control information.

**log record sequence number (LRSN)**

An identifier for a log record that is associated with a data sharing member. DB2 uses the LRSN for recovery in the data sharing environment.

**log truncation**

A process by which an explicit starting RBA is established. This RBA is the point at which the next byte of log data is to be written.

**LPL** See logical page list.

**LRECP**

See logical recovery pending.

**LRH** See log record header.

**LRSN** See log record sequence number.

**LU** See logical unit.

**LU name**

Logical unit name, which is the name by which VTAM refers to a node in a network.

**LUW** See logical unit of work.

**LUWID**

See logical unit of work identifier.

**mapping table**

A table that the REORG utility uses to map the associations of the RIDs of data records in the original copy and in the shadow copy. This table is created by the user.

**mass delete**

The deletion of all rows of a table.

**materialize**

- The process of putting rows from a view or nested table expression into a work file for additional processing by a query.
- The placement of a LOB value into contiguous storage. Because LOB values can be very large, DB2 avoids materializing LOB data until doing so becomes absolutely necessary.

**materialized query table**

A table that is used to contain information

that is derived and can be summarized from one or more source tables. Contrast with base table.

**MB** Megabyte (1 048 576 bytes).

**MBCS** See multibyte character set.

**member name**

The z/OS XCF identifier for a particular DB2 subsystem in a data sharing group.

**menu** A displayed list of available functions for selection by the operator. A menu is sometimes called a *menu panel*.

**metalanguage**

A language that is used to create other specialized languages.

**migration**

The process of converting a subsystem with a previous release of DB2 to an updated or current release. In this process, you can acquire the functions of the updated or current release without losing the data that you created on the previous release.

**mixed data string**

A character string that can contain both single-byte and double-byte characters.

**mode name**

A VTAM name for the collection of physical and logical characteristics and attributes of a session.

**modify locks**

An L-lock or P-lock with a MODIFY attribute. A list of these active locks is kept at all times in the coupling facility lock structure. If the requesting DB2 subsystem fails, that DB2 subsystem's modify locks are converted to retained locks.

**multibyte character set (MBCS)**

A character set that represents single characters with more than a single byte. UTF-8 is an example of an MBCS. Characters in UTF-8 can range from 1 to 4 bytes in DB2. Contrast with single-byte character set and double-byte character set. See also Unicode.

**multidimensional analysis**

The process of assessing and evaluating an enterprise on more than one level.

**Multiple Virtual Storage (MVS)**

An element of the z/OS operating system.

This element is also called the Base Control Program (BCP).

**multisite update**

Distributed relational database processing in which data is updated in more than one location within a single unit of work.

**multithreading**

Multiple TCBs that are executing one copy of DB2 ODBC code concurrently (sharing a processor) or in parallel (on separate central processors).

**MVS** See Multiple Virtual Storage.

**native SQL procedure**

An SQL procedure that is processed by converting the procedural statements to a native representation that is stored in the database directory, as is done with other SQL statements. When a native SQL procedure is called, the native representation is loaded from the directory, and DB2 executes the procedure. Contrast with external procedure and external SQL procedure.

**nested table expression**

A fullselect in a FROM clause (surrounded by parentheses).

**network identifier (NID)**

The network ID that is assigned by IMS or CICS, or if the connection type is RRSAF, the RRS unit of recovery ID (URID).

**new-function mode (NFM)**

The normal mode of operation that exists after successful completion of a version-to-version migration. At this stage, all new functions of the new version are available for use. A DB2 data sharing group cannot coexist with members that are still at the prior version level, and fallback to a prior version is not supported. Contrast with conversion mode, conversion mode\*, enabling-new-function mode, and enabling-new-function mode\*.

**NFM** See new-function mode.

**NID** See network identifier.

**node ID index**

See XML node ID index.

**nondeterministic function**

A user-defined function whose result is

not solely dependent on the values of the input arguments. That is, successive invocations with the same argument values can produce a different answer. This type of function is sometimes called a *variant* function. Contrast with deterministic function (sometimes called a *not-variant function*).

#### **nonleaf page**

A page that contains keys and page numbers of other pages in the index (either leaf or nonleaf pages). Nonleaf pages never point to actual data. Contrast with leaf page.

#### **nonpartitioned index**

An index that is not physically partitioned. Both partitioning indexes and secondary indexes can be nonpartitioned.

#### **nonpartitioned secondary index (NPSI)**

An index on a partitioned table space that is not the partitioning index and is not partitioned. Contrast with data-partitioned secondary index.

#### **nonpartitioning index**

See secondary index.

#### **nonscrollable cursor**

A cursor that can be moved only in a forward direction. Nonscrollable cursors are sometimes called forward-only cursors or serial cursors.

#### **normalization**

A key step in the task of building a logical relational database design. Normalization helps you avoid redundancies and inconsistencies in your data. An entity is normalized if it meets a set of constraints for a particular normal form (first normal form, second normal form, and so on). Contrast with denormalization.

#### **not-variant function**

See deterministic function.

**NPSI** See nonpartitioned secondary index.

**NUL** The null character ('\0'), which is represented by the value X'00'. In C, this character denotes the end of a string.

**null** A special value that indicates the absence of information.

#### **null terminator**

In C, the value that indicates the end of a

string. For EBCDIC, ASCII, and Unicode UTF-8 strings, the null terminator is a single-byte value (X'00'). For Unicode UTF-16 or UCS-2 (wide) strings, the null terminator is a double-byte value (X'0000').

#### **ODBC**

See Open Database Connectivity.

#### **ODBC driver**

A dynamically-linked library (DLL) that implements ODBC function calls and interacts with a data source.

**OLAP** See online analytical processing.

#### **online analytical processing (OLAP)**

The process of collecting data from one or many sources; transforming and analyzing the consolidated data quickly and interactively; and examining the results across different dimensions of the data by looking for patterns, trends, and exceptions within complex relationships of that data.

#### **Open Database Connectivity (ODBC)**

A Microsoft database application programming interface (API) for C that allows access to database management systems by using callable SQL. ODBC does not require the use of an SQL preprocessor. In addition, ODBC provides an architecture that lets users add modules called *database drivers*, which link the application to their choice of database management systems at run time. This means that applications no longer need to be directly linked to the modules of all the database management systems that are supported.

#### **ordinary identifier**

An uppercase letter followed by zero or more characters, each of which is an uppercase letter, a digit, or the underscore character. An ordinary identifier must not be a reserved word.

#### **ordinary token**

A numeric constant, an ordinary identifier, a host identifier, or a keyword.

#### **originating task**

In a parallel group, the primary agent that receives data from other execution units (referred to as *parallel tasks*) that are executing portions of the query in parallel.



**outer join**

The result of a join operation that includes the matched rows of both tables that are being joined and preserves some or all of the unmatched rows of the tables that are being joined. See also join, equijoin, full outer join, inner join, left outer join, and right outer join.

**overloaded function**

A function name for which multiple function instances exist.

**package**

An object containing a set of SQL statements that have been statically bound and that is available for processing. A package is sometimes also called an *application package*.

**package list**

An ordered list of package names that may be used to extend an application plan.

**package name**

The name of an object that is used for an application package or an SQL procedure package. An application package is a bound version of a database request module (DBRM) that is created by a BIND PACKAGE or REBIND PACKAGE command. An SQL procedural language package is created by a CREATE or ALTER PROCEDURE statement for a native SQL procedure. The name of a package consists of a location name, a collection ID, a package ID, and a version ID.

**page**

A unit of storage within a table space (4 KB, 8 KB, 16 KB, or 32 KB) or index space (4 KB, 8 KB, 16 KB, or 32 KB). In a table space, a page contains one or more rows of a table. In a LOB or XML table space, a LOB or XML value can span more than one page, but no more than one LOB or XML value is stored on a page.

**page set**

Another way to refer to a table space or index space. Each page set consists of a collection of VSAM data sets.

**page set recovery pending (PSRCP)**

A restrictive state of an index space. In this case, the entire page set must be recovered. Recovery of a logical part is prohibited.

**panel**

A predefined display image that defines the locations and characteristics of display fields on a display surface (for example, a *menu panel*).

**parallel complex**

A cluster of machines that work together to handle multiple transactions and applications.

**parallel group**

A set of consecutive operations that execute in parallel and that have the same number of parallel tasks.

**parallel I/O processing**

A form of I/O processing in which DB2 initiates multiple concurrent requests for a single user query and performs I/O processing concurrently (in *parallel*) on multiple data partitions.

**parallelism assistant**

In Sysplex query parallelism, a DB2 subsystem that helps to process parts of a parallel query that originates on another DB2 subsystem in the data sharing group.

**parallelism coordinator**

In Sysplex query parallelism, the DB2 subsystem from which the parallel query originates.

**Parallel Sysplex**

A set of z/OS systems that communicate and cooperate with each other through certain multisystem hardware components and software services to process customer workloads.

**parallel task**

The execution unit that is dynamically created to process a query in parallel. A parallel task is implemented by a z/OS service request block.

**parameter marker**

A question mark (?) that appears in a statement string of a dynamic SQL statement. The question mark can appear where a variable could appear if the statement string were a static SQL statement.

**parameter-name**

An SQL identifier that designates a parameter in a routine that is written by a user. Parameter names are required for SQL procedures and SQL functions, and they are used in the body of the routine

	to refer to the values of the parameters.		Each partition can contain a program,
	Parameter names are optional for external		part of a program, or data. A program
	routines.		library is an example of a partitioned data
			set.
	<b>parent key</b>		<b>partitioned index</b>
	A primary key or unique key in the		An index that is physically partitioned.
	parent table of a referential constraint.		Both partitioning indexes and secondary
	The values of a parent key determine the		indexes can be partitioned.
	valid values of the foreign key in the		
	referential constraint.		
	<b>parent lock</b>		<b>partitioned page set</b>
	For explicit hierarchical locking, a lock		A partitioned table space or an index
	that is held on a resource that might have		space. Header pages, space map pages,
	child locks that are lower in the hierarchy.		data pages, and index pages reference
	A parent lock is usually the table space		data only within the scope of the
	lock or the partition intent lock. See also		partition.
	child lock.		
	<b>parent row</b>		<b>partitioned table space</b>
	A row whose primary key value is the		A table space that is based on a single
	foreign key value of a dependent row.		table and that is subdivided into
			partitions, each of which can be processed
	<b>parent table</b>		independently by utilities. Contrast with
	A table whose primary key is referenced		segmented table space and universal table
	by the foreign key of a dependent table.		space.
	<b>parent table space</b>		<b>partitioning index</b>
	A table space that contains a parent table.		An index in which the leftmost columns
	A table space containing a dependent of		are the partitioning columns of the table.
	that table is a dependent table space.		The index can be partitioned or
			nonpartitioned.
	<b>participant</b>		<b>partner logical unit</b>
	An entity other than the commit		An access point in the SNA network that
	coordinator that takes part in the commit		is connected to the local DB2 subsystem
	process. The term participant is		by way of a VTAM conversation.
	synonymous with agent in SNA.		
	<b>partition</b>		<b>path</b>
	A portion of a page set. Each partition		See SQL path.
	corresponds to a single, independently		<b>PDS</b>
	extendable data set. The maximum size of		See partitioned data set.
	a partition depends on the number of		<b>period</b>
	partitions in the partitioned page set. All		In a table, an interval of time that is
	partitions of a given page set have the		defined by two datetime columns. A
	same maximum size.		period contains a begin column and an
			end column. See also begin column and
			end column.
	<b>partition-by-growth table space</b>		<b>physical consistency</b>
	A table space whose size can grow to		The state of a page that is not in a
	accommodate data growth. DB2 for z/OS		partially changed state.
	manages partition-by-growth table spaces		<b>physical lock (P-lock)</b>
	by automatically adding new data sets		A type of lock that DB2 acquires to
	when the database needs more space to		provide consistency of data that is cached
	satisfy an insert operation. Contrast with		in different DB2 subsystems. Physical
	range-partitioned table space. See also		locks are used only in data sharing
	universal table space.		environments. Contrast with logical lock
			(L-lock).
	<b>partitioned data set (PDS)</b>		
	A data set in disk storage that is divided		
	into partitions, which are called members.		



