CA IDMS™

Security Administration Guide

Release 18.5.00

technologies

This Documentation, which includes embedded help systems and electronically distributed materials, (hereinafter referred to as the “Documentation”) is for your informational purposes only and is subject to change or withdrawal by CA at any time. This Documentation is proprietary information of CA and may not be copied, transferred, reproduced, disclosed, modified or duplicated, in whole or in part, without the prior written consent of CA.

If you are a licensed user of the software product(s) addressed in the Documentation, you may print or otherwise make available a reasonable number of copies of the Documentation for internal use by you and your employees in connection with that software, provided that all CA copyright notices and legends are affixed to each reproduced copy.

The right to print or otherwise make available copies of the Documentation is limited to the period during which the applicable license for such software remains in full force and effect. Should the license terminate for any reason, it is your responsibility to certify in writing to CA that all copies and partial copies of the Documentation have been returned to CA or destroyed.

TO THE EXTENT PERMITTED BY APPLICABLE LAW, CA PROVIDES THIS DOCUMENTATION “AS IS” WITHOUT WARRANTY OF ANY KIND, INCLUDING WITHOUT LIMITATION, ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NONINFRINGEMENT. IN NO EVENT WILL CA BE LIABLE TO YOU OR ANY THIRD PARTY FOR ANY LOSS OR DAMAGE, DIRECT OR INDIRECT, FROM THE USE OF THIS DOCUMENTATION, INCLUDING WITHOUT LIMITATION, LOST PROFITS, LOST INVESTMENT, BUSINESS INTERRUPTION, GOODWILL, OR LOST DATA, EVEN IF CA IS EXPRESSLY ADVISED IN ADVANCE OF THE POSSIBILITY OF SUCH LOSS OR DAMAGE.

The use of any software product referenced in the Documentation is governed by the applicable license agreement and such license agreement is not modified in any way by the terms of this notice.

The manufacturer of this Documentation is CA.

Provided with “Restricted Rights.” Use, duplication or disclosure by the United States Government is subject to the restrictions set forth in FAR Sections 12.212, 52.227-14, and 52.227-19(c)(1) - (2) and DFARS Section 252.227-7014(b)(3), as applicable, or their successors.

CA Technologies Product References

This document references the following CA technologies products:

- CA ACF2™ for z/OS
- CA ADS™
- CA Common Services for z/OS (CCS)
- CA IDMS™/DB
- CA IDMS™/DC
- CA IDMS™/DC or CA IDMS™ UCF (DC/UCF)
- CA Top Secret® for z/OS

Contact CA Technologies

Contact CA Support

For your convenience, CA Technologies provides one site where you can access the information that you need for your Home Office, Small Business, and Enterprise CA Technologies products. At http://ca.com/support, you can access the following resources:

- Online and telephone contact information for technical assistance and customer services
- Information about user communities and forums
- Product and documentation downloads
- CA Support policies and guidelines
- Other helpful resources appropriate for your product

Providing Feedback About Product Documentation

If you have comments or questions about CA Technologies product documentation, you can send a message to techpubs@ca.com.

To provide feedback about CA Technologies product documentation, complete our short customer survey which is available on the CA Support website at http://ca.com/docs.

Documentation Changes

The following documentation updates were made for the 18.5.00, 2nd Edition release of this documentation:

- External Security Enforcement (see page 28)—Updated information about the use of SAF and RACROUTE.
- External Signon Security (see page 50)—Added information about the definition of PassTickets.
- Identifying Authorities to External System (see page 54)—Updated to reflect AISSF to RACROUTE authority changes.
- Defining Signon Security Options (see page 57)—Added information about using PassTickets.
- User Validation Processing (see page 61)—Added information about using PassTickets.

Contents

Chapter 6: Using CA IDMS Internal Security 65

Chapter 7: Securing Global Resources 83

Chapter 8: Securing System Resources 95

Chapter 9: Securing Database Resources 117

Chapter 10: Securing Application Dictionary Resources 153

Chapter 11: Syntax for Assembler Macros 161

Chapter 12: Notes on Security Statement Syntax 225

Chapter 13: Syntax for Securing Global Resources 231

Chapter 15: Syntax for Securing Database Resources 295

Chapter 16: Syntax for Security Display Statements 339

Chapter 17: DISPLAY/PUNCH ALL Syntax for Security Definitions 367

Appendix A: Security Macro JCL 379

Appendix B: Security Database Information and DSECTs 389

Appendix C: Privileges Required for Statements 447

Appendix D: User-Defined System Security Rules 457

Index 469

Chapter 1: Introduction

This guide provides information about installing and maintaining centralized security for CA IDMS resources. This guide is intended for the security administrator of the site.

The administrator should be familiar with concepts of CA IDMS/DB, CA IDMS/DC, and CA IDMS DC/UCF.

Note: If CA Top Secret for z/OS or CA ACF2 for z/OS is configured to protect CA IDMS resources for a release previous to CA IDMS Release 12.0, see the *CA IDMS Conversion Notebook* for information about security system conversion.

This document is both a user guide and a reference manual.

- Chapters 1-10 comprise the user guide portion of the document.

  Important: It is strongly recommended that you read Chapters 1-10 before designing and implementing security for CA IDMS.

- Chapters 11-17 contain reference information including security macro syntax.
- Appendixes A-D provide additional reference and usage information, including security macro JCL and security database record descriptions.

Note: This guide uses the terms CA IDMS/DB, CA IDMS/DC, CA IDMS UCF, DC/UCF, and CA IDMS DDS to identify specific CA IDMS components only when it is important to your understanding of the product.

This section contains the following topics:

Syntax Diagram Conventions

The syntax diagrams presented in this guide use the following notation conventions:

`UPPERCASE OR SPECIAL CHARACTERS`

Represents a required keyword, partial keyword, character, or symbol that must be entered completely as shown.

`lowercase`

Represents an optional keyword or partial keyword that, if used, must be entered completely as shown.

`italicized lowercase`

Represents a value that you supply.

`lowercase bold`

Represents a portion of the syntax shown in greater detail at the end of the syntax or elsewhere in the document.

```
◄─
```

Points to the default in a list of choices.

```
►►──────────────────
```

Indicates the beginning of a complete piece of syntax.

```
──────────────────►◄
```

Indicates the end of a complete piece of syntax.

```
──────────────────►
```

Indicates that the syntax continues on the next line.

```
►──────────────────
```

Indicates that the syntax continues on this line.

```
──────────────────►─
```

Indicates that the parameter continues on the next line.

```
─►──────────────────
```

Indicates that a parameter continues on this line.

```
►── parameter ───────►
```

Indicates a required parameter.

```
►───┬ parameter ┬──────►
    └ parameter ┘
```

Indicates a choice of required parameters. You must select one.

```
►───┬───────────┬──────►
    └ parameter ┘
```

Indicates an optional parameter.

```
►───┬───────────┬──────►
    ├ parameter ┤
    └ parameter ┘
```

Indicates a choice of optional parameters. Select one or none.

```
   ┌───────────┐
►─▼─ parameter ┴──────►
```

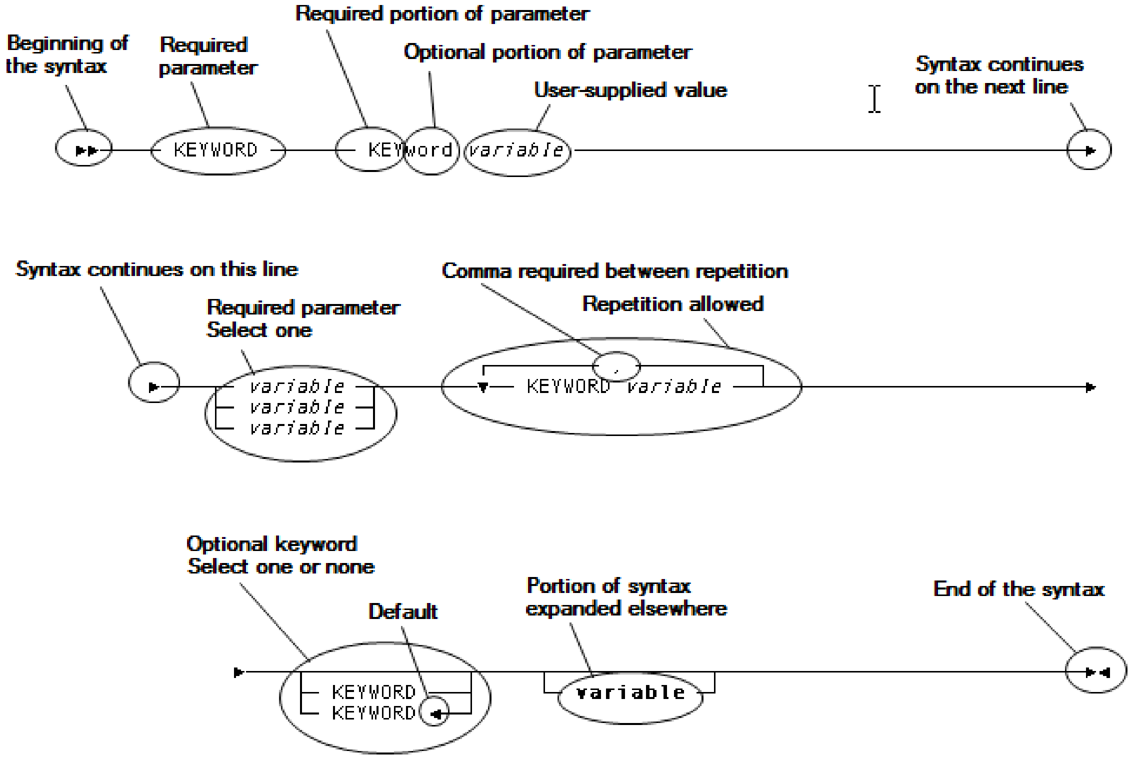
Indicates that you can repeat the parameter or specify more than one parameter.

```
   ┌──── , ──────┐
►─▼─ parameter ──┴──────►
```

Indicates that you must enter a comma between repetitions of the parameter.

Sample Syntax Diagram

The following sample explains how the notation conventions are used:


Required portion of parameter
Beginning of the syntax
Required parameter
Optional portion of parameter
User-supplied value
Syntax continues on the next line
KEYWORD
KEYword
variable
Syntax continues on this line
Comma required between repetition
Required parameter Select one
Repetition allowed
variable
variable
variable
,
KEYWORD variable
Optional keyword Select one or none
Default
Portion of syntax expanded elsewhere
End of the syntax
KEYWORD
KEYWORD
variable

Chapter 2: CA IDMS Centralized Security Overview

This section contains the following topics:

Security Administration

In the data processing environment, security administration, whether performed by a full-time security administrator in a large shop or by a DBA in a small shop, is a vital component of corporate success.

Why Secure Your System?

You secure your system to do the following:

- Protect confidential information from deliberate or accidental exposure
- Maintain the integrity of your corporate databases and dictionaries
- Prohibit or deter unauthorized access
- Meet corporate and departmental security standards
- Fulfill government and Department of Defense (DOD) requirements
- Adhere to privacy laws

Security Strategy

A comprehensive corporate security strategy must account for all types of physical and electronic access to systems, including the following:

- Physical access to the computer room
- Electronic access to the computer room (such as by dial-in lines)
- Access to hardware
- Access to software
- Access to corporate databases
- Access to applications

- Access to production, test, and quality assurance (QA) systems
- Access to data sets

A strategy that has not considered all types of authorized and unauthorized access is open to intentional and accidental corruption.

Installing and Implementing Security

A security system needs to be installed and then implemented. A common approach to organizing this process is to designate the following:

- A security administrator who manages and coordinates overall information security for the site
- A team that helps the security administrator plan for, install, and implement the security system

CA IDMS Centralized Security Administration

The security approach described in the previous section is well suited for CA IDMS centralized security administration. Three important reasons are as follows:

- CA IDMS centralized security can interface with an external security software system to protect CA IDMS resources.
- All CA IDMS authorities derive from the absolute authority of the security administrator.
- You can easily delegate to DCAs and DBAs the set of authorities they need to administer security on resources that are specific to systems and databases.

CA IDMS Centralized Security

This section describes the purposes of centralized security and defines the CA IDMS security domain.

The purposes of CA IDMS centralized security are to provide the following:

- A system for protecting CA IDMS resources when an external security system is not available or not used to protect CA IDMS resources.
- A system for protecting CA IDMS resources that is not administered or enforced with user exits.
- A system that can interface with an external security system, such as CA Top Secret for z/OS or RACF®, for the protection of some CA IDMS resources.

Protection of resources external to CA IDMS is not a purpose of CA IDMS centralized security.

Security Domain

The CA IDMS security domain is the set of CA IDMS DC and UCF systems and local mode jobs that share a set of user definitions.

If you specify that CA IDMS user validation is to be performed by an external security system, the CA IDMS security is the corporate security domain. If user validation is performed by CA IDMS internal security, the CA IDMS security domain is the set of DC systems that share a user catalog, the repository of CA IDMS user definitions.

Note: For more information about CA IDMS user validation, see Signon Processing (see page 57).

Architecture

This section describes the features and architecture of CA IDMS centralized security.

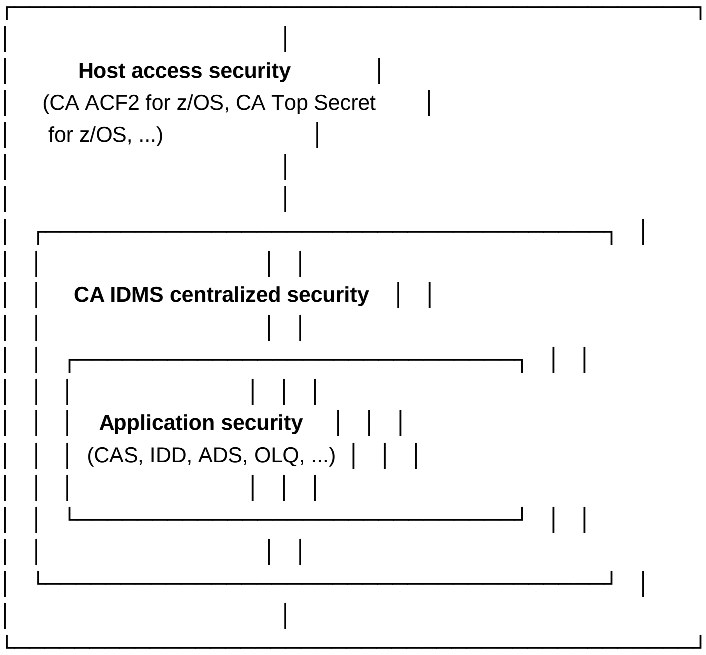
Architectural Features

CA IDMS centralized security architecture includes the following features:

- Full integration with CA IDMS software
- Availability to both system-supplied and user-written applications executing under the CA IDMS central version, in local mode, and through supported front-end software such as CICS and TSO in the CA IDMS environment
- Support for a distributed, client/server environment
- Compatibility with the Command Facility tool used for CA IDMS data definition
- ANSI-compliant security syntax where possible

Multi-layered Security Scheme

In combination with application security for dictionary resources, this architecture offers multi-layered security for CA IDMS systems, as shown in the following illustration:


Host access security
(CA ACF2 for z/OS, CA Top Secret
for z/OS, ...)
CA IDMS centralized security
Application security
(CAS, IDD, ADS, OLQ, ...)

The SRTT

This section describes the Security Resource Type Table (SRTT).

Purpose of the SRTT

The Security Resource Type Table (SRTT) is a load module in which you store this information that CA IDMS centralized security needs at runtime:

- Each resource type to be secured.
- The system that will enforce security on the resource (internal or external).
- For resources to be secured externally, information that the external security system needs to service a security check request on the resource.

Using the SRTT

At installation, CA IDMS provides the default RHDCSRTT module. The security option for each resource defined by CA IDMS is set to 'OFF' (no security), and SVCNUM is set to the SVC number specified in the installation parameters.

You modify the RHDCSRTT module with an assembly of the #SECRTT macro. The SRTT is loaded at system start-up, and can be reloaded dynamically using the DCMT VARY NUCLEUS command for the RHDCSRTT module.

The scope of the SRTT is one or more CA IDMS systems, depending on your security scheme.

Security Options by Resource Type

For each resource type you can specify one of these security options in the SRTT:

- EXTERNAL—Security enforcement for the resource follows rules defined in the external security system.
- INTERNAL—Security enforcement for the resource follows rules defined to the CA IDMS internal security system.
- OFF—No security enforcement for the resource (the default at installation).

Generating the SRTT

The SRTT is created by issuing a sequence of #SECRTT macros.

The #SECRTT macro can specify one of four types of action:

- **Initial**—Denotes the beginning of the SRTT assembly.
- **Entry**—Specifies a security option for *all* occurrences of a given resource type.
- **Occurrence override**—Specifies a security option for one or more occurrences of a given resource type that overrides the entry specification for the resource type (not applicable to all resource types).
- **Final**—Denotes the end of the SRTT assembly.

Occurrence Overrides

For an individual occurrence of a database, task, or program, you can override in the SRTT the security option you specify for the corresponding resource type.

For example, you can omit an SRTT entry for the database resource type and create an occurrence override for database PROD specifying internal security. The result will be that any checks on resources associated with database PROD will be routed to internal security, while checks on resources associated with all other databases will not be checked.

Note: The resource name you give on an occurrence override is treated as a wildcard. In the previous example, all databases in the domain of the SRTT with names beginning 'PROD' are secured internally.

Performance Consideration

You may gain a performance advantage by using an override to turn off security for occurrences of a secured resource type. Runtime security processing checks for an occurrence override in the SRTT before checking resource authorizations in the security database.

External Security Enforcement

External Security Specifications

In each SRTT entry that specifies external security, you define the format of the resource name that will be routed to the external security system in a security check. At runtime, the central security interface uses this information to map the IDMS internal resource name to the external resource name before routing the request for a security check to the external system.

Standard Security Interface

CA IDMS centralized security uses IBM’s System Authorization Facility (SAF) as the interface to external security systems. On a security check, IDMS centralized security issues RACROUTE calls providing the names of resource type and resource occurrence being checked and the keyword that equates to the authority needed.

Security Definitions

You do not need a user catalog if you plan to protect *all* CA IDMS resources with an external system. All required definitions would reside in the external security system. The one requirement within CA IDMS would be to build the SRTT with external security specifications for all secured resources.

However, user definitions and user profile definitions are accessed during signon processing if they exist, regardless of how signon is secured. Therefore, you may wish to use the user catalog even if all security checks are routed to the external system. For more information about profiles in signon processing, see Securing User Profiles (see page 90).

Internal Security Enforcement

CA IDMS Internal Security

When an entry for a resource type in the SRTT specifies internal security, only users and groups who have been defined in the user catalog can be granted the privilege to access the resource. You grant privileges with the appropriate GRANT statements.

CA IDMS Internal Security Administration

You can delegate internal security administration by granting to selected users:

- Administration privileges
- The privilege to grant their privileges to other users

You perform CA IDMS internal security administration functions by issuing security authorization statements through the CA IDMS Command Facility. When you do this, CA IDMS does the following:

1. Accepts syntax that specifies the security administration request (for example, a CREATE USER statement).
2. Verifies that you, the issuer of the request, have the authority to issue the request.
3. Updates the data in the appropriate internal security repository.

Security Definitions

Required for Security Checks

When CA IDMS centralized security receives a request for a security check, it first determines from the SRTT whether the resource is secured. If it is, centralized security routes the request to CA IDMS internal security or the external security system, depending on the security option specified for the resource in the SRTT.

The resulting security check accesses security definitions of resources and resource authorizations to determine whether the executing user has the authority to access the resource in the way that is indicated on the security request.

Resource Definitions

Securable resources are the entities in the CA IDMS environment defined by CA IDMS or the user to which you control access.

Securable resources defined by CA IDMS are as follows:

- User
- Group
- User profile
- System
- Signon
- System profile
- Application activity
- Queue (1)
- Access module (runtime) (1)
- Load module (loadable entity) (1)
- Program (load module) (1)
- Task (1)
- Database
- Area (2)
- Run unit (1) (2)

- Access module (definition) (2)
- Non-SQL schema (2)
- SQL schema (2)
- Table (2)
- Database name table
- DMCL

(1) Occurrences of this resource can be grouped in a category using the CREATE RESOURCE statement.

(2) This resource type is secured automatically when the database resource type is secured.

Authorization IDs

An authorization ID identifies a user or group whom you can authorize to access resources.

If the security option for the resource is external, the authorization ID (and the authorities given to it) are defined in the external security system.

If the security option for the resource is internal, the authorization ID (and the privileges granted to it) are defined in the internal security system. You define authorization IDs to CA IDMS with the CREATE USER and CREATE GROUP statements.

Resource Authorization

An authority is the ability to access a resource in a particular way. A resource authorization is an authority that is associated with a resource definition and an authorization ID.

If the security option for a resource is external, resource authorizations are specified in the external security system.

If the security option for a resource is internal, resource authorizations are specified in the internal security system by **granting privileges**. You give users the privilege to access a resource with a GRANT statement, and you take away the privilege with a REVOKE statement.

Runtime Security Processing

Security Checking

The CA IDMS centralized security facility handles all security checks issued during CA IDMS processing. Security requests are routed to the central security interface to provide uniform validation of requests.

Security Enforcement

The security option that is specified in the SRTT for the resource type or resource type occurrence determines how security is enforced.