**physically complete**

The state in which the concurrent copy process is completed and the output data set has been created.

**piece** A data set of a nonpartitioned page set.

**plan** See application plan.

**plan allocation**

The process of allocating DB2 resources to a plan in preparation for execution.

**plan member**

The bound copy of a DBRM that is identified in the member clause.

**plan name**

The name of an application plan.

**P-lock** See physical lock.

**point of consistency**

A time when all recoverable data that an application accesses is consistent with other data. The term point of consistency is synonymous with sync point or commit point.

**policy** See CFRM policy.

**postponed abort UR**

A unit of recovery that was inflight or in-abort, was interrupted by system failure or cancellation, and did not complete backout during restart.

**precision**

In SQL, the total number of digits in a decimal number (called the *size* in the C language). In the C language, the number of digits to the right of the decimal point (called the *scale* in SQL). The DB2 information uses the SQL terms.

**precompilation**

A processing of application programs containing SQL statements that takes place before compilation. SQL statements are replaced with statements that are recognized by the host language compiler. Output from this precompilation includes source code that can be submitted to the compiler and the database request module (DBRM) that is input to the bind process.

**predicate**

An element of a search condition that expresses or implies a comparison operation.

**prefix** A code at the beginning of a message or record.

**preformat**

The process of preparing a VSAM linear data set for DB2 use, by writing specific data patterns.

**prepare**

The first phase of a two-phase commit process in which all participants are requested to prepare for commit.

**prepared SQL statement**

A named object that is the executable form of an SQL statement that has been processed by the PREPARE statement.

**primary authorization ID**

The authorization ID that is used to identify the application process to DB2.

**primary group buffer pool**

For a duplexed group buffer pool, the structure that is used to maintain the coherency of cached data. This structure is used for page registration and cross-invalidation. The z/OS equivalent is *old* structure. Compare with secondary group buffer pool.

**primary index**

An index that enforces the uniqueness of a primary key.

**primary key**

In a relational database, a unique, nonnull key that is part of the definition of a table. A table cannot be defined as a parent unless it has a unique key or primary key.

**principal**

An entity that can communicate securely with another entity. In Kerberos, principals are represented as entries in the Kerberos registry database and include users, servers, computers, and others.

**principal name**

The name by which a principal is known to the DCE security services.

**privilege**

The capability of performing a specific function, sometimes on a specific object. See also explicit privilege.

**privilege set**

- For the installation SYSADM ID, the set of all possible privileges.

- For any other authorization ID, including the PUBLIC authorization ID, the set of all privileges that are recorded for that ID in the DB2 catalog.

#### **process**

In DB2, the unit to which DB2 allocates resources and locks. Sometimes called an application process, a process involves the execution of one or more programs. The execution of an SQL statement is always associated with some process. The means of initiating and terminating a process are dependent on the environment.

#### **program**

A single, compilable collection of executable statements in a programming language.

#### **program temporary fix (PTF)**

A solution or bypass of a problem that is diagnosed as a result of a defect in a current unaltered release of a licensed program. An authorized program analysis report (APAR) fix is corrective service for an existing problem. A PTF is preventive service for problems that might be encountered by other users of the product. A PTF is *temporary*, because a permanent fix is usually not incorporated into the product until its next release.

#### **protected conversation**

A VTAM conversation that supports two-phase commit flows.

#### **PSRCP**

See page set recovery pending.

**PTF** See program temporary fix.

#### **QSAM**

See queued sequential access method.

**query** A component of certain SQL statements that specifies a result table.

#### **query block**

The part of a query that is represented by one of the FROM clauses. Each FROM clause can have multiple query blocks, depending on DB2 processing of the query.

#### **query CP parallelism**

Parallel execution of a single query, which is accomplished by using multiple tasks. See also Sysplex query parallelism.

#### **query I/O parallelism**

Parallel access of data, which is accomplished by triggering multiple I/O requests within a single query.

#### **queued sequential access method (QSAM)**

An extended version of the basic sequential access method (BSAM). When this method is used, a queue of data blocks is formed. Input data blocks await processing, and output data blocks await transfer to auxiliary storage or to an output device.

#### **quiesce point**

A point at which data is consistent as a result of running the DB2 QUIESCE utility.

**RACF** Resource Access Control Facility. A component of the z/OS Security Server.

#### **range-partitioned table space**

A type of universal table space that is based on partitioning ranges and that contains a single table. Contrast with partition-by-growth table space. See also universal table space.

**RBA** See relative byte address.

**RCT** See resource control table.

**RDO** See resource definition online.

#### **read stability (RS)**

An isolation level that is similar to repeatable read but does not completely isolate an application process from all other concurrently executing application processes. See also cursor stability, repeatable read, and uncommitted read.

#### **rebind**

The creation of a new application plan for an application program that has been bound previously. If, for example, you have added an index for a table that your application accesses, you must rebind the application in order to take advantage of that index.

#### **rebuild**

The process of reallocating a coupling facility structure. For the shared communications area (SCA) and lock structure, the structure is repopulated; for the group buffer pool, changed pages are usually cast out to disk, and the new

structure is populated only with changed pages that were not successfully cast out.

**record** The storage representation of a row or other data.

**record identifier (RID)**

A unique identifier that DB2 uses to identify a row of data in a table. Compare with row identifier.

**record identifier (RID) pool**

An area of main storage that is used for sorting record identifiers during list-prefetch processing.

**record length**

The sum of the length of all the columns in a table, which is the length of the data as it is physically stored in the database. Records can be fixed length or varying length, depending on how the columns are defined. If all columns are fixed-length columns, the record is a fixed-length record. If one or more columns are varying-length columns, the record is a varying-length record.

**Resource Recovery Services attachment facility (RRSAF)**

A DB2 subcomponent that uses Resource Recovery Services to coordinate resource commitment between DB2 and all other resource managers that also use RRS in a z/OS system.

**recovery**

The process of rebuilding databases after a system failure.

**recovery log**

A collection of records that describes the events that occur during DB2 execution and indicates their sequence. The recorded information is used for recovery in the event of a failure during DB2 execution.

**recovery manager**

A subcomponent that supplies coordination services that control the interaction of DB2 resource managers during commit, abort, checkpoint, and restart processes. The recovery manager also supports the recovery mechanisms of other subsystems (for example, IMS) by acting as a participant in the other subsystem's process for protecting data that has reached a point of consistency.

A coordinator or a participant (or both), in the execution of a two-phase commit, that can access a recovery log that maintains the state of the logical unit of work and names the immediate upstream coordinator and downstream participants.

**recovery pending (RECP)**

A condition that prevents SQL access to a table space that needs to be recovered.

**recovery token**

An identifier for an element that is used in recovery (for example, NID or URID).

**RECP** See recovery pending.

**redo** A state of a unit of recovery that indicates that changes are to be reapplied to the disk media to ensure data integrity.

**reentrant code**

Executable code that can reside in storage as one shared copy for all threads. Reentrant code is not self-modifying and provides separate storage areas for each thread. See also threadsafe.

**referential constraint**

The requirement that nonnull values of a designated foreign key are valid only if they equal values of the primary key of a designated table.

**referential cycle**

A set of referential constraints such that each base table in the set is a descendent of itself. The tables that are involved in a referential cycle are ordered so that each table is a descendent of the one before it, and the first table is a descendent of the last table.

**referential integrity**

The state of a database in which all values of all foreign keys are valid. Maintaining referential integrity requires the enforcement of referential constraints on all operations that change the data in a table on which the referential constraints are defined.

**referential structure**

A set of tables and relationships that includes at least one table and, for every table in the set, all the relationships in which that table participates and all the tables to which it is related.

**refresh age**

The time duration between the current

time and the time during which a materialized query table was last refreshed.

**registry**

See registry database.

**registry database**

A database of security information about principals, groups, organizations, accounts, and security policies.

**relational database**

A database that can be perceived as a set of tables and manipulated in accordance with the relational model of data.

**relational database management system (RDBMS)**

A collection of hardware and software that organizes and provides access to a relational database.

**relational schema**

See SQL schema.

**relationship**

A defined connection between the rows of a table or the rows of two tables. A relationship is the internal representation of a referential constraint.

**relative byte address (RBA)**

The offset of a data record or control interval from the beginning of the storage space that is allocated to the data set or file to which it belongs.

**remigration**

The process of returning to a current release of DB2 following a fallback to a previous release. This procedure constitutes another migration process.

**remote**

Any object that is maintained by a remote DB2 subsystem (that is, by a DB2 subsystem other than the local one). A *remote view*, for example, is a view that is maintained by a remote DB2 subsystem. Contrast with local.

**remote subsystem**

Any relational DBMS, except the local subsystem, with which the user or application can communicate. The subsystem need not be remote in any physical sense, and might even operate on the same processor under the same z/OS system.

**reoptimization**

The DB2 process of reconsidering the access path of an SQL statement at run time; during reoptimization, DB2 uses the values of host variables, parameter markers, or special registers.

**reordered row format**

A row format that facilitates improved performance in retrieval of rows that have varying-length columns. DB2 rearranges the column order, as defined in the CREATE TABLE statement, so that the fixed-length columns are stored at the beginning of the row and the varying-length columns are stored at the end of the row. Contrast with basic row format.

**REORG pending (REORP)**

A condition that restricts SQL access and most utility access to an object that must be reorganized.

**REORP**

See REORG pending.

**repeatable read (RR)**

The isolation level that provides maximum protection from other executing application programs. When an application program executes with repeatable read protection, rows that the program references cannot be changed by other programs until the program reaches a commit point. See also cursor stability, read stability, and uncommitted read.

**repeating group**

A situation in which an entity includes multiple attributes that are inherently the same. The presence of a repeating group violates the requirement of first normal form. In an entity that satisfies the requirement of first normal form, each attribute is independent and unique in its meaning and its name. See also normalization.