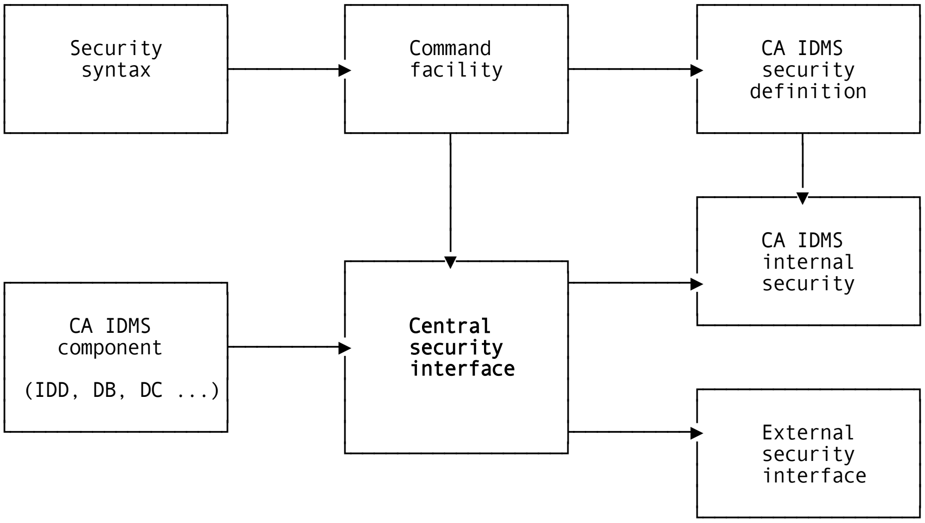
Example: You can control the execution of tasks with the external security system, but control access to a particular database with CA IDMS internal security.

If the security option for the resource being checked is external, the request is routed to the external security system. The external security system returns a value to the centralized security interface representing the result of the check.

If the security option for the resource being checked is internal, CA IDMS centralized security verifies that the user holds the required permission.

Centralized Security Diagram

This illustration shows the flow of processing in the CA IDMS centralized security system:


Security syntax
Command facility
CA IDMS security definition
CA IDMS internal security
CA IDMS component
(IDD, DB, DC ...)
Central security interface
External security interface

Security Application Programming Interface

What You can Do

A user-written application in Assembler language can issue security requests to CA IDMS centralized security. All requests pass a Security Request Block (SRB). A request can be issued from a user mode or a system mode application.

Security Macros

You can issue a security request using the following:

- #SECHECK—To check authorization for accessing a resource.
- #SECSGON—To validate the user and sign the user on.
- #SECSGOF—To sign the user off.

Note: For more information about how to use security macros, see the following sections:

- #SECHECK (see page 195)
- #SECSGON (see page 218)
- #SECSGOF (see page 215)

Chapter 3: Activating CA IDMS Security

This section contains the following topics:

Installation Defaults

Security at Installation

The RHDCSRTT module provided with installation of CA IDMS contains an initialized entry for each resource defined by CA IDMS.

The initial security option for each resource is 'OFF'. This means that at installation, there is no security enforcement through CA IDMS centralized security. Any user can create security definitions and grant privileges. Security information is stored in the security databases but has no effect until security is activated. Furthermore, the SVCNUM parameter is set to the SVC number that has been specified in the installation parameters.

Note: The identifier of the user who performed the installation is recorded as the owner of the demonstration database schemas in the schema definitions. Schema ownership affects security processing if security for the DB resource is subsequently activated.

Preserving the Initialized RHDCSRTT

You should preserve a secure copy of the initialized RHDCSRTT module provided at installation. This ensures that after you activate security, you can turn it off again simply by using the initialized RHDCSRTT module.

To preserve the initialized RHDCSRTT, you n take one of these steps:

- Create a backup copy of the initialized RHDCSRTT module that is provided in the installed CA IDMS load library. Store the backup copy in a secure load library and restore it to the CA IDMS installed load library if needed.
- Link your modified SRTT into a different load library during initial testing. To activate security, concatenate the load library containing the new SRTT with the CA IDMS load library. To deactivate security, remove the load library containing the new SRTT from the STEPLIB/CDMSLIB concatenation, or rename the RHDCSRTT load module.

Planning the Security Scheme

Getting Started

The basic questions in planning your CA IDMS security scheme are:

- What resources will be secured?
- For secured resources, will enforcement be external or internal?
- Will the same security option apply to a given resource across the domain?

To help you answer these questions, you should become familiar with the information in the following chapters:

- Using CA IDMS Internal Security (see page 65)
- Securing Database Resources (see page 117)

Creating Security Definitions

External Security Definitions

If security enforcement for a CA IDMS resource is external, you do not need to store information about the resource in the CA IDMS security database.

Before you create the SRTT entries that contain external security information, be sure to complete the steps necessary to define resources and security rules for resources in the external security system. When you create the SRTT with entries that include SECBY=EXTERNAL, you activate external security enforcement for the resource types specified in those entries.

Note: For more information, see Using External Security (see page 45).

For external security checking, you must specify in the #SECRTT entry the format of the resource name that centralized security should forward to the external security system. You construct the external resource name format in the SRTT to match the format you have specified for the resource in the external security system.

Note: For more information, see Constructing an External Resource Name (see page 46).

Internal Security Definitions

To secure one or more resources internally, you must define users to the user catalog and grant privileges on the resources to users. You can create any or all of these definitions before activating security in the SRTT. Remember, however, that until you activate security for CA IDMS administration privileges, any users signed on to the system can also manipulate internal security definitions (and any user *can* sign on until signon is secured).

As you plan the sequence of creating the security definitions needed for your security scheme, be aware of these considerations:

- You can add a user to a group in one of these ways:
  - Using the GROUP parameter of the CREATE/ALTER USER statement.

    This specifies the user's default group. Only one group can be specified.
  - Using the ADD USER clause of the CREATE/ALTER GROUP statement.

    You can add the user to any number of groups in this way. You must first create the user.
- You associate a user profile with a user in the CREATE/ALTER USER statement. You can do this before you define the profile; the information is stored, and a warning is issued.
- Profiles may be associated with user identifiers but not group identifiers.
- The scope of a user profile is the domain, but a system profile, like the grant of signon privilege which associates a system profile with a user, is system-specific.
- You can grant privileges on resources defined by CA IDMS to authorization IDs before the authorization IDs are defined; the information is stored, and a warning is issued.
- You can grant privileges on resources defined by CA IDMS to authorization IDs before the resources are defined *except* for resources specified in the CREATE RESOURCE statement (systems, activities, and categories).
- If you create groups corresponding to sets of privileges that you will grant, the number of CREATE GROUP statements required is likely to be much less than the number of GRANT statements to users that you would otherwise issue.
- In general, it is easier to restrict grants of privilege only to groups because it is easier to grant a set of privileges implicitly to a user by adding the user to the group than it is to grant each privilege individually to the user.

- A group may consist of only one user.
- The group PUBLIC is an authorization ID (defined by the system the first time a privilege is granted to it) to which all users belong by default. An appropriate step to consider is transferring ownership of the demonstration database schemas to PUBLIC.
- Categories allow you to group system resources so that you can grant execution privilege on the category (multiple resources) to a user or a group (multiple users) with one statement.
- A resource occurrence may participate in only one category.
- In general, you ease the administration of internal security if you establish a 1:1:1 correspondence among groups, categories, and execution privileges.
- When you create application activities, name the activities in a way that will allow you to use a wildcard in grants of execution privilege on activities.

  Note: For more information, see Using a Wildcard (see page 76).
- If you plan to activate security for the system dictionary and the user catalog, consider doing it with DB occurrence overrides.

  Note: For guidance in planning security for the system dictionary and the user catalog, see Securing the Dictionaries and the User catalog (see page 149).

Activating Security

Planning to Activate Security

Before you modify the initialized SRTT, do the following:

- Designate one system for testing security processing.
- Plan the sequence of activating security options so that you can activate and test one at a time.
- Consider activating the SGON (signon) resource first.
- If you plan to secure signon externally and one or more other resources internally, activate security for administration privileges before you grant signon to users who will not hold administration privileges.

External Security for Signon Processing

To activate external security for signon processing, the #SECRTT assembly must include an entry for resource type SGON.

Note: For more information, see External Signon Security (see page 50).

Planning to Activate Internal Security

Before you activate internal security, you should:

1. Verify that the user catalog (SYSUSER.DDLSEC) area is specified in the startup JCL for the test system.

   If the default access mode to the area, as specified in the DMCL, is not UPDATE, then you must issue a DCMT VARY AREA statement before you update the user catalog.

   Note: For more information about DCMT commands, see the *CA IDMS System Tasks and Operator Commands Guide*.

2. At the least, create the following definitions:

   - Users—defining the users who will hold SYSADMIN privilege is sufficient to begin. Creating a SYSADMIN group is recommended.
   - Signon and SYSADMIN privileges granted to the designated users.

     If you now activate internal security, the designated users will be able to sign on to the CA IDMS system and to administer the security system.

Activating Internal Security

If you have granted signon and SYSADMIN privileges, the logical first step to activate internal security is to secure the signon and SYSADMIN resources. You do this by including these entries in the #SECRTT assembly:

```
#SECRTT TYPE=ENTRY,                    X
   RESTYPE=SYSA,                    X
   SECBY=INTERNAL

#SECRTT TYPE=ENTRY,                    X
   RESTYPE=SGON,                    X
   SECBY=INTERNAL
```

Securing Security Definitions

A knowledgeable user can access security definitions with local mode access to the user catalog or system dictionary. You can prevent this access by securing these entities as databases and granting privileges on categories of run units.

Note: For more information, see Securing the Dictionaries and the User Catalog (see page 149).

How to Generate the SRTT

The #SECRTT Macro

You generate a new SRTT by using the #SECRTT macro:

1. The first #SECRTT macro initializes SRTT values for all CA IDMS resources.

   The value of the TYPE parameter on the first macro must be INITIAL.

2. One or more additional #SECRTT macros to override initial values.

   The value of the TYPE parameter on these macros must be ENTRY if the security option is for a resource type (or OCCURRENCE if the security option is for an individual occurrence of a database, task, or program).

   If you specify SECBY=EXTERNAL, you must also specify the following:

   - The external resource class that you have defined in the external security system as the equivalent of the CA IDMS resource type.
   - The external resource name format that you have defined in the external security system for identifying the CA IDMS resource.

3. The final #SECRTT indicates that the table is to be generated with values as specified by the preceding macros in the series.

   The value of the TYPE parameter on the last macro must be FINAL.

#SECRTT Assembly

The table is generated and linked only if each #SECRTT statement in the series assembles without error. If one or more statements receives an error, only a listing results.

Note: For complete documentation of the #SECRTT, see #SECRTT (see page 204).

The RHDCSRTT module should be linked into a secure dataset to prevent unwarranted access to or manipulation of the security system.