**replay detection mechanism**

A method that allows a principal to detect whether a request is a valid request from a source that can be trusted or whether an untrustworthy entity has captured information from a previous exchange and is replaying the information exchange to gain access to the principal.

**request commit**

The vote that is submitted to the prepare phase if the participant has modified data and is prepared to commit or roll back.

**requester**

The source of a request to access data at a remote server. In the DB2 environment, the requester function is provided by the distributed data facility.

**resource**

The object of a lock or claim, which could be a table space, an index space, a data partition, an index partition, or a logical partition.

**resource allocation**

The part of plan allocation that deals specifically with the database resources.

**resource control table**

A construct of previous versions of the CICS attachment facility that defines authorization and access attributes for transactions or transaction groups. Beginning in CICS Transaction Server Version 1.3, resources are defined by using resource definition online instead of the resource control table. See also resource definition online.

**resource definition online (RDO)**

The recommended method of defining resources to CICS by creating resource definitions interactively, or by using a utility, and then storing them in the CICS definition data set. In earlier releases of CICS, resources were defined by using the resource control table (RCT), which is no longer supported.

**resource limit facility (RLF)**

A portion of DB2 code that prevents dynamic manipulative SQL statements from exceeding specified time limits. The resource limit facility is sometimes called the governor.

**resource limit specification table (RLST)**

A site-defined table that specifies the limits to be enforced by the resource limit facility.

**resource manager**

- A function that is responsible for managing a particular resource and that guarantees the consistency of all updates made to recoverable resources within a logical unit of work. The

resource that is being managed can be physical (for example, disk or main storage) or logical (for example, a particular type of system service).

- A participant, in the execution of a two-phase commit, that has recoverable resources that could have been modified. The resource manager has access to a recovery log so that it can commit or roll back the effects of the logical unit of work to the recoverable resources.

**restart pending (RESTP)**

A restrictive state of a page set or partition that indicates that restart (backout) work needs to be performed on the object.

**RESTP**

See restart pending.

**result set**

The set of rows that a stored procedure returns to a client application.

**result set locator**

A 4-byte value that DB2 uses to uniquely identify a query result set that a stored procedure returns.

**result table**

The set of rows that are specified by a SELECT statement.

**retained lock**

A MODIFY lock that a DB2 subsystem was holding at the time of a subsystem failure. The lock is retained in the coupling facility lock structure across a DB2 for z/OS failure.

**RID** See record identifier.

**RID pool**

See record identifier pool.

**right outer join**

The result of a join operation that includes the matched rows of both tables that are being joined and preserves the unmatched rows of the second join operand. See also join, equijoin, full outer join, inner join, left outer join, and outer join.

**RLF** See resource limit facility.

**RLST** See resource limit specification table.

**role** A database entity that groups together one or more privileges and that can be



	assigned to a primary authorization ID or to PUBLIC. The role is available only in a trusted context.		<b>rowid</b>	A value that uniquely identifies a row in a table and does not change.
	<b>rollback</b>		<b>row lock</b>	A lock on a single row of data.
	The process of restoring data that was changed by SQL statements to the state at its last commit point. All locks are freed. Contrast with commit.		<b>row-positioned fetch orientation</b>	The specification of the desired placement of the cursor as part of a FETCH statement, with respect to a single row (for example, NEXT, LAST, or ABSOLUTE <i>n</i> ). Contrast with rowset-positioned fetch orientation.
	<b>root page</b>		<b>rowset</b>	A set of rows for which a cursor position is established.
	The index page that is at the highest level (or the beginning point) in an index.		<b>rowset cursor</b>	A cursor that is defined so that one or more rows can be returned as a rowset for a single FETCH statement, and the cursor is positioned on the set of rows that is fetched.
	<b>routine</b>		<b>rowset-positioned fetch orientation</b>	The specification of the desired placement of the cursor as part of a FETCH statement, with respect to a rowset (for example, NEXT ROWSET, LAST ROWSET, or ROWSET STARTING AT ABSOLUTE <i>n</i> ). Contrast with row-positioned fetch orientation.
	A database object that encapsulates procedural logic and SQL statements, is stored on the database server, and can be invoked from an SQL statement or by using the CALL statement. The main classes of routines are procedures and functions.		<b>row trigger</b>	A trigger that is defined with the trigger granularity FOR EACH ROW.
	<b>row</b>		<b>RRSAF</b>	See Resource Recovery Services attachment facility.
	The horizontal component of a table. A row consists of a sequence of values, one for each column of the table.		<b>RS</b>	See read stability.
	<b>row-begin column</b>		<b>savepoint</b>	A named entity that represents the state of data and schemas at a particular point in time within a unit of work.
	A generated column that is defined with the AS ROW BEGIN clause. The value is assigned whenever a row is inserted into the table or any column in the row is updated. A row-begin column is intended for use as the first column of a SYSTEM_TIME period. See also generated column, row-end column, and transaction-start-ID column.		<b>SBCS</b>	See single-byte character set.
	<b>row change timestamp column</b>		<b>SCA</b>	See shared communications area.
	A generated column that is defined with the AS ROW CHANGE TIMESTAMP clause. A row change timestamp column provides a way for the database manager to automatically generate and maintain a timestamp value for each row that is inserted or updated in a table. A table can have no more than one row change timestamp column.		<b>scalar function</b>	An SQL operation that produces a single value from another value and is expressed as a function name, followed by a list of arguments that are enclosed in parentheses.
	<b>row-end column</b>		<b>scale</b>	In SQL, the number of digits to the right of the decimal point (called the precision
	A generated column that is defined with the AS ROW END clause. The value is assigned whenever a row is inserted into the table or any column in the row is updated. A row-end column is intended for use as the second column of a SYSTEM_TIME period. See also generated column, row-begin column, and transaction-start-ID column.			

in the C language). The DB2 information uses the SQL definition.

**schema**

The organization or structure of a database.

A collection of, and a way of qualifying, database objects such as tables, views, routines, indexes or triggers that define a database. A database schema provides a logical classification of database objects.

**scrollability**

The ability to use a cursor to fetch in either a forward or backward direction. The FETCH statement supports multiple fetch orientations to indicate the new position of the cursor. See also fetch orientation.

**scrollable cursor**

A cursor that can be moved in both a forward and a backward direction.

**search condition**

A criterion for selecting rows from a table. A search condition consists of one or more predicates.

**secondary authorization ID**

An authorization ID that has been associated with a primary authorization ID by an authorization exit routine.

**secondary group buffer pool**

For a duplexed group buffer pool, the structure that is used to back up changed pages that are written to the primary group buffer pool. No page registration or cross-invalidation occurs using the secondary group buffer pool. The z/OS equivalent is *new* structure.

**secondary index**

A nonpartitioning index that is useful for enforcing a uniqueness constraint, for clustering data, or for providing access paths to data for queries. A secondary index can be partitioned or nonpartitioned. See also data-partitioned secondary index (DPSI) and nonpartitioned secondary index (NPSI).

**section**

The segment of a plan or package that contains the executable structures for a single SQL statement. For most SQL statements, one section in the plan exists for each SQL statement in the source

program. However, for cursor-related statements, the DECLARE, OPEN, FETCH, and CLOSE statements reference the same section because they each refer to the SELECT statement that is named in the DECLARE CURSOR statement. SQL statements such as COMMIT, ROLLBACK, and some SET statements do not use a section.

**security label**

A classification of users' access to objects or data rows in a multilevel security environment."

**segment**

A group of pages that holds rows of a single table. See also segmented table space.

**segmented table space**

A table space that is divided into equal-sized groups of pages called segments. Segments are assigned to tables so that rows of different tables are never stored in the same segment. Contrast with partitioned table space and universal table space.

**self-referencing constraint**

A referential constraint that defines a relationship in which a table is a dependent of itself.

**self-referencing table**

A table with a self-referencing constraint.

**sensitive cursor**

A cursor that is sensitive to changes that are made to the database after the result table has been materialized.

**sequence**

A user-defined object that generates a sequence of numeric values according to user specifications.

**sequential data set**

A non-DB2 data set whose records are organized on the basis of their successive physical positions, such as on magnetic tape. Several of the DB2 database utilities require sequential data sets.

**sequential prefetch**

A mechanism that triggers consecutive asynchronous I/O operations. Pages are fetched before they are required, and several pages are read with a single I/O operation.



**serialized profile**

A Java object that contains SQL statements and descriptions of host variables. The SQLJ translator produces a serialized profile for each connection context.

**server** The target of a request from a remote requester. In the DB2 environment, the server function is provided by the distributed data facility, which is used to access DB2 data from remote applications.

**service class**

An eight-character identifier that is used by the z/OS Workload Manager to associate user performance goals with a particular DDF thread or stored procedure. A service class is also used to classify work on parallelism assistants.

**service request block**

A unit of work that is scheduled to execute.

**session**

A link between two nodes in a VTAM network.

**session protocols**

The available set of SNA communication requests and responses.

**set operator**

The SQL operators UNION, EXCEPT, and INTERSECT corresponding to the relational operators union, difference, and intersection. A set operator derives a result table by combining two other result tables.

**shared communications area (SCA)**

A coupling facility list structure that a DB2 data sharing group uses for inter-DB2 communication.

**share lock**

A lock that prevents concurrently executing application processes from changing data, but not from reading data. Contrast with exclusive lock.

**shift-in character**

A special control character (X'0F') that is used in EBCDIC systems to denote that the subsequent bytes represent SBCS characters. See also shift-out character.

**shift-out character**

A special control character (X'0E') that is used in EBCDIC systems to denote that

the subsequent bytes, up to the next shift-in control character, represent DBCS characters. See also shift-in character.

**sign-on**

A request that is made on behalf of an individual CICS or IMS application process by an attachment facility to enable DB2 to verify that it is authorized to use DB2 resources.

**simple page set**

A nonpartitioned page set. A simple page set initially consists of a single data set (page set piece). If and when that data set is extended to 2 GB, another data set is created, and so on, up to a total of 32 data sets. DB2 considers the data sets to be a single contiguous linear address space containing a maximum of 64 GB. Data is stored in the next available location within this address space without regard to any partitioning scheme.

**simple table space**

A table space that is neither partitioned nor segmented. Creation of simple table spaces is not supported in DB2 Version 9.1 for z/OS. Contrast with partitioned table space, segmented table space, and universal table space.

**single-byte character set (SBCS)**

A set of characters in which each character is represented by a single byte. Contrast with double-byte character set or multibyte character set.

**single-precision floating point number**

A 32-bit approximate representation of a real number.

**SMP/E**

See System Modification Program/Extended.

**SNA** See Systems Network Architecture.

**SNA network**

The part of a network that conforms to the formats and protocols of Systems Network Architecture (SNA).

**socket** A callable TCP/IP programming interface that TCP/IP network applications use to communicate with remote TCP/IP partners.

**sourced function**

A function that is implemented by another built-in or user-defined function

| that is already known to the database  
| manager. This function can be a scalar  
| function or an aggregate function; it  
| returns a single value from a set of values  
| (for example, MAX or AVG). Contrast  
| with built-in function, external function,  
| and SQL function.