#SECRTT Macro Example

```
#SECRTT TYPE=INITIAL,                   X
    ENVNAME=TEST,                     X
    SVCNUM=235

#SECRTT TYPE=OCCURRENCE,                      X
    RESTYPE=DB,                     X
    RESNAME='CUSTDB',                   X
    SECBY=INTERNAL

#SECRTT TYPE=ENTRY,                      X
    RESTYPE=SYST,                    X
    SECBY=INTERNAL

#SECRTT TYPE=ENTRY,                      X
    RESTYPE=SGON,                     X
    SECBY=EXTERNAL,                    X
    EXTCLS='SYSTEM',                  X
    EXTNAME=(RESNAME)

#SECRTT TYPE=ENTRY,                      X
    RESTYPE=TASK,                    X
    SECBY=INTERNAL

#SECRTT TYPE=ENTRY,                      X
    RESTYPE=SPGM,                     X
    SECBY=EXTERNAL,                    X
    EXTCLS='PROGRAM',                   X
    EXTNAME=(RESNAME)

#SECRTT TYPE=OCCURRENCE,                      X
    RESTYPE=SPGM,                     X
    RESNAME='RHDCBYE',                   X
    SECBY=OFF

#SECRTT TYPE=FINAL

END
```

Notes on the Example

- TYPE=INITIAL begins the SRTT definition.
- ENVNAME=TEST provides a default name qualifier that you can use in constructing an external resource name.
- SVCNUM=235 specifies the SVC number defined at install time.

- TYPE=OCCURRENCE signifies an occurrence override.

  In this case:

  - Resource checks for database CUSTDB and associated schemas, tables, and access modules will be processed by internal security; all other databases and their resources are unsecured.
  - Resource checks for program RHDCBYE will not be routed to the external system, but resource checks for all other programs will be routed to the external security system.

- Resource checks on system resources (RESTYPE=SYST) will be processed by internal security, but signon processing (including password validation) will be processed by the external security system.
- TYPE=FINAL ends the SRTT.

Note: For more information about #SECRTT usage and #SECRTT syntax, see the chapter Syntax for Assembler Macros (see page 161).

Dynamic Security Refresh

You can make changes to your security scheme and then activate those changes without cycling a CV. After changing security definitions using the #SECRTT macro and reassembling the RHDCSRTT module, you issue existing DCMT commands to vary the RHDCSRTT nucleus module to new copy and reload it.

Benefit

You can respond to changes in your security environment without bringing down a system and cycling a CV. For example, you can change the security mapping for a resource type or you can make changes to category and activity definitions.

What Gets Refreshed

When you reload the RHDCSRTT module, the following security definitions are refreshed and any changes you made to them are immediately implemented:

- Access module table
- Category tables
- Activity and category bit map tables

Signon Security Changes Not Immediately Implemented

Signon and system group security definitions *are not* refreshed when RHDCSRTT is reloaded; users signed on to the system remain signed on even after the reload. Any changes made to signon and system group security for users signed on to a system when a reload is done, do not take place until those users sign off of the system and then sign on again.

Example

After you change a security scheme and modify the RHDCSRTT module, perform the following to activate the changes:

- Issue the DCMT VARY NUCLEUS syntax to vary module RHDCSRTT to new copy.
- Issue the DCMT VARY NUCLEUS RELOAD command to reload the changed (new) RHDCSRTT nucleus module.

The following example shows these commands.

```
dcmt vary nucleus module rhdcsrtt n c

    VARY NUCLEUS MODULE RHDCSRTT NEW COPY
IDMS DC283001 V104 USER:ABBTH01  NUCLEUS MODULE RHDCSRTT MARKED TO NEW COPY
```

```
    VARY NUCLEUS RELOAD
IDMS DC283003 V104 USER:ABBTH01  NUCLEUS MODULE RHDCSRTT RELOADED
IDMS DC283004 V104 USER:ABBTH01  CSA/NUCLEUS VECTOR TABLE UPDATED
FOR NUCLEUS MODULE RHDCSRTT
IDMS DC283007 V104 USER:ABBTH01  SECURITY TABLES REFRESHED SUCCESSFULLY
```

Note: For more information about the DCMT VARY NUCLEUS command, see the *CA IDMS System Tasks and Operator Commands Guide*.

Chapter 4: Using External Security

This section contains the following topics:

SRTT Requirements

Essential to the Security System

The SRTT is the essential foundation of the CA IDMS security system because a resource is unsecured unless security for it is specified in the SRTT.

To secure a resource externally, you must include information in the SRTT that identifies the resource to the external system. This information must include an external resource class and an external resource name.

For external security, you do not need to create any resource definitions within CA IDMS itself.

SRTT Entries for External Enforcement

You maintain the following information in the SRTT about resources that are secured externally:

- **Resource type**—A keyword representing a type of resource, such as program, table, or database.

  Certain keywords are reserved for resource types defined by CA IDMS. You can specify any one- to four-character keyword to define your own resource type as long as the meaning of and rules for the resource type are defined in your external system.

  Note: For keywords reserved by CA IDMS, see #SECRTT (see page 204).

- **Security option**—Always EXTERNAL, specified in the SECBY= parameter.

- **External resource class**—The name of the resource type as defined in the external security software.

- **Resource name** (optional)—A specific occurrence of a resource type (resource types database, task, and program only).
- **External resource name format**—The format of the resource name as defined in the external security software.
- **Environment name** (optional, specified on the initial #SECRTT macro)—The name of a CA IDMS processing environment to be associated with the resource.

Specifying External Resource Class and Name

An external security check on a resource occurrence depends upon an external resource class and external resource name supplied on the *entry* for the resource type in the SRTT. External resource classes and names specified on occurrence overrides are ignored by the runtime system.

Therefore, you must create an SRTT entry with the external resource class and name for a resource type whether you are securing all occurrences of the resource type externally or only some occurrences.

In the following example, an SRTT entry for tasks is created even though the specified security option is 'OFF'. The purpose of the entry is to provide information needed to perform an external security check on the OPER task, for which external security is specified in the occurrence override that follows.

```
#SECRTT TYPE=ENTRY,                     X
   RESTYPE=TASK,                    X
   SECBY=OFF,                     X
   EXTNAME=(RESTYPE,RESNAME)                X
   EXTCLS='IDMSTASK'

#SECRTT TYPE=OCCUR,                     X
   RESTYPE=TASK,                    X
   RESNAME='OPER',                  X
   SECBY=EXT
```

Constructing an External Resource Name

How You Do It

As in the previous example, the external resource name format is specified in the EXTNAME parameter of the #SECRTT macro. In this parameter, you list keywords to represent the fields that comprise the external resource name format.

Note: For complete documentation of the macro, see #SECRTT (see page 204).

If you do not specify the EXTNAME parameter, the external resource name by default consists of only the name of the base resource on the security request.

Runtime Usage of EXTNAME Values

At runtime, the external resource name is constructed using the values in the current security request that correspond to the keywords you specified in the EXTNAME parameter.

For example, if the SRTT entry for resource type TABL (table) includes an EXTNAME parameter that specifies (ENVIR,RESTYPE,SCHEMA,RESNAME), the external resource name format for a table is:

environment-name.TABL.*schema-name*.*table-name*

The following example represents the actual resource name sent to the external security system using values from the current security request:

PROD.TABL.USA.EMPLOYEE

Order of Name Fields

The order of the fields in the external resource name passed to the external security system is determined by the order of the keywords that you list on the EXTNAME parameter of the #SECRTT entry. For a given set of fields, you can specify any possible order to format the external resource name.

The format of the external resource name defined in the EXTNAME parameter of the SRTT must match the format used to identify the resource in the external security system.

Environment Name Qualifier

The environment name qualifier is significant only when security is external. You specify environment name on the initial #SECRTT macro.

The environment name distinguishes resources in the domain of the current SRTT from like-named resources in the domain of another SRTT. You specify an environment name if such distinctions are necessary to your security scheme.

For example, you can specify PROD as the environment name in the SRTT that governs production systems. This means that you can qualify the external resource names of resources in production systems with PROD and specify rules for them in the external security system that are different from like-named resources in test systems, which may have different environment names or no environment name.

Thus, if you have a database named EMPDB in both the test and production environments, you can write a security rule in the external system that is applied only when the security check is for EMPDB qualified by 'PROD' (that is, PROD.EMPDB if the external resource name is *environment-name*.*database-name*, or EMPDB.PROD if the external resource name is *database-name*.*environment-name*).

External Resource Name Keywords

This table presents the keywords that you can specify in the EXTNAME parameter of the #SECRTT macro (the required characters appear in upper case) and the value from the current security request that corresponds to each keyword:

EXTNAME keyword	Value from current security request
ACTIvity	Concatenation of application name and application function number. For more information, see #SECRTT (see page 204).
APPLname	*application-name*
DBNAme	*database-name*
DDNAme	*dd-name*
ENVIr	*environment-name*
RESName	The name of the resource occurrence(1)
RESType	The resource type keyword, from the SRTT (for example, SLOD)
SCHEma	*schema-name* (SQL)
SSNAme	*subschema-name*
SYSTem	*system-identifier*
VERSion	*version-number*

Note: (1) For RESTYPE ACTI, the RESNAME value is *application-name*.

You can always specify the RESNAME, RESTYPE, and ENVIR keywords in formatting the external resource name. The tables that follow indicate the values of RESNAME and RESTYPE for each resource type and the other keywords available in constructing an external resource name for the resource type.

Naming Global Resources

This table presents the keywords that you can use to construct external resource names for global resources:

Resource	RESNAME	RESTYPE	Other available keywords
SYSADMIN	@RESERVED@	SYSA	
User	*user-identifier*	USER	
Group	*group-identifier*	GROU	

Resource	RESNAME	RESTYPE	Other available keywords
User profile	*profile-name*	UPRF	

Naming System Resources

This table presents the keywords that you can use to construct external resource names for system resources:

Resource	RESNAME	RESTYPE	Other available keywords
DCADMIN	@RESERVED@	DCA	
System	*system-identifier*	SYST	
Signon	*system-identifier*	SGON	
System profile	*profile-name*	SPRF	
Activity	*application-name*	ACTI	APPLname,ACTIvity
Task	*task-code*	TASK	SYSTem
Load module	*load-module-name*	SLOD	DBNAme,VERSion
Queue	*queue-name*	QUEU	SYSTem
Access module	*access-module-name*	SACC	DBNAme,SCHEma
Program	*program-name*	SPGM	SYSTem,DDNAme

Naming Database Resources

This table presents the keywords that you can use to construct external resource names for database resources:

Resource	RESNAME	RESTYPE	Other available keywords
Database	*database-name*	DB	
Area	*area-name*	AREA	DBNAme
Rununit	*program-name*	NRU	DBNAme,SSNAme
SQL schema	*schema-name*	QSCH	DBNAme
Non-SQL defined schema	*nonsql-schema-name*	NSCH	DBNAme,VERSion
Access module	*access-module-name*	DACC	DBNAme,SCHEma

Resource	RESNAME	RESTYPE	Other available keywords
Table	*table-name*	TABL	DBNAme,SCHEma
DMCL	*dmcl-name*	DMCL	
Database name table	*database-table-name*	DBTB	

Note: There is no resource type keyword for DBADMIN privilege.