**source program**

A set of host language statements and SQL statements that is processed by an SQL precompiler.

**source table**

A table that can be a base table, a view, a table expression, or a user-defined table function.

**source type**

An existing type that DB2 uses to represent a distinct type.

**space** A sequence of one or more blank characters.

**special register**

| A storage area that DB2 defines for an  
| application process to use for storing  
| information that can be referenced in SQL  
| statements. Examples of special registers  
| are SESSION\_USER and CURRENT  
| DATE.

**specific function name**

A particular user-defined function that is known to the database manager by its specific name. Many specific user-defined functions can have the same function name. When a user-defined function is defined to the database, every function is assigned a specific name that is unique within its schema. Either the user can provide this name, or a default name is used.

**SPUFI** See SQL Processor Using File Input.

**SQL** See Structured Query Language.

**SQL authorization ID (SQL ID)**

The authorization ID that is used for checking dynamic SQL statements in some situations.

**SQLCA**

See SQL communication area.

**SQL communication area (SQLCA)**

A structure that is used to provide an application program with information about the execution of its SQL statements.

**SQL connection**

An association between an application process and a local or remote application server or database server.

**SQLDA**

See SQL descriptor area.

**SQL descriptor area (SQLDA)**

A structure that describes input variables, output variables, or the columns of a result table.

**SQL escape character**

The symbol that is used to enclose an SQL delimited identifier. This symbol is the double quotation mark ("). See also escape character.

**SQL function**

| A user-defined function in which the  
| CREATE FUNCTION statement contains  
| the source code. The source code is a  
| single SQL expression that evaluates to a  
| single value. The SQL user-defined  
| function can return the result of an  
| expression. See also built-in function,  
| external function, and sourced function.

**SQL ID**

See SQL authorization ID.

**SQLJ** Structured Query Language (SQL) that is embedded in the Java programming language.

**SQL path**

An ordered list of schema names that are used in the resolution of unqualified references to user-defined functions, distinct types, and stored procedures. In dynamic SQL, the SQL path is found in the CURRENT PATH special register. In static SQL, it is defined in the PATH bind option.

**SQL PL**

See SQL procedural language.

**SQL procedural language (SQL PL)**

A language extension of SQL that consists of statements and language elements that can be used to implement procedural logic in SQL statements. SQL PL provides statements for declaring variables and condition handlers, assigning values to variables, and for implementing procedural logic. See also inline SQL PL.

**SQL procedure**

A user-written program that can be

invoked with the SQL CALL statement.  
An SQL procedure is written in the SQL procedural language. Two types of SQL procedures are supported: external SQL procedures and native SQL procedures. See also external procedure and native SQL procedure.

#### **SQL processing conversation**

Any conversation that requires access of DB2 data, either through an application or by dynamic query requests.

#### **SQL Processor Using File Input (SPUFI)**

A facility of the TSO attachment subcomponent that enables the DB2I user to execute SQL statements without embedding them in an application program.

#### **SQL return code**

Either SQLCODE or SQLSTATE.

#### **SQL routine**

A user-defined function or stored procedure that is based on code that is written in SQL.

#### **SQL schema**

A collection of database objects such as tables, views, indexes, functions, distinct types, schemas, or triggers that defines a database. An SQL schema provides a logical classification of database objects.

#### **SQL statement coprocessor**

An alternative to the DB2 precompiler that lets the user process SQL statements at compile time. The user invokes an SQL statement coprocessor by specifying a compiler option.

#### **SQL string delimiter**

A symbol that is used to enclose an SQL string constant. The SQL string delimiter is the apostrophe ('), except in COBOL applications, where the user assigns the symbol, which is either an apostrophe or a double quotation mark (").

**SRB** See service request block.

#### **stand-alone**

An attribute of a program that means that it is capable of executing separately from DB2, without using DB2 services.

#### **star join**

A method of joining a dimension column of a fact table to the key column of the

corresponding dimension table. See also join, dimension, and star schema.

#### **star schema**

The combination of a fact table (which contains most of the data) and a number of dimension tables. See also star join, dimension, and dimension table.

#### **statement handle**

In DB2 ODBC, the data object that contains information about an SQL statement that is managed by DB2 ODBC. This includes information such as dynamic arguments, bindings for dynamic arguments and columns, cursor information, result values, and status information. Each statement handle is associated with the connection handle.

#### **statement string**

For a dynamic SQL statement, the character string form of the statement.

#### **statement trigger**

A trigger that is defined with the trigger granularity FOR EACH STATEMENT.

#### **static cursor**

A named control structure that does not change the size of the result table or the order of its rows after an application opens the cursor. Contrast with dynamic cursor.

#### **static SQL**

SQL statements, embedded within a program, that are prepared during the program preparation process (before the program is executed). After being prepared, the SQL statement does not change (although values of variables that are specified by the statement might change).

#### **storage group**

A set of storage objects on which DB2 for z/OS data can be stored. A storage object can have an SMS data class, a management class, a storage class, and a list of volume serial numbers.

#### **stored procedure**

A user-written application program that can be invoked through the use of the SQL CALL statement. Stored procedures are sometimes called procedures.

**string** See binary string, character string, or graphic string.

**strong typing**

A process that guarantees that only user-defined functions and operations that are defined on a distinct type can be applied to that type. For example, you cannot directly compare two currency types, such as Canadian dollars and U.S. dollars. But you can provide a user-defined function to convert one currency to the other and then do the comparison.

**structure**

- A name that refers collectively to different types of DB2 objects, such as tables, databases, views, indexes, and table spaces.
- A construct that uses z/OS to map and manage storage on a coupling facility. See also cache structure, list structure, or lock structure.

**Structured Query Language (SQL)**

A standardized language for defining and manipulating data in a relational database.

**structure owner**

In relation to group buffer pools, the DB2 member that is responsible for the following activities:

- Coordinating rebuild, checkpoint, and damage assessment processing
- Monitoring the group buffer pool threshold and notifying castout owners when the threshold has been reached

**subcomponent**

A group of closely related DB2 modules that work together to provide a general function.

**subject table**

The table for which a trigger is created. When the defined triggering event occurs on this table, the trigger is activated.

**subquery**

A SELECT statement within the WHERE or HAVING clause of another SQL statement; a nested SQL statement.

**subselect**

That form of a query that includes only a SELECT clause, FROM clause, and optionally a WHERE clause, GROUP BY clause, HAVING clause, ORDER BY clause, or FETCH FIRST clause.

**substitution character**

A unique character that is substituted during character conversion for any characters in the source program that do not have a match in the target coding representation.

**subsystem**

In z/OS, a service provider that performs one or many functions, but does nothing until a request is made. For example, each WebSphere MQ for z/OS queue manager or instance of a DB2 for z/OS database management system is a z/OS subsystem.

**surrogate pair**

A coded representation for a single character that consists of a sequence of two 16-bit code units, in which the first value of the pair is a high-surrogate code unit in the range U+D800 through U+DBFF, and the second value is a low-surrogate code unit in the range U+DC00 through U+DFFF. Surrogate pairs provide an extension mechanism for encoding 917 476 characters without requiring the use of 32-bit characters.

**SVC dump**

A dump that is issued when a z/OS or a DB2 functional recovery routine detects an error.

**sync point**

See commit point.

**syncpoint tree**

The tree of recovery managers and resource managers that are involved in a logical unit of work, starting with the recovery manager, that make the final commit decision.

**synonym**

In SQL, an alternative name for a table or view. Synonyms can be used to refer only to objects at the subsystem in which the synonym is defined. A synonym cannot be qualified and can therefore not be used by other users. Contrast with alias.

**Sysplex**

See Parallel Sysplex.

**Sysplex query parallelism**

Parallel execution of a single query that is accomplished by using multiple tasks on more than one DB2 subsystem. See also query CP parallelism.

**system administrator**

The person at a computer installation who designs, controls, and manages the use of the computer system.

**system agent**

A work request that DB2 creates such as prefetch processing, deferred writes, and service tasks. See also allied agent.

**system authorization ID**

The primary DB2 authorization ID that is used to establish a trusted connection. A system authorization ID is derived from the system user ID that is provided by an external entity, such as a middleware server.

**system conversation**

The conversation that two DB2 subsystems must establish to process system messages before any distributed processing can begin.

**system-defined routine**

In DB2 10 for z/OS and later, an object (function or procedure) for which system DBADM and SQLADM authorities have implicit execute privilege on the routine and any packages executed within the routine.

**System Modification Program/Extended (SMP/E)**

A z/OS tool for making software changes in programming systems (such as DB2) and for controlling those changes.

**system period**

A pair of columns with system-maintained values that indicates the period of time when a row is valid. See also system-period temporal table and system-period data versioning.

**system-period data versioning**

Automatic maintenance of historical data by the database manager by using a system period. See also system period and system-period temporal table.

**system-period temporal table**

A table that is defined with system-period data versioning. See also bitemporal table, system-period data versioning, and history table.

**Systems Network Architecture (SNA)**

The description of the logical structure, formats, protocols, and operational sequences for transmitting information

through and controlling the configuration and operation of networks.

**table**

A named data object consisting of a specific number of columns and some number of unordered rows. See also base table or temporary table. Contrast with auxiliary table, clone table, materialized query table, result table, and transition table.

**table-controlled partitioning**

A type of partitioning in which partition boundaries for a partitioned table are controlled by values that are defined in the CREATE TABLE statement.

**table function**

A function that receives a set of arguments and returns a table to the SQL statement that references the function. A table function can be referenced only in the FROM clause of a subselect.

**table locator**

A mechanism that allows access to trigger tables in SQL or from within user-defined functions. A table locator is a fullword integer value that represents a transition table.

**table space**

A page set that is used to store the records in one or more tables. See also partitioned table space, segmented table space, and universal table space.

**table space set**

A set of table spaces and partitions that should be recovered together for one of the following reasons:

- Each of them contains a table that is a parent or descendent of a table in one of the others.
- The set contains a base table and associated auxiliary tables.

A table space set can contain both types of relationships.

**task control block (TCB)**

A z/OS control block that is used to communicate information about tasks within an address space that is connected to a subsystem. See also address space connection.

**TB**

Terabyte. A value of 1 099 511 627 776 bytes.



**TCB** See task control block.

**TCP/IP**

A network communication protocol that computer systems use to exchange information across telecommunication links.

**TCP/IP port**

A 2-byte value that identifies an end user or a TCP/IP network application within a TCP/IP host.

**template**

A DB2 utilities output data set descriptor that is used for dynamic allocation. A template is defined by the TEMPLATE utility control statement.

**temporal table**

A table that records the period of time when a row is valid.

**temporary table**

A table that holds temporary data. Temporary tables are useful for holding or sorting intermediate results from queries that contain a large number of rows. The two types of temporary table, which are created by different SQL statements, are the created temporary table and the declared temporary table. Contrast with result table. See also created temporary table and declared temporary table.

**textual XML format**

A representation of XML data that uses character values, an approach that allows for direct reading by people.

**thread** See DB2 thread.

**threadsafe**

A characteristic of code that allows multithreading both by providing private storage areas for each thread, and by properly serializing shared (global) storage areas.

**three-part name**

The full name of a table, view, or alias. It consists of a location name, a schema name, and an object name, separated by a period.

**time** A three-part value that designates a time of day in hours, minutes, and seconds.

**timeout**

Abnormal termination of either the DB2

subsystem or of an application because of the unavailability of resources. Installation specifications are set to determine both the amount of time DB2 is to wait for IRLM services after starting, and the amount of time IRLM is to wait if a resource that an application requests is unavailable. If either of these time specifications is exceeded, a timeout is declared.

**Time-Sharing Option (TSO)**

An option in z/OS that provides interactive time sharing from remote terminals.

**timestamp**

A seven-part value that consists of a date and time. The timestamp is expressed in years, months, days, hours, minutes, seconds, and microseconds.

**timestamp with time zone**

A two-part value that consists of a timestamp and time zone. The timestamp with time zone is expressed in years, months, days, hours, minutes, seconds, microseconds, time zone hours, and time zone minutes.

**trace**

A DB2 facility that provides the ability to monitor and collect DB2 monitoring, auditing, performance, accounting, statistics, and serviceability (global) data.

**transaction**

An atomic series of SQL statements that make up a logical unit of work. All of the data modifications made during a transaction are either committed together as a unit or rolled back as a unit.

**transaction lock**

A lock that is used to control concurrent execution of SQL statements.

**transaction program name**

In SNA LU 6.2 conversations, the name of the program at the remote logical unit that is to be the other half of the conversation.

**transaction-start-ID column**

A generated column that is defined with the AS TRANSACTION START ID clause. The value is assigned whenever a row is inserted into the table or any column in the row is updated. A transaction-start-ID column is intended for use in a

system-period temporal table. See also generated column, row-begin column, and row-end column.

**transition table**

A temporary table that contains all the affected rows of the subject table in their state before or after the triggering event occurs. Triggered SQL statements in the trigger definition can reference the table of changed rows in the old state or the new state. Contrast with auxiliary table, base table, clone table, and materialized query table.

**transition variable**

A variable that contains a column value of the affected row of the subject table in its state before or after the triggering event occurs. Triggered SQL statements in the trigger definition can reference the set of old values or the set of new values.

**tree structure**

A data structure that represents entities in nodes, with a most one parent node for each node, and with only one root node.

**trigger**

A database object that is associated with a single base table or view and that defines a rule. The rule consists of a set of SQL statements that run when an insert, update, or delete database operation occurs on the associated base table or view.

**trigger activation**

The process that occurs when the trigger event that is defined in a trigger definition is executed. Trigger activation consists of the evaluation of the triggered action condition and conditional execution of the triggered SQL statements.

**trigger activation time**

An indication in the trigger definition of whether the trigger should be activated before or after the triggered event.

**trigger body**

The set of SQL statements that is executed when a trigger is activated and its triggered action condition evaluates to true. A trigger body is also called triggered SQL statements.

**trigger cascading**

The process that occurs when the

triggered action of a trigger causes the activation of another trigger.

**triggered action**

The SQL logic that is performed when a trigger is activated. The triggered action consists of an optional triggered action condition and a set of triggered SQL statements that are executed only if the condition evaluates to true.

**triggered action condition**

An optional part of the triggered action. This Boolean condition appears as a WHEN clause and specifies a condition that DB2 evaluates to determine if the triggered SQL statements should be executed.

**triggered SQL statements**

The set of SQL statements that is executed when a trigger is activated and its triggered action condition evaluates to true. Triggered SQL statements are also called the trigger body.

**trigger granularity**

In SQL, a characteristic of a trigger, which determines whether the trigger is activated:

- Only once for the triggering SQL statement
- Once for each row that the SQL statement modifies

**triggering event**

The specified operation in a trigger definition that causes the activation of that trigger. The triggering event is comprised of a triggering operation (insert, update, or delete) and a subject table or view on which the operation is performed.

**triggering SQL operation**

The SQL operation that causes a trigger to be activated when performed on the subject table or view.

**trigger package**

A package that is created when a CREATE TRIGGER statement is executed. The package is executed when the trigger is activated.

**trust attribute**

An attribute on which to establish trust. A trusted relationship is established based on one or more trust attributes.



<b>trusted connection</b>	DB2 after Version 7 or that are specified as type 2 indexes in Version 4 or later.
A database connection whose attributes match the attributes of a unique trusted context defined at the DB2 database server.	<b>UCS-2</b> Universal Character Set, coded in 2 octets, which means that characters are represented in 16-bits per character.
<b>trusted connection reuse</b>	<b>UDF</b> See user-defined function.
The ability to switch the current user ID on a trusted connection to a different user ID.	<b>UDT</b> User-defined data type. In DB2 for z/OS, the term distinct type is used instead of user-defined data type. See distinct type.
<b>trusted context</b>	<b>uncommitted read (UR)</b>
A database security object that enables the establishment of a trusted relationship between a DB2 database management system and an external entity.	The isolation level that allows an application to read uncommitted data. See also cursor stability, read stability, and repeatable read.
<b>trusted context default role</b>	<b>underlying view</b>
A role associated with a trusted context. The privileges granted to the trusted context default role can be acquired only when a trusted connection based on the trusted context is established or reused.	The view on which another view is directly or indirectly defined.
<b>trusted context user</b>	<b>undo</b> A state of a unit of recovery that indicates that the changes that the unit of recovery made to recoverable DB2 resources must be backed out.
A user ID to which switching the current user ID on a trusted connection is permitted.	<b>Unicode</b>
<b>trusted context user-specific role</b>	A standard that parallels the ISO-10646 standard. Several implementations of the Unicode standard exist, all of which have the ability to represent a large percentage of the characters that are contained in the many scripts that are used throughout the world.
A role that is associated with a specific trusted context user. It overrides the trusted context default role if the current user ID on the trusted connection matches the ID of the specific trusted context user.	<b>union</b> An SQL operation that involves the UNION set operator, which combines the results of two SELECT statements. Unions are often used to merge lists of values that are obtained from two tables.
<b>trusted relationship</b>	<b>unique constraint</b>
A privileged relationship between two entities such as a middleware server and a database server. This relationship allows for a unique set of interactions between the two entities that would be impossible otherwise.	An SQL rule that no two values in a primary key, or in the key of a unique index, can be the same.
<b>TSO</b> See Time-Sharing Option.	<b>unique index</b>
<b>TSO attachment facility</b>	An index that ensures that no identical key values are stored in a column or a set of columns in a table.
A DB2 facility consisting of the DSN command processor and DB2I. Applications that are not written for the CICS or IMS environments can run under the TSO attachment facility.	<b>unit of recovery (UOR)</b>
<b>typed parameter marker</b>	A recoverable sequence of operations within a single resource manager, such as an instance of DB2. Contrast with unit of work.
A parameter marker that is specified along with its target data type. It has the general form:	<b>unit of work (UOW)</b>
CAST(? AS data-type)	A recoverable sequence of operations within an application process. At any
<b>type 2 indexes</b>	
Indexes that are created on a release of	

time, an application process is a single unit of work, but the life of an application process can involve many units of work as a result of commit or rollback operations. In a multisite update operation, a single unit of work can include several *units of recovery*. Contrast with unit of recovery.

**universal table space**

A table space that is both segmented and partitioned. Contrast with partitioned table space, segmented table space, partition-by-growth table space, and range-partitioned table space.

**unlock**

The act of releasing an object or system resource that was previously locked and returning it to general availability within DB2.

**untyped parameter marker**

A parameter marker that is specified without its target data type. It has the form of a single question mark (?).

**updatability**

The ability of a cursor to perform positioned updates and deletes. The updatability of a cursor can be influenced by the SELECT statement and the cursor sensitivity option that is specified on the DECLARE CURSOR statement.

**update hole**

The location on which a cursor is positioned when a row in a result table is fetched again and the new values no longer satisfy the search condition. See also delete hole.

**update trigger**

A trigger that is defined with the triggering SQL operation update.

**UR** See uncommitted read.

**user-defined data type (UDT)**

See distinct type.

**user-defined function (UDF)**

A function that is defined to DB2 by using the CREATE FUNCTION statement and that can be referenced thereafter in SQL statements. A user-defined function can be an external function, a sourced function, or an SQL function. Contrast with built-in function.

**user view**

In logical data modeling, a model or representation of critical information that the business requires.

**UTF-16**

Unicode Transformation Format, 16-bit encoding form, which is designed to provide code values for over a million characters and a superset of UCS-2. The CCSID value for data in UTF-16 format is 1200. DB2 for z/OS supports UTF-16 in graphic data fields.

**UTF-8**

Unicode Transformation Format, 8-bit encoding form, which is designed for ease of use with existing ASCII-based systems. The CCSID value for data in UTF-8 format is 1208. DB2 for z/OS supports UTF-8 in mixed data fields.

**value**

The smallest unit of data that is manipulated in SQL.

**variable**

A data element that specifies a value that can be changed. A COBOL elementary data item is an example of a host variable. Contrast with constant.

**variant function**

See nondeterministic function.

**varying-length string**

A character, graphic, or binary string whose length varies within set limits. Contrast with fixed-length string.

**version**

A member of a set of similar programs, DBRMs, packages, or LOBs.

- **A version of a program** is the source code that is produced by precompiling the program. The program version is identified by the program name and a timestamp (consistency token).
- **A version of an SQL procedural language routine** is produced by issuing the CREATE or ALTER PROCEDURE statement for a native SQL procedure.
- **A version of a DBRM** is the DBRM that is produced by precompiling a program. The DBRM version is identified by the same program name and timestamp as a corresponding program version.