Naming Examples

This example presents the possible combinations of external resource name fields for a DC task. The actual number of fields that you specify depends on how the resource name is defined in the external security system:

environment-name.TASK.*system-identifier.task-code*
environment-name.TASK.*task-code*
environment-name.system-identifier.task-code
environment-name.task-code
TASK.*system-identifier.task-code*
TASK.*task-code*
system-identifier.task-code
task-code

Defining External Signon Security

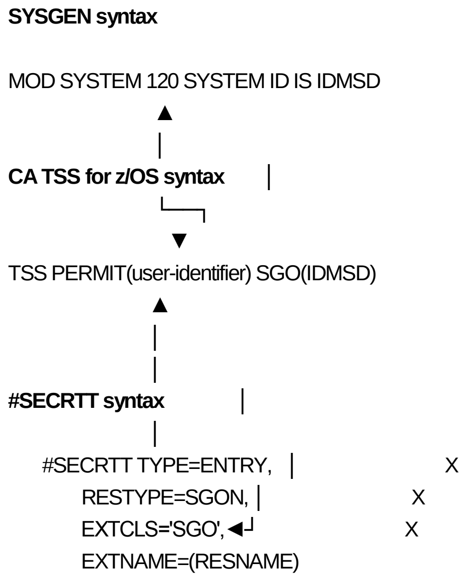
Defining SRTT Entries

To secure system signon externally, add an entry to the SRTT for the signon (SGON) resource type.

The applicable resource name for the signon resource type is *system-identifier*; it matches the value in the SYSTEM ID parameter of the system generation SYSTEM statement. Thus, the name of the resource defined in the external system must match the system identifier.

The following examples for CA TSS and CA ACF2 show the relationships that must exist between the system identifier in system generation and the resource identifier in the external security definition; and between the resource class in the external security definition and the external class in the SRTT entry.

Example for CA Top Secret (TSS)



SYSGEN syntax

```
MOD SYSTEM 120 SYSTEM ID IS IDMSD
```

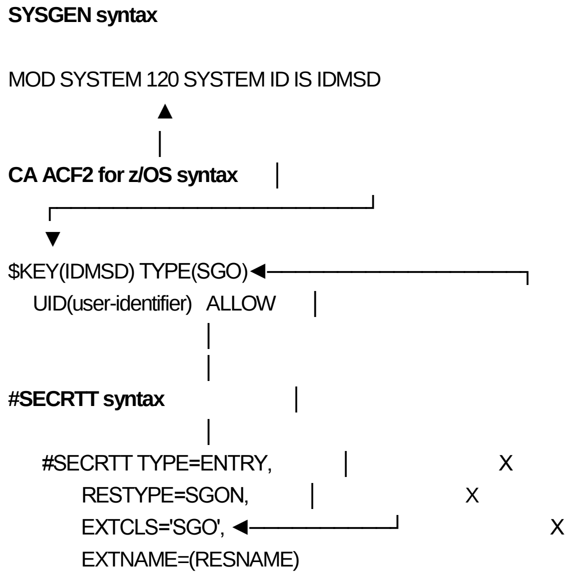
CA TSS for z/OS syntax

```
TSS PERMIT(user-identifier) SGO(IDMSD)
```

#SECRTT syntax

```
    #SECRTT TYPE=ENTRY,                    X
          RESTYPE=SGON,                    X
          EXTCLS='SGO',                    X
          EXTNAME=(RESNAME)
```

Example for CA ACF2



SYSGEN syntax

```
MOD SYSTEM 120 SYSTEM ID IS IDMSD
```

CA ACF2 for z/OS syntax

```
$KEY(IDMSD) TYPE(SGO)
   UID(user-identifier)   ALLOW
```

#SECRTT syntax

```
    #SECRTT TYPE=ENTRY,                    X
          RESTYPE=SGON,                    X
          EXTCLS='SGO',                            X
          EXTNAME=(RESNAME)
```

Optionally Defining PassTickets

PassTickets can be used as an alternative to a password.

Note: For more information about PassTickets, see Signon Processing (see page 57).

To use PassTickets for externally secured signon, add PassTicket definitions to the particular external security system being used. Depending on the external security system in use, these definitions can include:

- Defining a resource class for PassTickets.
- Granting ownership of the resource used for PassTickets.
- Defining a session key for each application for which PassTickets are used.
- Granting permission to a user to the resource used for PassTicket validation.

Determining Applid

The applid specified in the definition of the PassTicket to the external security system is a unique identifier for the IDMS CV system. It is composed of the first VTAM line defined to the system. If no VTAM lines exist, it is composed of the system nodename.

See the following sections for examples of the external security definitions needed to allow PassTicket use.

Example for CA Top Secret (CA TSS)

This CA TSS example shows the external security definitions needed to allow PassTicket use:

1. Define the resource class PTKTDATA:

   ```
   TSS ADDTO(RDT) RESCLASS(PTKTDATA) ACLIST(ALL,READ,UPDATE) MAXLEN(37)
   ```

2. Add IDMSDEPT department ownership for resources of class PTKTDATA:

   ```
   TSS ADDTO(IDMSDEPT) PTKTDATA(IRRPTAUTH)
   ```

3. Add a session key for each applid (PSTKAPPL):

   ```
   TSS ADDTO(NDT) PSTKAPPL(IDMSSY73) SESSKEY(0123456789ABCDEF)
   TSS ADDTO(NDT) PSTKAPPL(IDMSSY74) SESSKEY(ABCDEF0123456789)
   ```

4. Add permission for JOHN_SMITH to generate and use a PassTicket for SYSTEM 73:

   ```
   TSS PERMIT(JOHN_SMITH) PTKTDATA(IRRPTAUTH.IDMSSY73.JOHN_SMITH)
   ACCESS(READ,UPDATE)
   ```

Example for CA ACF2

This CA ACF2 example shows the external security definitions needed to allow PassTicket use:

1. Define the CA IDMS PassTicket session key(s) and assign them to IDMS application IDs (or CV nodenames):

   ```
   SET PROFILE(PTKTDATA) DIVISION(SSIGNON)
   INSERT IDMSSY73 SSKEY(0123456789ABCDEF)
   INSERT IDMSSY74 SSKEY(ABCDEF0123456789)
   F ACF2,REBUILD(PTK),CLASS(P)
   END
   ```

2. Issue the following commands to assign PassTicket session key(s) for specific user(s) (here: JOHN_SMITH):

   ```
   ACFNRULE KEY(IRRPTAUTH) TYPE(PTK) ADD(IDMSSY73.JOHN_SMITH) UID(JOHN_SMITH)
   SERVICE(READ,UPDATE) ALLOW)
   F ACF2,REBUILD(PTK)
   ```

Example for IBM RACF

This IBM RACF example shows the external security definitions needed to allow PassTicket use:

1. Issue the following commands to activate the PassTicket class:

   ```
   SETROPTS CLASSACT(PTKTDATA)
   SETROPTS RACLIST(PTKTDATA)
   SETROPTS GENERIC(PTKTDATA)
   ```

2. Issue the following commands to define profile(s) for the IDMS application IDs (or CV nodenames) and specify the session key(s):

   ```
   RDEFINE PTKTDATA IDMSSY73 SSIGNON(KEYMASKED(0123456789ABCDEF)) UACC(NONE)
   RDEFINE PTKTDATA IDMSSY74 SSIGNON(KEYMASKED(ABCDEF0123456789)) UACC(NONE)
   ```

3. Issue the following commands to define profile(s) and enable UPDATE access to the IDMS PassTicket resource for specific user(s) (here: JOHN_SMITH):

   ```
   RDEFINE PTKTDATA IRRPTAUTH.IDMSSY73.JOHN_SMITH UACC(NONE)
   PERMIT IRRPTAUTH.IDMSSY73.JOHN_SMITH CLASS(PTKTDATA) ID(JOHN_SMITH)
   ACCESS(READ,UPDATE)
   RDEFINE PTKTDATA IRRPTAUTH.IDMSSY74.JOHN_SMITH UACC(NONE)
   PERMIT IRRPTAUTH.IDMSSY74.JOHN_SMITH CLASS(PTKTDATA) ID(JOHN_SMITH)
   ACCESS(READ,UPDATE)
   ```

4. Issue the following command to refresh the PTKTDATA class:

   ```
   SETROPTS RACLIST(PTKTDATA) REFRESH
   ```

External Signon Security Checking

External signon security checking consists of two phases:

1. Validation of the executing user in the external security system.
2. If the first phase is successful, a check on the user's authority to access the system identified in the current request.

Note: For more information, see Signon Processing (see page 57).

External Database Security Considerations

Before you secure a database externally, weigh the following considerations:

- If you add an SRTT entry that secures the DB resource type externally, you automatically secure a group of database resource types externally.

  Note: For more information, see Securing Database Resources (see page 117).

- If the SRTT contains one or more occurrence overrides that specify external security for resource type DB, you must also add an SRTT entry specifying external resource class and external resource name for each of the database resource types that are automatically secured externally.
- External security checks for database resources initiated by SQL bulk processing may have a discernible effect on processing time.

Note: For more information, see SQL Security Enforcement (see page 132).

Identifying Authorities to the External System

How It is Done

When a security check is issued for a resource that is secured externally, CA IDMS central security converts the authority that is being checked to a RACROUTE keyword. This table shows the RACROUTE keywords that may be used if security is external and the comparable privileges in CA IDMS internal security.

Type of authorities	RACROUTE keyword	CA IDMS privilege
Runtime authorities	CONTROL	DELETE

	UPDATE	INSERT
		UPDATE
		DBAWRITE

Type of authorities	RACROUTE keyword	CA IDMS privilege

	READ	SELECT
		EXECUTE(1)
		DBAREAD

Definition authorities	ALTER	REFERENCES
	READ	DISPLAY USE
	CONTROL	DROP
	UPDATE	ALTER
	CONTROL	CREATE
Administration authorities	ALTER	SYSADMIN DCADMIN DBADMIN

(1) No keyword is passed if EXECUTE is for a resource that can be categorized.

Chapter 5: Signon Processing

This section contains the following topics:

Signon Security Options

Installation Default

At CA IDMS installation, the security option for signon (the SGON resource in the SRTT) is 'OFF'.

This means that when the online user requests signon, or the first security check request is issued on behalf of the executing user in a local mode batch application, signon is unsecured and unvalidated. In an unvalidated signon, the user is successfully signed on whether or not the user ID and password have been defined.

Internal Signon Security

If you specify the internal security option for the SGON resource in an #SECRTT macro, signon is secured and password checking will be performed by the internal security system.

External Signon Security

If you specify the external security option for the SGON resource in a #SECRTT macro, signon is secured and password or PassTicket checking will be performed by the external security system.

If the external security system issues a failure or even a warning on user identification and validation, signon fails. This means that you cannot use internal security as a backup security system when you specify external security for signon.

Using Pass Tickets

The value of the password passed to the external security system and used for external signon authorization can be either a:

- password associated with the user in the external security system

  or

- a PassTicket.

A PassTicket is a temporary substitute for a password that is used for authentication for a specific application. A PassTicket is generated from values associated with the userid and application to which the signon is targeted and is valid for only a short period of time. PassTickets are often used as an alternative to sending a clear text password over a network, thereby improving network security.

Note: For externally secured signons, PassTickets are treated in the same way as passwords in the remainder of the signon processing description.

To enhance performance, CA IDMS uses password caching in multi-signon CV environments. Caching is also applied to PassTickets, as PassTickets are implemented by an external security system.