	<ul style="list-style-type: none"> <li>• <b>A version of an application package</b> is the result of binding a DBRM within a particular database system. The application package version is identified by the same program name and consistency token as the DBRM.</li> <li>• <b>A version of a LOB</b> is a copy of a LOB value at a point in time. The version number for a LOB is stored in the auxiliary index entry for the LOB.</li> <li>• <b>A version of a record</b> is a copy of the record at a point in time.</li> </ul>	<p>dispatchable units (service request blocks and tasks) in multiple address spaces, allowing them to be reported on and managed by WLM as part of a single work request.</p>
<b>view</b>	A logical table that consists of data that is generated by a query. A view can be based on one or more underlying base tables or views, and the data in a view is determined by a SELECT statement that is run on the underlying base tables or views.	<p><b>write to operator (WTO)</b> An optional user-coded service that allows a message to be written to the system console operator informing the operator of errors and unusual system conditions that might need to be corrected (in z/OS).</p>
<b>Virtual Storage Access Method (VSAM)</b>	An access method for direct or sequential processing of fixed- and varying-length records on disk devices.	<b>WTO</b> See write to operator.
<b>Virtual Telecommunications Access Method (VTAM)</b>	An IBM licensed program that controls communication and the flow of data in an SNA network (in z/OS).	<b>WTOR</b> Write to operator (WTO) with reply.
<b>volatile table</b>	A table for which SQL operations choose index access whenever possible.	<b>XCF</b> See cross-system coupling facility.
<b>VSAM</b>	See Virtual Storage Access Method.	<b>XES</b> See cross-system extended services.
<b>VTAM</b>	See Virtual Telecommunications Access Method.	<b>XML</b> See Extensible Markup Language.
<b>warm start</b>	The normal DB2 restart process, which involves reading and processing log records so that data that is under the control of DB2 is consistent. Contrast with cold start.	<b>XML attribute</b> A name-value pair within a tagged XML element that modifies certain features of the element.
<b>WLM application environment</b>	A z/OS Workload Manager attribute that is associated with one or more stored procedures. The WLM application environment determines the address space in which a given DB2 stored procedure runs.	<b>XML column</b> A column of a table that stores XML values and is defined using the data type XML. The XML values that are stored in XML columns are internal representations of well-formed XML documents.
<b>WLM enclave</b>	A construct that can span multiple	<b>XML data type</b> A data type for XML values.
		<b>XML element</b> A logical structure in an XML document that is delimited by a start and an end tag. Anything between the start tag and the end tag is the content of the element.
		<b>XML index</b> An index on an XML column that provides efficient access to nodes within an XML document by providing index keys that are based on XML patterns.
		<b>XML lock</b> A column-level lock for XML data. The operation of XML locks is similar to the operation of LOB locks.
		<b>XML node</b> The smallest unit of valid, complete structure in a document. For example, a node can represent an element, an attribute, or a text string.

<b>XML node ID index</b>	
	An implicitly created index, on an XML
	table that provides efficient access to XML
	documents and navigation among
	multiple XML data rows in the same
	document.
<b>XML pattern</b>	
	A slash-separated list of element names,
	an optional attribute name (at the end), or
	kind tests, that describe a path within an
	XML document in an XML column. The
	pattern is a restrictive form of path
	expressions, and it selects nodes that
	match the specifications. XML patterns are
	specified to create indexes on XML
	columns in a database.
<b>XML publishing function</b>	
	A function that returns an XML value
	from SQL values. An XML publishing
	function is also known as an XML
	constructor.
<b>XML schema</b>	
	In XML, a mechanism for describing and
	constraining the content of XML files by
	indicating which elements are allowed
	and in which combinations. XML schemas
	are an alternative to document type
	definitions (DTDs) and can be used to
	extend functionality in the areas of data
	typing, inheritance, and presentation.
<b>XML schema repository (XSR)</b>	
	A repository that allows the DB2 database
	system to store XML schemas. When
	registered with the XSR, these objects
	have a unique identifier and can be used
	to validate XML instance documents.
<b>XML serialization function</b>	
	A function that returns a serialized XML
	string from an XML value.
<b>XML table</b>	
	An auxiliary table that is implicitly
	created when an XML column is added to
	a base table. This table stores the XML
	data, and the column in the base table
	points to it.
<b>XML table space</b>	
	A table space that is implicitly created
	when an XML column is added to a base
	table. The table space stores the XML
	table. If the base table is partitioned, one
	partitioned table space exists for each
	XML column of data.
	<b>X/Open</b>
	An independent, worldwide open systems
	organization that is supported by most of
	the world's largest information systems
	suppliers, user organizations, and
	software companies. X/Open's goal is to
	increase the portability of applications by
	combining existing and emerging
	standards.
	<b>XRF</b> See Extended Recovery Facility.
<b>XSR</b>	See XML schema repository.
<b>zIIP</b>	See IBM System z9 Integrated Processor.
<b>z/OS</b>	An operating system for the System z
	product line that supports 64-bit real and
	virtual storage.
<b>z/OS Distributed Computing Environment (z/OS DCE)</b>	A set of technologies that are provided by
	the Open Software Foundation to
	implement distributed computing.



---

# Index

## A

- access checking
  - mandatory 111, 113
  - object security labels 111
  - user security labels 111
- access control
  - applications 85
  - auditing 281
  - authorities 5, 16, 22
  - authorization IDs 21, 22
  - authorizing 85, 86
  - catalog tables 101
  - CICS 127
  - columns 189
  - data definition control
    - overview 215
  - DB2 subsystem
    - RACF 7
  - exit routines 6, 229
  - external 127
    - overview 127
  - external DB2 data sets 8
  - IMS 127
  - internal
    - overview 21
  - internal DB2 data 4
  - managing 85
  - multilevel security 6
  - object ownership 6
  - privileges 5, 16, 22
  - RACF 7, 127
  - restricting 86
  - roles 5, 21, 22
  - rows 189
  - subsystems
    - local 7, 153
    - remote 7, 159
  - trusted connections 205
  - trusted contexts
    - overview 205
- access control authorization routine 243
  - ACEE 243, 244
  - active 262
  - authorization IDs 244
  - debugging 262
  - default 243
  - expected output 259
  - EXPLAIN STMTCACHE 247
  - invoking 243
  - overview 241
  - packages
    - inoperative 245
  - parameter list 249
  - processing exceptions 261
  - reason codes 260
  - return codes 259
  - specifying 242
- access profile
  - defining 128
  - RACF 128
- access requests
  - remote
    - permitting 135
- ACCESSCTRL authority 31, 42, 52, 55
  - granting privileges 59
  - revoking privileges 59
- accessibility
  - keyboard x
  - shortcut keys x
- accessing
  - data 59
- activating
  - client certificate name filters 266
- adding
  - RACF groups 132
  - SAF user mapping plug-in 152
- address spaces
  - started-task 129
  - WLM-established 138
- altering
  - tables 58
- application registration tables (ART)
  - ART columns 216
- applications
  - authorization
    - validating 83
  - executing 83
  - RRSAF 83
- AT-TLS
  - configuring 263
  - data protection 263
- attachment requests
  - remote 166
- audit classes 283
- audit policies
  - audit categories 288
  - creating 291
  - displaying 291
  - overview 288
- audit policy 1
- audit trace records 285
  - collecting 287
  - formatting 287
- audit trace reports 284
- audit traces 282, 294
  - audit classes 283
  - limitations 285
  - starting 286
  - stopping 286
- auditing
  - access
    - overview 281
  - administrative authorities 293
  - AUDIT ALL option 16
  - audit categories 288
  - audit policies 288
  - authorization IDs 282, 294, 295
  - distributed environment 288
  - security measures 294
  - tables 295
    - AUDIT clause 293

- authentication level
  - configuring 264
  - Secure Socket Layer (SSL) 264
- authorities
  - access control 5
  - ACCESSCTRL 1
  - ACCESSCTRL authority 31, 42, 59
  - administrative 24, 31, 36, 37, 38, 39, 40, 41, 42, 45, 51, 52, 55, 57, 58, 59
    - auditing 293
  - DATAACCESS 1
  - DATAACCESS authority 31, 42, 59
  - DB2 catalogs 44
  - DBADM authority 31, 39, 42, 103
  - DBCTRL authority 31, 39
  - DBMAINT authority 31, 39
  - directories 44
  - installation SYSADM authority 31, 36
  - installation SYSOPR authority 31, 38
  - managing 51
  - migrating 55
  - PACKADM authority 31, 40
  - REVOKE statement 74
  - revoking 74
  - SECADM 1
  - SECADM authority 31, 41, 42, 57
  - security 41
  - separating 1, 52
  - SQLADM 1
  - SQLADM authority 31, 42
  - SYSADM authority 31, 36, 42, 70, 74
  - SYSCTRL authority 31, 37
  - SYSOPR authority 31, 38
  - system DBADM 1
  - system DBADM authority 31, 40, 42, 58, 103
  - utilities 44
- authorization
  - access control 47
  - failure code 163
  - views 246
- authorization IDs 63, 68, 71, 72
  - access control 21
  - audit traces 282
  - auditing 282, 294, 295
  - caching 84, 85
  - determining 96
  - dynamic SQL 92, 96
  - managing access 21
  - packages 84
  - plans 84
  - primary 22, 45
  - RACF 22
  - routines 85
  - secondary 22, 45, 65
    - connection processing 156
    - sign-on processing 158
  - sign-on processing 157
  - SQL 22, 45
  - translating
    - inbound IDs 167
    - outbound IDs 183
  - validation 91
- automatic rebind
  - roles 245

## B

- bind behavior 94
  - attributes 93
- BINDAGENT
  - RACF 246
  - roles 246

## C

- caching
  - authorization IDs 84
    - plans 84
    - routines 85
  - EXECUTE privilege
    - packages 247
    - plans 247
    - routines 247
  - roles 84
  - security labels 117
  - SQL statements 248
- catalog tables
  - privilege records 101
- catalogs 44
- CD-ROM, books on 309
- CICS 7
  - connection routines
    - samples 159
  - sign-on routines
    - samples 159
- client certificate 264
- client certificate name filters
  - activating 266
  - creating 266
- collecting
  - audit trace records 287
- collection privileges
  - CREATE IN privilege 24
- column access control 1, 202
  - activating 198, 200
  - column masks 189, 191, 198, 200
  - deactivating 198, 200
- column masks 189, 191, 192
  - creating 198
  - modifying 200
- column-level encryption
  - password hints 275
  - views 274
- commands
  - DB2
    - DISPLAY DATABASE command 301
    - DISPLAY TRACE command 286
      - required authorization 135
    - START TRACE command 286
    - STOP TRACE command 286
  - DISPLAY DATABASE command 301
  - RACF
    - ADDGROUP command 132
    - WLM REFRESH command 140
- communications databases (CDB)
  - requesters 175
  - servers 161
- configuring
  - AT-TLS 263
  - enterprise identity mapping (EIM)
    - domain controller 151
  - IMS 137



- configuring *(continued)*
  - LDAP
    - server 149
  - RACF
    - LDAP server 150
  - Secure Socket Layer (SSL) 263
  - SNA 126
  - SSL authentication level 264
  - TCP/IP 126
- connection
  - requests
    - connection processing 153
    - local 153
    - managing 153
- connection processing
  - attachment requests 163, 180
  - BATCH 154
  - CICS 154
  - IMS 154
  - RACF 154
  - secondary authorization IDs 156
  - TSO 154
- connection requests
  - inbound 170, 171
  - managing 163, 170, 174
  - outbound 174
  - processing 154, 171
  - SNA-based 163
  - TCP/IP-based 170, 171
- connection routines 238
  - debugging 239
  - expected output 236
  - input values 235
  - invoking 231
  - overview 229
  - parameter list 231, 234
  - processing 237
  - samples 230
  - session variables 240
  - specifying 230
- connections
  - VTAM 129
- creating
  - audit policies 291
  - client certificate name filters 266
  - materialized query tables 248
  - table spaces 227
  - trusted connections
    - local 207

## D

- data
  - access control 4
  - accessing 68
  - changes
    - tracking 298
  - consistency 296, 299, 300, 301, 302
    - SQL queries 300
    - verifying 299
  - distributed 68
  - integrity 296, 297
  - internal integrity reports 302
  - operation logs 301
  - transactions
    - database balancing 298
  - uniqueness 297