Under certain circumstances, this means that PassTickets can be used more than once. For this reason, CA recommends the implementation of the following points to maximize security when you implement PassTickets:

- Do not allow multiple signons
- Do not allow signon retention for external run-units
- Start Central Versions as started tasks

When signing in to CA IDMS through IDMS Server, there are some cases where a PassTicket behaves like a password, that means the PassTicket can be used multiple times.

Signon When Security Options are Mixed

The security option for some other resources can be different from the security option for signon. For example, signon security might be external while security for other resources is internal.

In the case of mixed security options, the user must be identified to both the external and internal security systems, and a request for signon invokes signon to both security systems. Password checking is performed by either the external system or the internal system, depending on security option for the SGON resource in the SRTT.

If a DDS connection between two CVs exists and one system has internal security while the other has external security, the CV running with internal security must have the userid of its job defined to allow two-phase commit resynchronization processing to complete successfully across the connected CVs.

What is Signon Processing?

Signon Processing Functions

The major function of signon processing is to identify and validate the user requesting CA IDMS services. In addition, signon processing will also cache user-related information such as the list of groups to which a user belongs and profile information.

Explicit Signon

From within a DC/UCF system, signon processing can be initiated explicitly by executing the SIGNON task code or by linking to RHDCSNON from within a user-written application. If CA IDMS/DC is directly controlling terminal access, then an explicit signon must be issued in order to identify the user accessing DC/UCF from an interactive terminal.

Automatic Signon

Signon processing occurs automatically under the following conditions:

- In local mode batch, signon processing occurs within the batch address space when the first security check is issued.
- Within the central version, system signon processing occurs when the first database request is issued from the externally executing application. This applies to applications executing in batch, CMS, TSO, or a front-end teleprocessing monitor such as DC or CICS.
- In UCF applications, signon processing occurs in the UCF back-end when the UCF connection is made from the front-end application.

General Processing Flow

The processing at each step of signon, and whether or not a particular step is actually executed, is based on a number of factors, such as the environment in which signon is occurring and how signon processing is controlled. These factors and their influence on signon processing are discussed later in this chapter.

Signon processing consists of the following steps:

1. Identify the user requesting CA IDMS services.
2. In DC/UCF:
    - If a user is already signed to the terminal, sign the user off.
    - If the user is signing on to an interactive terminal and is already signed on to another interactive terminal, deny the signon request unless multiple signon is allowed.

3. Validate the user and password. An asymmetric uni-directional non-unique hash routine is used to "encrypt" the password associated with a user id. When a user signs on, the password entered is processed using the hash routine and compared to the "encrypted" password value associated with the user id.
4. In DC/UCF, update the user's password if requested (explicit signon requests only).
5. Build the group list for the user.
6. Build the session profile from system and user profile information, subject to specifications on the initial #SECRTT.
7. If signon is the result of linking to RHDCSNON, invoke the CLIST identified by the CLIST attribute, if one exists in the session profile.

Identifying the User

Explicit Signon

When an explicit signon request is issued by executing the SIGNON task or linking to RHDCSNON, the user is identified by the user ID specified on the signon request. The password to be used for verification is also specified as part of the signon request.

Automatic Signon

During automatic signon, the user is identified by the authorization ID under which the application is being executed.

If the signon does not occur within the CA IDMS system, the executing user is extracted from the operating system by issuing the appropriate call to the integration services layer of the CA Common Services architecture.

When signon processing is initiated automatically, no password verification is performed by CA IDMS software. CA IDMS assumes that the user has already been validated by the environment within which the application is executing.

Default Signon

You can allow CA IDMS to perform a signon using a specific name when a security check request is issued and the user is not signed on. This is done by specifying DFLTSGN=YES and the DFLTUID parameter on the initial #SECRTT macro.

Note: For more information, see #SECRTT (see page 204).

User Validation Processing

Dependencies

User validation processing is dependent on the following:

- Whether signon processing is controlled externally, internally or not at all (OFF).
- Whether IDMS resources are controlled externally or internally.
- Whether an explicit or automatic signon is being done.
- Whether the signon is occurring within the DC/UCF system or within a separate address space.

Internally Secured Signon

If signon processing is occurring within a DC/UCF system (whether explicit or automatic), the user must have been granted the SIGNON privilege on the DC/UCF system. If this condition is not satisfied, the user is not signed on and all subsequent security checks will fail.

The user being signed on must also have been defined in the user catalog with a CREATE USER statement, and, in the case of an explicit signon, the password specified must match the password associated with the user definition. If either of these conditions is not satisfied, the user is not signed on and all subsequent security checks will fail.

The user will also be signed on to the external security system if one or more IDMS resources are controlled externally. No password verification takes place for the external signon. If the external signon request fails, the user will not be signed on.

Externally Secured Signon

If signon is controlled externally, the user being signed on must be defined to the external security system. In the case of an explicit signon, the password must match the password associated with the user definition in the external security system or the PassTicket must be validated by the external security system. If the external signon request fails, the user is not signed on and all subsequent security checks will fail.

The user will also be signed on to the internal security system if one or more IDMS resources are controlled internally. No password verification takes place for the internal signon. If the internal signon request fails because the user is not defined to CA IDMS, processing will continue but subsequent security checks for internally controlled resources may fail since the user has no associated groups other than PUBLIC.

No Signon Security

If security for the SGON resource is 'OFF', the user will be signed on to the internal security system without password verification. Regardless of whether the user is defined in the user catalog, processing will continue.

The user will also be signed on to the external security system if one or more IDMS resources are controlled externally. The processing of the external signon request is the same as if signon processing were being controlled internally.

Additional Signon Processing

Updating the Password

In an explicit signon request to CA IDMS/DC, the user can change the password if the user is not already signed on to another terminal. The user can request a change in password during signon processing whether internal or external security is used to control signon processing.

If signon processing is controlled internally, the user's request can be honored if the user catalog (SYSUSER.DDLCSEC area) is available in update mode to the system for which the signon request is issued. Thus, to prevent users from updating their passwords, you can make the user catalog available to users in retrieval mode only.

If signon processing is controlled externally, the user's ability to update the password is subject to any restrictions imposed by the external security system.

Building the User's Group List

As part of internal signon processing, an in-core list of group IDs is built and anchored in the SON control block. The list includes the authorization IDs of all groups of which the user is a member as well as the group PUBLIC.

If signon is secured externally, you can still take advantage of CA IDMS groups to administer security. However, users must be defined in the user catalog in order to be included in a group.

Building the Session Profile

As part of signon processing, the security system will attempt to locate a system profile and a user profile for the user unless directed not to by a USRPROF=OFF or SYSPROF=OFF specification in the initial #SECRTT.

- If no user profile was specified in the user definition, or if there is no user definition in the user catalog (and signon is validated externally), the security system will search the user catalog for, if specified, the user profile designated in the USRPROF= parameter of the initial #SECRTT macro.
- If USRPROF= is not specified, a default user profile definition whose name matches the ID of the signed-on user.

If a user profile is found, the system builds a session profile with the attributes defined in the user profile.

If no **system profile** was specified in the grant of signon privilege to the user, or if there is no grant of signon privilege (and signon is validated externally), the security system will search the system dictionary for the following:

- If specified, the system profile designated in the SYSPROF= parameter of the initial #SECRTT macro.
- If SYSPROF= is not specified, a system profile named 'DEFAULT'.

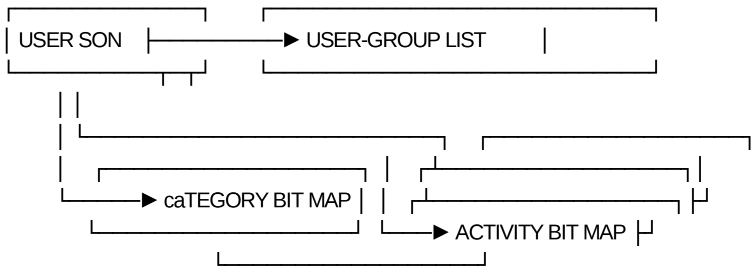
If a system profile is found, the attributes specified in the system profile are merged into the session profile. If user and system profile attributes match, the attribute value in the system profile takes precedence.

Note: For more information about how to tailor user and system profiles when signon is secured externally, see Securing User Profiles (see page 90).

Signon Control Block

Pointers to Security Data

When the user is successfully signed on, a signon control block (SON) is constructed. User authority data is brought into memory and linked to the SON, as shown in the following illustration:



When Data is Linked to the SON

The following table shows when the various security data are brought into memory and linked to the SON.

Security data	Brought into memory
User-group list	During signon
category bit map	At the first security check which requires categories
Activity bit map	When the application issues its first activity security check

Retaining Signon Information

You can specify that CA IDMS should retain signon information originating from external request units (ERUs). In some situations this provides a performance benefit.

To retain ERU signon information, specify SGNRETN=*time-interval* on the initial #SECRTT macro.

Note: For more information, see #SECRTT (see page 204).

Chapter 6: Using CA IDMS Internal Security

This section contains the following topics:

CA IDMS Resources

CA IDMS resource types are grouped as follows:

- Global resources
- System resources
- Database resources

Global Resources

What is a Global Resource?

A global resource is an entity which is shared by all CA IDMS processing in the security domain.

The following table shows the global resource types and the corresponding resource type keywords used in the SRTT and security information databases:

Resource type	Keyword
User	USER
Group	GROU
User profile	UPRF

User Catalog

The definition of a global resource is stored in the user catalog. The user catalog is an area (SYSUSER.DDLSEC) that is shared for retrieval by all CA IDMS processing in the security domain.

Users

The user resource type represents the end users, programmers, and administrators who will be accessing systems and databases in the CA IDMS security domain. Users are identified by a user ID that must be unique across the domain.

You maintain the definitions of users in CA IDMS with the CREATE/ALTER/DROP USER statements.

Groups

The group resource type represents a collection of users. The following are some important concepts related to groups:

- All users in a group implicitly hold all privileges granted to the group.
- You can assign a user to any number of groups.
- Every user belongs to the group PUBLIC.
- A group cannot be assigned to another group.

You maintain groups with the CREATE/ALTER/DROP GROUP statements.

User Profiles

A user profile is a set of attributes that apply to a given user for both online and batch execution in any system in the domain. You create a user profile with the CREATE USER PROFILE statement.

An attribute specifies an environmental default for a user session. An attribute is expressed as a keyword and an associated value for the keyword. For example, SCHEMA=MISTEST is an attribute, of which the keyword is SCHEMA.

Even though the user profile is defined in the CA IDMS user catalog, it is possible for user profile attributes to be invoked whether signon is secured internally or externally.

Note: For more information about user profiles, see Securing User Profiles (see page 90).

System Profiles

You can also create a system profile to associate with one or more users in granting signon privilege to a given system. The attributes in a system profile apply to a user session on a specified system. If both a user profile and a system profile are found when the user signs on, the attributes in the two profiles are merged into a user session profile. System profile attributes take precedence over user profile attributes for those attributes defined with an OVERRIDE parameter equal to YES.

Note: For more information about creating system profiles, see the *CA IDMS System Tasks and Operator Commands Guide*.

System Resources

What is a System Resource

A system resource is an entity shared by all CA IDMS processing under the central version.

The following table shows the system resource types and the corresponding resource type keywords used in the SRTT and security information databases:

Resource type	Keyword
Activity	ACTI
Application	SAPP
Category	CATE
Signon	SGON
System	SYST
System profile	SPRF

System Dictionary

The system dictionary includes all information required to establish, maintain, and control the processing environment. System resources are defined in the DDLDML area of the system dictionary. A system resource is available to all systems generated from the system dictionary.

Purpose of Categories

The category is a mechanism that allows you to group occurrences of several resource types that you have secured internally so that you can grant privilege on the group of resources.

When you create a category, you assign it a name, allowing you to associate a meaningful identifier with the resources. For example, if you secure tasks internally, you might create a category 'SYS_TASKS' and add the DCMT and DCUF tasks to it. If you secure both tasks and programs, one category could contain both task and program resources.

You can define as many as 32,768 categories for your security scheme.

The following table shows resource types that can be categorized and the corresponding resource type keywords used in the SRTT and security information databases:

Resource type	Keyword
Task	TASK
Program	SPGM
Load module	SLOD
Access module (loadable entity)	SACC
Run unit	NRU
Queue	QUEU

Important! If you secure the DB resource, you secure run units and access modules system-wide. You must then categorize load modules in order to grant users execution privilege on them, and you must do the same with access modules unless you choose to grant execution privilege on individual access modules rather than grouping them first.

For more information, see Securing Database Resources (see page 117).

Defining a Category

You add resources to a category with a CREATE or ALTER CATEGORY statement, as in this example:

```
create category dcmt
  add program cdmslib.rhdcmt*;
```

Granting Privilege on the Category

After you define the category, the only means of access to a resource in the category is execution privilege on the category. You give this privilege to a user with a GRANT statement, as illustrated in this example:

```
grant execute
    on category dcmt
    to sam;
```

Runtime Category Selection

At runtime a given resource name may appear to qualify for assignment in more than one category. Consider these two categories:

```
create category dcmt
  add program cdmslib.rhdcmt*;

create category dcmtab
  add program cdmslib.rhdcmtab;
```

When the security system processes as security check, it determines the category of the resource being checked by selecting the mask that is closest to the fully qualified name of the resource. For example, given the preceding two categories, the security system will determine that:

- Use of resource CDMSLIB.RHDCMTXY requires execution privilege on category DCMT.
- Use of resource CDMSLIB.RHDCMTAB requires execution privilege on category DCMTAB.

Database Resources

What is a Database Resource

A database resource is an entity associated with the definition of or access to a database.

Database Resource Types

The following table shows the database resources type and the corresponding resource type keywords used in the SRTT and security information databases:

Resource type	Keyword
Database ■ Area ■ Run unit ■ SQL schema ■ Non-SQL defined schema ■ Table ■ Access module	DB ■ AREA ■ NRU ■ QSCH ■ NSCH ■ TABLE ■ DACC
DBTABLE	DBTB
DMCL	DMCL

Securing Database Resources

If you specify internal security for the database (DB) resource type, you automatically secure the other resource types listed with DB in the preceding table.

You can grant privileges on the individual resource types, but you cannot turn security off in the SRTT for the resource types that are grouped with DB when DB is secured.

Database Occurrence Overrides

Using an occurrence override in the SRTT, you can specify a security option for an individual database associated with the system dictionary. For example, in one SRTT entry you can specify no security (the default) for resource type DB and in another entry specify internal security for the production database.

Ownership

Ownership is an attribute of an SQL schema. A user who issues a CREATE SCHEMA statement owns the schema that is created.

A schema owner implicitly holds all access and definition privileges on the tables, functions, procedures, table procedures, views, and access modules associated with the schema. The owner also has the authority to grant those privileges to others.

An owner cannot *grant* ownership to another user but can *transfer* ownership. In this way, ownership and its privileges are relinquished to the other user.

The DBMS does not check for ownership. It requests a check for a specific privilege such as SELECT privilege on a table, and the security system returns a positive response if the user in question is the owner of the object.

CA IDMS Privileges

How Privileges Work

If you specify the internal security option for a resource, the resource is secured against access by users who have not been granted privilege on the resource. This security applies to the resource in all systems within the CA IDMS security domain that share the SRTT.

Whoever has authority to grant a privilege has the authority to revoke it. All authority to grant and revoke CA IDMS privileges derives from users who hold SYSADMIN privilege.

Types of Privilege

There are three types of CA IDMS privileges:

Administration privileges	Definition privileges	Access privileges
SYSADMIN	CREATE	SIGNON
DCADMIN	ALTER	EXECUTE
DBADMIN	DROP	SELECT
	DISPLAY	INSERT
	USE	UPDATE
	REFERENCES	DELETE
		DBAREAD
		DBAWRITE

Administration Privileges

Administration privileges allow a user to grant and revoke security privileges within a particular scope:

- **SYSADMIN** in effect allows a user to grant all privileges and should be restricted to the security administrator.
- **DCADMIN** allows a user to grant privileges for one or more CA IDMS systems.
- **DBADMIN** allows a user to grant privileges for a database.

Definition Privileges

Definition privileges allow a user to manipulate the definition of certain resources. You can grant definition privileges singly or as a group (the DEFINE privilege).

The following are the individual definition privileges:

- Create
- Alter
- Drop
- Display
- Use
- References (tables)

Access Privileges

Access privileges give users the authority to access specified resources at runtime.

There are three categories of access privileges:

- **Execution** privilege allows a user to execute an access module, activity, or category.
- **Table access** privileges allow a user to perform these operations on data contained in a table:
  - Select
  - Insert
  - Update
  - Delete

- **Special access** privileges refer to the authority to signon to a system or to execute utility functions against an area of the database:
  - SIGNON allows a user to sign on to a specified CA IDMS system.
  - DBAREAD allows a user to run read-only utilities against an area.
  - DBAWRITE allows a user to run utilities that perform read-write functions against an area.

Granting and Revoking Privileges

Absolute Authority of SYSADMIN

Once you have defined the authorization ID of the security administrator with SYSADMIN privilege and you have secured administration privileges, the privilege to create additional users in the CA IDMS security domain, and the privilege to grant those users privileges, must derive from the security administrator.

Applicability of Privileges

CA IDMS privileges are applicable to resources for which the security option is 'INTERNAL'. If the security option for a resource is 'OFF', a user can access the resource without holding a privilege. If the security option for a resource is 'EXTERNAL', the user's authority to access the resource is determined by the external security system.

Therefore, to use the system of CA IDMS privileges, you must ensure that the runtime security option for the resources to which privileges apply is 'INTERNAL'.

Granting privilege

You grant privileges with a GRANT statement. Implicit in each administration privilege is the authority to grant certain privileges:

- SYSADMIN can grant privileges on SYSADMIN, DCADMIN, DBADMIN, and on global resources.
- DCADMIN can grant privileges on DCADMIN and on system resources.
- DBADMIN can grant privileges on DBADMIN and on database resources.

A GRANT statement includes an ON parameter which specifies the resource to which the privileges apply and a TO parameter which specifies the users or groups to whom you are giving the privileges.

Note: For more information about GRANT statement syntax, see the following sections:

- Syntax for Securing Global Resources (see page 231)
- Syntax for Securing System Resources (see page 261)
- Syntax for Securing Database Resources (see page 295)

Duration of Privileges

A user holds privileges explicitly granted to the user until one of these actions occurs:

- The privileges are explicitly taken away by means of the REVOKE statement.
- The user is physically deleted from the user catalog.

A user implicitly holds privileges granted to a group to which the user belongs until one of these actions occurs:

- The user is dropped from the group.
- The privileges are revoked from the group.
- The group is dropped.

Revoking Privileges

Privileges are taken away with the REVOKE statement. A user who has the authority to grant a privilege also has the authority to revoke it.

A REVOKE statement includes an ON parameter which specifies the resource to which the privileges apply and a FROM parameter which specifies the users or groups from whom you are revoking the privileges.

Note: For more information about REVOKE statement syntax, see the following sections:

- Syntax for Securing Global Resources (see page 231)
- Syntax for Securing System Resources (see page 261)
- Syntax for Securing Database Resources (see page 295)

GRANT and REVOKE Example

The first statement gives a table access privilege to user PSD, and the second statement revokes the privilege:

```
grant select
  on table demoempl.employee
  to psd;

revoke select
  on table demoempl.employee
  from psd;
```

Granting WITH GRANT OPTION

Grantable Privilege

When you grant a definition or access privilege to a user, you can also give the user the authority to grant the same privilege to another user; in effect, to pass on the privilege. This authority is called the grantable privilege.

To give a grantable privilege to a user, you specify WITH GRANT OPTION at the end of the GRANT statement.

Giving grantable privileges is an essential technique in decentralizing security administration.

Grantable Privilege Example

In this example, the GRANT statement gives user PSD SELECT privilege on the demoempl.employee table, as well as the authority to assign that privilege to other users:

```
grant select
  on table demoempl.employee
  to psd
  with grant option;
```

User PSD can now use the GRANT statement to issue the SELECT privilege on the demoempl.employee table to other users.

Restrictions on Grantable Privilege

Not all privileges can be grantable privileges. These privileges *cannot* be grantable:

- All administration privileges
- Signon privilege
- Execution privileges on activities and categories

A user holding a grantable privilege does not necessarily have the authority to grant the privilege WITH GRANT OPTION.

Note: For more information about restrictions on passing grantable privilege, see the discussion of the WITH GRANT OPTION parameter under the applicable GRANT statements in the following chapters:

- Syntax for Securing Global Resources (see page 231)
- Syntax for Securing System Resources (see page 261)
- Syntax for Securing Database Resources (see page 295)

Omitting WITH GRANT OPTION

If you omit WITH GRANT OPTION when you grant a definition or access privilege, the named users receive the definition or access privilege, but it is not grantable. Therefore, the users cannot give the privilege to other users.

Grantable Privilege with REVOKE Statements

Unless you hold an administration privilege, you can revoke a privilege only if you hold the same grantable privilege. For example, a user cannot revoke CREATE privilege on SYSTEM88 unless the user holds grantable CREATE privilege on SYSTEM88.

Specifying Groups

Granting Privileges to a Group

You can grant privileges to a group as well as to individual users. All users in the group hold privileges that you give to the group. Users you add to a group hold all privileges assigned to the group; users you remove from the group lose all group privileges.

Revoking Group Privileges

You cannot revoke a privilege from an individual user if the user belongs to a group that holds the privilege. Rather, you must take one of these steps:

- Drop the user from the group using the ALTER GROUP statement.

  This action removes *all privileges* the user held as a result of being in the group.

- Remove the privilege from the group.

  This action removes the privilege from *all members* of the group who hold the privilege as a result of being in the group.

Using a Wildcard

What Wildcarding Is

Wildcarding is the use of a single character to represent one or more characters omitted from a string. An entity name with a wildcard character identifies all the entities whose names match the pattern established by the wildcarded name.