- data consistency
  - locks 299
  - referential integrity 299
- data definition control
  - access control 215
  - application registration tables (ART) 218
  - controlling
    - application name 219
    - application name with exceptions 220
    - object name with exceptions 222
    - object names 221
  - data definition statements 215
- DB2 support
  - DSNTIPZ panel 215
  - installing 218
  - disabling 225
  - managing 215
  - object registration tables (ORT) 218
  - restarting 225
  - stopping 225
- data definition statements 47
  - data definition control 215
- data protection
  - DB2 built-in functions 272
  - encryption 8
    - overview 263
  - RACF 8
    - overview 263
  - Secure Socket Layer (SSL)
    - DB2 support 263
- data sets
  - adding groups 269
  - creating 272
  - generic profiles
    - adding authorization IDs 271
    - creating 270
  - protecting 269
- DATAACCESS authority 31, 42, 52, 55
  - accessing data 59
- database balancing
  - incomplete transactions 298
  - lost transactions 298
- database privileges
  - CREATETAB privilege 25
  - CREATETS privilege 25
  - DISPLAYDB privilege 25
  - DROP privilege 25
  - IMAGCOPY privilege 25
  - LOAD privilege 25
  - RECOVERDB privilege 25
  - REORG privilege 25
  - REPAIR privilege 25
  - STARTDB privilege 25
  - STATS privilege 25
  - STOPDB privilege 25
- DB2 books online 309
- DB2 directories 44
- DB2 Information Center for z/OS solutions 309
- DB2 support
  - enterprise identity mapping (EIM) 148
  - z/OS identity filter 152
- DB2I 211
- DBADM authority 31, 39, 42
  - managing access 17
- DBCTRL authority 31, 39
- DBMAINT authority 31, 39, 42

- decryption
  - DB2 built-in functions 272
- define behavior 94
  - attributes 93
- definer, description 87
- defining
  - column-level encryption 273
  - DB2 resources
    - RACF 127
  - external security profiles 213
  - trusted contexts 206
  - user-defined functions (UDF) 90
- DELETE statement 123, 192
- denial-of-service attack
  - managing 173
- dependent privileges 70
- disability x
- disabling
  - data definition control 225
- DISPLAY TRACE command 286
- displaying
  - audit policies 291
- distinct types
  - stored procedures 142
- distributed access
  - implementing 11, 12
  - planning 11
  - servers
    - central 11
    - remote 12
  - views 11
- distributed environment
  - auditing 288
- DRDA access
  - security mechanisms 159, 160
- DROP statement 72
- dropping
  - views 247
- DSN command processor 211
- DSNDB01 database 44
- DSNDB06 database 44
- dynamic SQL
  - authorization 92
- DYNAMICRULES 92, 96
- DYNAMICRULES(BIND)
  - roles 245

## E

- edit procedures 110
- encrypted data
  - defining columns 273
  - performance optimization 277
  - predicates 277
- encrypting
  - AES 186
  - DES 186
  - non-character values 277
  - passwords 185, 186
- encryption
  - AT-TLS 263
  - column level
    - defining 273
  - column-level encryption 274, 275
  - data protection 263
  - DB2 built-in functions 272
  - encrypted data 273, 277

- encryption (*continued*)
  - non-character values 277
  - options 8
  - password hints 275
  - performance optimization 277
  - predicates 277
  - Secure Socket Layer (SSL) 263, 267, 268
  - value-level encryption 275, 276
- enterprise identity mapping (EIM)
  - configuring 151
  - DB2 support 148
  - domain controller 151
  - implementing 148
- establishing
  - trusted connections
    - remote 208, 209
- executing
  - stored procedure 144
- exit routines
  - access control 6, 229
  - managing access 229
- EXPLAIN STMTCACHE
  - SQL statements 247
- explicit privileges 24
  - collection privileges 24
  - database privileges 24, 25
  - distinct type privileges 24
  - function privileges 24
  - granting 60, 61
  - JAR privileges 24
  - managing 60
  - package privileges 24, 25
  - plan privileges 24, 26
  - procedure privileges 24
  - routine privileges 24, 26
  - schema privileges 24, 26
  - sequence privileges 24
  - system privileges 24, 27
  - table privileges 24, 28
  - usage privileges 24, 29
  - use privileges 24, 29
  - view privileges 24, 28
- explicit view privileges
  - ALTER privilege 28
  - DELETE privilege 28
  - GRANT ALL privilege 28
  - INDEX privilege 28
  - INSERT privilege 28
  - REFERENCES privilege 28
  - SELECT privilege 28
  - TRIGGER privilege 28
  - UPDATE privilege 28

## F

- field procedures 110
- formatting
  - audit trace records 287

## G

- general-use programming information, described 317
- global temporary tables 109
- GRANT statement 62, 63, 65, 66, 67, 68
  - PUBLIC clause 60, 61, 62
  - ROLE AS OBJECT OWNER clause 61

- granting
  - privileges 59
  - write-down privileges 113
- GUIP symbols 317

## I

- image copies 294
- implementing
  - column access control 189
  - enterprise identity mapping (EIM) 148
  - multilevel security 107
  - row access control 189
  - user-defined functions (UDF) 88
  - z/OS identity filter 152
- implementor, description 87
- implicit privileges 24
  - granting
    - object ownership 80
  - managing 80, 87
    - object ownership 77
    - stored procedures 77
  - object ownership 30, 78, 80
    - changing 79
    - trusted contexts 79
  - routines 87
- inbound IDs
  - associating
    - secondary IDs 170
  - managing 166
  - translating 167
- indexes
  - creating 225
  - dropping 227
  - managing 225
  - naming 226
- INSERT statement 119, 192
  - column access control 202
  - row access control 202
- installation SYSADM authority 31, 36
- installation SYSOPR authority 31, 38
- installing
  - DB2 support
    - data definition control 218
- invoke behavior 94
  - attributes 94
- invoker, description 87

## J

- JAR files 143

## K

- Kerberos
  - authenticating 167
  - authentication
    - RACF 147

## L

- LDAP
  - configuring 149
    - RACF 150
  - z/OS 149

- library 309

## M

- managing
  - authorities 51
  - connection requests
    - outbound 174
    - SNA-based 163
  - data
    - multilevel-secure environment 118
  - denial-of-service attack 173
- materialized query tables 110
  - dropping 73
- MERGE statement 122, 192
- migrating
  - ACCESSCTRL authority 55
  - authorities 55
  - DATAACCESS authority 55
  - SECADM authority 55
  - SYSADM authority 55
  - SYSCTRL authority 55
  - system DBADM authority 55
- multilevel security
  - access control 6
  - advantages 107
  - constraints 110
  - discretionary access checking 111
  - distributed environment 126
  - edit procedures 110
  - field procedures 110
  - global temporary tables 109
  - implementing 113, 115
  - mandatory access checking 111
  - materialized query tables 110
  - objects 109, 113
  - rows 115
  - security categories 109
  - security labels 108
  - security levels 109
  - SNA
    - configuring 126
  - tables
    - adding 116
    - columns 116, 117
    - creating 116
    - removing 117
  - TCP/IP
    - configuring 126
  - triggers 111
  - users 109
  - validation procedures 110
  - views 117

## O

- object owners
  - managing access 18
- object ownership
  - access control 6
  - aliases 30
  - changing 79
  - databases 30
  - distinct types 30
  - implicit privileges 77
    - granting 80

- object ownership (*continued*)
  - indexes 30
  - JAR 30
  - packages 30, 80, 81
  - plans 30, 80, 81
  - privileges
    - implicit 30
  - qualified names 78
  - roles 30
  - sequences 30
  - storage groups 30
  - stored procedures 30
  - synonyms 30
  - table spaces 30
  - tables 30
  - trusted contexts 30, 79
  - unqualified names 78
  - user-defined functions 30
  - views 30
- object registration tables (ORT)
  - ORT columns 216
- object sets
  - registering 224
- objectives 9
- objects
  - multilevel security 109
- online 309
- online books 309
- outbound IDs
  - translating 183

## P

- PACKADM authority 31, 40
- package ownership
  - changing 80, 81
  - creating 80, 81
  - trusted contexts 81
- package privileges
  - BIND privilege 25
  - COPY privilege 25
  - EXECUTE privilege 25
  - GRANT ALL privilege 25
- packages
  - access authorization 85
  - authorization
    - validating 82, 83
  - binding 67
  - executing 82, 83
  - inoperative 74
  - invalidating 74
  - rebinding 67
  - unqualified names 81
- parameters
  - REVOKE DEP PRIV parameter 70, 74
  - SEPARATE SECURITY parameter 36, 37, 58, 59, 189, 202
  - SEPARATE\_SECURITY parameter 1, 31, 51, 52, 55, 57, 190, 191, 196, 198, 200, 203
  - system 1, 31, 36, 37, 51, 52, 55, 57, 58, 59, 70, 74, 189, 190, 191, 196, 198, 200, 202, 203
- PassTickets
  - configuring 186
  - RACF 167
- passwords
  - changing 163
  - encrypting 167, 186
  - RACF-encrypted 185
- passwords (*continued*)
  - sending 185
- performance optimization
  - encrypted data 277
- plan ownership
  - changing 80, 81
  - creating 80, 81
  - trusted contexts 81
- plan privileges
  - BIND privilege 26
  - EXECUTE privilege 26
- plans
  - access authorization 85, 86
  - authorization
    - validating 82
  - binding 66, 67
  - executing 82
    - remotely 86
  - rebinding 67
  - unqualified names 81
- port of entry 165
  - RACF APPCPORT class 135
  - RACF SERVAUTH class 136
- predicates
  - encrypted data 277
- preventing
  - SQL injection attacks 174
- primary authorization ID 22
- privileges
  - access control 5
  - application programmers 45
  - authorization IDs 45, 102
  - catalog tables 103
  - composite
    - using 100
  - CREATE DATABASE statement 50
  - CREATE INDEX statement 50
  - CREATE STOGROUP statement 50
  - CREATE TABLE statement 50
  - CREATE TABLESPACE statement 50
  - CREATE VIEW statement 50
  - CREATEIN privilege
    - granting 141
  - data
    - distributed 68
  - data definition statements 47
  - database administrators 45
  - dependent 70
  - DROP statement 72
  - dynamic SQL 50
  - EXECUTE privilege
    - caching 247
  - executing
    - routines 87
  - EXPLAIN 1
  - explicit 24, 25, 26, 27, 28, 29, 60, 61
  - GRANT statement 50, 63, 65, 66, 67, 68
  - granted 102
  - granting 15, 61, 62, 63, 65, 66, 67, 68, 80, 88
    - distinct types 142
  - JAR files 143
    - stored procedure packages 143
    - stored procedures 143
  - implicit 24, 30, 77, 78, 79, 80, 81, 87, 214
  - information center consultants 45
  - multiple grants 103
  - object ownership 30

- privileges (*continued*)
  - package administrators 45
  - packages 48, 67, 101
  - plans 48, 66, 67, 101
  - production binders 45
  - PUBLIC ID 61, 62
  - query users 45
  - restrictions 74
  - REVOKE statement 50, 68, 70, 71, 72, 73, 74
  - revoking 68, 70, 71, 72, 73, 74
  - roles 45, 102
  - routines 88
  - security administrators 45
  - static SQL 50
  - system administrators 45
  - system operators 45
  - system programmers 45
  - trusted contexts 214
  - user analysts 45
  - users
    - group 65
    - remote 62
  - views 62
- product-sensitive programming information, described 317
- programming interface information, described 317
- PSPI symbols 317
- PUBLIC clause
  - GRANT statement 60, 61, 62
- PUBLIC ID 61, 62

## R

- RACF
  - access authorization
    - protected resources 129
    - SERVER resource class 138
  - access checking
    - DSNR class 128
    - SERVER class 128
  - access control 7, 127
    - non-DB2 resources 140
  - authorization 8, 137
  - authorizing
    - group access 134
  - BINDAGENT 246
  - data protection 8, 263, 269
  - DB2 resources
    - defining 127
  - defining
    - access profiles 128
    - DB2 resources 146
    - user IDs 129
  - encrypted passwords 185
  - groups
    - adding 132
  - Kerberos authentication 147
  - managing access 127
    - DB2 127
  - PassTickets 186
  - roles 246
- RACF access control module 262
- RACF groups
  - creating 15, 16
- recovery logs 294
- registering
  - object sets 224

- registration tables
  - adding columns 227
  - application registration tables (ART) 216
  - creating 225
  - dropping 227
  - managing 225
  - naming 226
  - object registration tables (ORT) 216
  - updating 227
- Resource Recovery Services attachment facility (RRSAF)
  - RACF profiles 138
  - stored procedures 138
- retrieving
  - authorization IDs 102, 103, 104, 105
  - multiple grants 103
  - packages 105
  - plans 105
  - privilege records 106
  - roles 102, 103, 104, 105
- reusing
  - trusted connections 210, 211, 212
- REVOKE statement 68, 70, 71, 72, 73, 74
  - ROLE AS OBJECT OWNER clause 72
- revoking
  - privileges 59
- ROLE AS OBJECT OWNER clause
  - GRANT statement 61
- roles 22, 45, 63, 68
  - access control 5, 21
  - automatic rebind 245
  - BINDAGENT 246
  - caching 84
  - dropping 72
  - DYNAMICRULES(BIND) 245
  - managing 21
  - packages 84
  - privileges 61
  - RACF 246
  - ROLE AS OBJECT OWNER clause 61
  - SECADM authority 57
  - trusted connection 23
  - trusted contexts 23, 57
- routine privileges
  - EXECUTE ON FUNCTION privilege 26
  - EXECUTE ON PROCEDURE privilege 26
- routines
  - access authorization
    - simplifying 100
  - access control authorization routine 241, 242, 243, 244, 245, 247, 249, 259, 260, 261, 262
  - authorization IDs 105
  - connection routines 229, 230, 231, 234, 235, 236, 237, 238, 239, 240
  - executing 87
  - implicit privileges 87
  - privileges 87
    - granting 88
  - roles 105
  - sign-on routines 229, 230, 231, 234, 235, 236, 237, 238, 239, 240
  - stored procedures 87
  - user-defined functions (UDF) 87
- row access control 1, 202
  - activating 196
  - deactivating 196
  - row permissions 189, 190, 196
- row and column access control 203

row and column access control (*continued*)

- access types 192
- column masks 192
- implementing 189
- row permissions 192
- rules 192

row permissions 189, 190, 192

- creating 196

RRSAF 83, 212

run behavior

- attributes 92

## S

SAF

- user mapping plug-in
  - adding 152

scenarios

- security plans 9

schema privileges

- ALTERIN privilege 26
- CREATEIN privilege 26
- DROPIN privilege 26

schemas

- stored procedures 141

SECADM authority 31, 41, 42, 52, 55

- roles 57
- trusted contexts 57

secondary authorization ID

- RACF ID 22
- SQL ID 22

secondary IDs

- privileges 65

Secure Socket Layer (SSL)

- authentication level 264
- configuring 264
- configuring 263
- data protection 263
- DB2 requesters
  - configuring 268
- DB2 servers
  - configuring 267

security

- active security measures 281
  - column access control 189
  - DB2 1
  - DB2 10 for z/OS 1
  - DB2 solutions
    - overview 1
  - getting started 1
  - mechanisms 159, 160
    - DRDA access 159
  - multilevel 107, 108, 109, 110, 111, 113, 115, 116, 117, 118, 119, 120, 122, 123, 125, 126
    - implementing 107
    - managing 118
  - profiles
    - defining 213
  - row access control 189
  - security measures 294
- security categories 109
- security labels 108
- caching 117
  - columns 116
  - determining 109
  - objects 111, 112
  - RACF resource classes 113

security labels (*continued*)

- relationships
  - dominance 112
- users 111, 112

security levels 109

security plans 9

- access control 16, 18
- access restrictions 9, 13
- auditing access 13, 16
- distributed access 11, 12
- privileges
  - granting 15

- RACF groups
  - creating 15, 16

scenarios 9

SELECT privilege 10

tables

- keys 14
- triggers 14
- updating 14

views 10

- creating 10, 13

SELECT statement 118, 192

separating

- ACCESSCTRL authority 52
- authorities 52
- DATAACCESS authority 52
- SECADM authority 52
- SYSADM authority 52
- SYSCTRL authority 52
- SYSOPR authority 52
- system DBADM authority 52

server certificate 264

shortcut keys

- keyboard x

sign-on processing

- requests 153
- usage 153

sign-on requests

- authorization IDs 157
- secondary authorization IDs 158

sign-on routines 238

- debugging 239
- expected output 236
- input values 235
- invoking 231
- overview 229
- parameter list 231, 234
- processing 237
- samples 230
- session variables 240
- specifying 230

simplifying

- access authorization
  - routines 100

SNA access

- protocols 163
- security mechanisms 159, 160

softcopy publications 309

SQL 174

- attributes 94
- CONNECT statement 212
- CREATE statement 225
- dynamic 50, 92, 93, 94, 96
- static 50

SQL injection attacks

- preventing 174

- SQL statements
  - caching 248
- SQLADM authority 31, 42
- START TRACE command 286
- starting
  - audit traces 286
- STOP TRACE command 286
- stopping
  - audit traces 286
  - data definition control 225
- stored procedure packages 143
- stored procedures 93, 94, 143
  - access control
    - non-DB2 resources 140
  - creating 139
  - distinct types 142
  - executing
    - remote 144
  - required authorization 139
    - managing 137
  - trusted contexts 144
  - WLM 138, 139, 140
- subsystems
  - access control 7, 127
  - managing 127
- syntax diagram
  - how to read xi
- SYSADM authority 31, 36, 52, 55, 74
  - managing access 17
- SYSCTRL authority 31, 37, 52, 55
- SYSIBM.IPNAMES columns 176
- SYSIBM.LOCATIONS columns 179
- SYSIBM.LUNAMES columns 161, 175
- SYSIBM.USERNAMES columns 162, 178
- SYSOPR authority 31, 38, 52
- system administrator
  - privileges 63
- system DBADM authority 31, 40, 42, 52, 55
  - altering tables 58
- System Management Facility (SMF) 282
- system privileges
  - ARCHIVE privilege 27
  - BINDADD privilege 27
  - BINDAGENT privilege 27
  - BSDS privilege 27
  - CREATE\_SECURE\_OBJECT privilege 27
  - CREATEALIAS privilege 27
  - CREATEDBA privilege 27
  - CREATEDBC privilege 27
  - CREATESG privilege 27
  - CREATETMTAB privilege 27
  - DEBUGSESSION privilege 27
  - DISPLAY privilege 27
  - EXPLAIN privilege 27
  - MONITOR1 privilege 27
  - MONITOR2 privilege 27
  - RECOVER privilege 27
  - STOPALL privilege 27
  - STOSPACE privilege 27
  - TRACE privilege 27
- system programmer 46

## T

- table privileges
  - ALTER privilege 28
  - DELETE privilege 28

- table privileges (*continued*)
  - GRANT ALL privilege 28
  - INDEX privilege 28
  - INSERT privilege 28
  - REFERENCES privilege 28
  - SELECT privilege 28
  - TRIGGER privilege 28
  - UPDATE privilege 28
- table spaces
  - registration tables 227
- tables
  - auditing 295
  - authorization IDs 104
  - catalogs
    - access control 101
    - privileges 101
  - creating 248
  - packages 105
  - plans 105
  - roles 104
  - updating 14
- tasks
  - DB2-started 129
- TCP/IP
  - connection requests
    - protecting 146
  - protocols 170, 171
- tracking
  - data changes 298
- translating
  - inbound IDs 167
  - outbound IDs 183
- triggers 111, 203
  - creating 203
- TRUNCATE statement 125
- trusted connections
  - local
    - creating 207
    - reusing 211, 212
  - remote
    - establishing 208, 209
    - reusing 211
  - reusing 210
  - roles 23
  - trusted contexts 206
- trusted contexts 214
  - access control 205
  - ASUSER 213
  - BINDAGENT 246
  - defining 206
  - managing 205
  - object ownership 79
  - package ownership 81
  - plan ownership 81
  - RACF 246
  - roles 23, 57, 246
  - stored procedure 144
  - trusted connections 205, 206, 213

## U

- UPDATE statement 120, 192
- updating
  - registration tables 227
- usage privileges
  - USAGE ON DISTINCT TYPE privilege 29
  - USAGE ON JAR privilege 29



- usage privileges (*continued*)
  - USAGE ON SEQUENCE privilege 29
- use privileges
  - USE OF BUFFERPOOL privilege 29
  - USE OF STOGROUP privilege 29
  - USE OF TABLESPACE privilege 29
- user-defined functions (UDF) 93, 94, 200
  - defining 90
  - implementing 88
  - invoking 248
  - using 91
- users
  - group 65
  - multilevel security 109
- using
  - user-defined functions (UDF) 91
- utilities 125
  - CHECK DATA utility 300
  - CHECK INDEX utility 300
  - CHECK LOB utility 300
  - REPORT utility 301

## V

- validation procedures 110
- value-level encryption
  - defining 275
  - using passwords hints 276
- verifying
  - VTAM partner LUs 166
- views 62
  - authorization 246
  - creating 10, 13
  - dropping 72, 247
  - privileges records 106
- VTAM
  - APPL statement 166
  - connection control 129, 165
  - conversation-level security 166
  - partner LU verification 166
  - passwords
    - choosing 165
- VTAM partner LUs
  - authenticating 167
  - verifying 166

## W

- WLM
  - refreshing 140
  - stored procedures 138, 140
    - creating 139
- write-down control
  - mandatory access checking 113
  - write-down privileges 113
- write-down privileges
  - granting 113

## Z

- z/OS console logs 294
- z/OS identity filter
  - DB2 support 152
  - implementing 152





Product Number: 5605-DB2  
5697-P31

Printed in USA

SC19-3496-00



Spine information:

DB2 10 for z/OS

Managing Security