Why You Use Wildcards

In most cases, you can use a wildcard when naming the resources to which the privileges in a GRANT statement apply. This allows you to do the following:

- Enforce high-level naming conventions
- Grant privileges on groups of resources

Document convention: If a parameter value in a security statement can include a wildcard, the parameter description that follows the statement syntax diagram explicitly notes your ability to use a wildcard.

How to Wildcard: The wildcard character is the asterisk (*). You can use the wildcard only as the last character in a resource name. For example, * and A* and ABCD* are valid, but *A and A*BC are not.

Wildcarding Qualified Resource Names

In some cases, wildcarding is permitted only on the last one or two identifiers in a qualified resource name. For example, when you grant or revoke area access privileges, you identify the area as *segment-name*.*area-name*; the qualifier *segment-name* is required, but wildcarding is not permitted on *area-name*. In such a case, these are examples valid and invalid resource names:

Valid area names

APPLDICT.HR*

APPLDICT.*

Valid area names

APPLDICT*

APPL*

Specific restrictions on wildcarding are described appropriately in the syntax parameter descriptions found in the statement syntax chapters later in this manual.

Note: Special considerations apply to the effect of using a wildcard in reference to categories generally and in CREATE RESOURCE statement particularly. For more information, see the Usage (see page 272) section of CREATE RESOURCE in the chapter Syntax for Securing System Resources.

Granting and Revoking with a Wildcard

When you grant a privilege on resources using a wildcard, you must use the same wildcard to revoke the privileges.

For example, if you grant CREATE privilege on category HR* to user ABC, you must issue a REVOKE CREATE or REVOKE DEFINE statement on category HR* to revoke the privilege from user ABC. Revoking privilege on category *, category H*, category HR, or category HRA has no effect on privileges granted on category HR*.

Considerations in Revoking Privileges

Through the use of groups and wildcards in a GRANT statement, a user can be given the same privilege on a resource more than once. A REVOKE statement revokes the privileges specified in the statement only on the specified resource name and only from the specified user or group. Thus, it is possible for a user to retain a privilege even after it has been revoked.

For example, suppose:

- User PKB is in the group SALES_ADMIN.
- PKB has been granted the CREATE privilege on the access module name SALES_SCH.SALES_FORECAST.
- SALES_ADMIN has been granted the CREATE privilege on all access modules named SALES_SCH.SALES* where * is a wildcard character.

You can revoke the CREATE privilege on SALES_FOREcaST from the user identifier PKB. However, PKB can still create an access module by that name in the SALES_SCH schema because PKB is a member of SALES_ADMIN.

Efficiency Considerations

In security administration, you can perform your task more efficiently by making consistent use of groups, wildcards, and categories. This strategy will also produce runtime efficiency.

Using Groups

Your security strategy should isolate user roles that require similar types of privileges. You can then establish groups for each user role. This allows you to grant and revoke privileges at the group level, thus reducing the number of statements needed to administer the security scheme.

Note: Groups also enhance efficiency by improving runtime performance.

Comparison of Groups and No Groups

- **Without groups**

  Without groups, you must list each user ID for each GRANT statement:

  ```
  grant access on table qa.employee to psd, rkn, jfd, wxe, lsb;
  grant access on table qa.job to psd, rkn, jfd, wxe, lsb;
  grant access on table qa.benefits to psd, rkn, jfd, wxe, lsb;
  grant access on table qa.department to psd, rkn, jfd, wxe, lsb;

  grant execute on access module qa.empdbmod to psd, rkn, jfd, wxe, lsb;
  grant execute on access module qa.empdbret to psd, rkn, jfd, wxe, lsb;
  ```

  Now to revoke privileges for one of the users, you must code a REVOKE statement for each GRANT statement:

  ```
  revoke access on table qa.employee from rkn;
  revoke access on table qa.job from rkn;
  revoke access on table qa.benefits from rkn;
  revoke access on table qa.department from rkn;

  revoke execute on access module qa.empdbmod from rkn;
  revoke execute on access module qa.empdbret from rkn;
  ```

- **With groups**

  After you create the group, you list only the group name on each GRANT statement:

  ```
  create group qagroup
     add user psd, rkn, jfd, wxe, lsb;

  grant access on table qa.employee to qagroup;
  grant access on table qa.job to qagroup;
  grant access on table qa.benefits to qagroup;
  grant access on table qa.department to qagroup;

  grant execute on access module qa.empdbmod to qagroup;
  grant execute on access module qa.empdbret to qagroup;
  ```

  To revoke privileges from one of the users, you simply drop the user from the group:

  ```
  alter group qagroup
    drop user rkn;
  ```

Using Wildcards

Your strategy should isolate resources that require similar types of security. You can then grant privileges on them using a wildcard. This allows you to implement your strategy at a higher level, thus eliminating the need to issue a GRANT statement for individual resources.

Wildcard Examples

Without wildcards, you must issue a separate statement for each table when you grant table access privileges:

- **Without wildcards**

  ```
  grant access on table qa.employee to qagroup;
  grant access on table qa.job to qagroup;
  grant access on table qa.benefits to qagroup;
  grant access on table qa.department to qagroup;

  grant execute on access module qa.empdbmod to qagroup;
  grant execute on access module qa.empdbret to qagroup;
  ```

- **With wildcards**

  You can use a wildcard to grant table access privileges on all tables in the qa schema:

  ```
  grant access on table qa.* to qagroup;

  grant execute on access module qa.* to qagroup;
  ```

Using categories

CA IDMS provides the category mechanism to help you to manage privileges on runtime resources efficiently.

Summary Example

This series of statements uses groups, wildcards, and categories to secure the resources available for two levels of use, as described in the definition of groups **hrdisp** and **hrupd**:

```
create group hrdisp
  description 'HR users who can display Employee'
  add user lsd, lhn, pxw, gsr, hxm, fbs;

create group hrupd
  description 'HR users who can update Employee'
  add user gsr, hxm, fbs;

create resource category benefits_display
  add access module appldict.prod.bendis
  add load module   appldict.v0001.benefits
  add program       cdmslib.bendisp
  add task          bendisp;
```

```
create resource category benefits_update
   add access module appldict.prod.benupd
   add program       cdmslib.benupd
   add task          benupd;

grant execute on category benefits_display to hrdisp;
grant execute on category benefits_* to hrupd;
```

Chapter 7: Securing Global Resources

This section contains the following topics:

CA IDMS Security Domain

What It Is

The CA IDMS security domain is the set of DC systems and local mode applications sharing a single user catalog and SRTT.

Global Resources

The scope of global resources is domain-wide. These resources are:

- Users
- Groups
- User profiles
- SYSADMIN privilege

User catalog

The user catalog is the CA IDMS repository that contains:

- The definition of all authorization IDs (users and groups) within the domain
- The specification of authorization IDs holding SYSADMIN privilege for the domain
- The definition of user profiles
- The privileges held by users and groups on global resources

The User Catalog

Defining a CA IDMS Security Domain

You include multiple CA IDMS systems in a security domain by specifying an identical set of physical characteristics for the SYSUSER.DDLSEC segment in each system in the domain, and specifying the same physical data set in the startup JCL or the global DMCL.

Note: For more information about defining physical database characteristics, see the *CA IDMS Database Administration Guide*.

Use of the User Catalog

The user catalog is accessed by all DC/UCF systems and local mode batch applications executing in the security domain. It is a central location used for validating passwords and retrieving user information.

Only a DC/UCF system that has the user catalog in update mode can be used to define and administer global resources.

Securing the User Catalog

After you have specified a resource option (other than 'OFF') for the DB resource type, the user catalog is secured.

You can grant access to the user catalog in one or more of these ways:

- Granting DBAREAD/DBAWRITE privileges on area SYSUSER.DDLSEC
- Granting DBADMIN privilege on DB SYSUSER
- Granting USE privilege on non-SQL defined schema IDMSSECU and granting definition privilege on an SQL schema for IDMSSECU

Note: For more information, see Securing the Dictionaries and the User catalog (see page 149).

Ensuring Use of the Correct User Catalog

You can ensure that only the correct user catalog is accessed at runtime.

If the operating system or spooler supports installation-written exits for scanning and validating JCL, a system programmer can write an exit to verify that the correct system dictionary and user catalog are used by each central version and local mode job.

Alternatively, in an operating system that supports dynamic file allocation, you can specify the data set name of the user catalog in the DSNAME parameter of the CREATE FILE statement and NULL for the external file name in the ASSIGN TO parameter. At runtime the data set name is obtained from the DMCL, which contains the segment associated with the file.

Securing SYSADMIN Privilege

About SYSADMIN Privilege

SYSADMIN privilege authorizes the holder to grant and revoke privileges on any resource within the domain. It also enables the holder to define resources and to delegate administration privileges.

In sum, the holder of SYSADMIN privilege can administer the security system.

Until you secure the SYSADMIN resource, any user can administer SYSADMIN privilege.

How to Secure SYSADMIN

To secure SYSADMIN internally, include an entry in the SRTT:

```
#SECRTT   TYPE=ENTRY,                        X
   RESTYPE=SYSA,                         X
   SECBY=INTERNAL
```

To secure SYSADMIN externally, include an entry in the SRTT:

```
#SECRTT   TYPE=ENTRY,                        X
   RESTYPE=SYSA,                         X
   SECBY=EXTERNAL,                         X
   Additional parameters required
```

Note: For more information, see #SECRTT (see page 204).

Restricting SYSADMIN

Since SYSADMIN is the master security definition privilege, it is very important to restrict the granting of SYSADMIN authority.

Consider assigning SYSADMIN to a group rather than an individual user so that security can be administered in a timely fashion should the primary administrator be unavailable.

Decentralizing Administration

The holder of SYSADMIN can decentralize security administration by granting to appropriate users:

- DCADMIN privilege, which allows a user to define system resources and grant access to those resources
- DBADMIN privilege on a database, which allows a user to define database resources and grant access to those resources
- Definition privilege on global resources

You should carefully restrict grants of administration privileges. A user with administrative privilege can grant and revoke privileges on all resources within the scope of the administration privilege.

Granting Administration Privileges

You can give SYSADMIN, DCADMIN, and DBADMIN privileges to one or more users with a grant statement, as in this example of a statement that grants DBADMIN privilege on a specified database:

```
grant dbadmin
  on db testdb
  to devdba;
```

More Information

For more information about granting administration privileges, see the following sections:

- GRANT Administration Privilege (see page 250) in the chapter "Syntax for Securing Global Resources"
- GRANT Administration Privilege (see page 298) in the chapter "Syntax for Securing System Resources"
- GRANT Administration Privilege (see page 250) in the chapter "Syntax for Securing Database Resources"

Securing Users

About Users

Defining users in the CA IDMS user catalog is essential if the security option for one or more resources is internal, even if signon processing is controlled externally. A security check on an internally secured resource fails if the executing user is not defined in the user catalog.

Until you secure the user resource, any user can define users in the user catalog.

How to Secure Users

To secure users internally, include an entry in the SRTT:

```
#SECRTT   TYPE=ENTRY,
    RESTYPE=USER,                        X
    SECBY=INTERNAL                        X
```

To secure users externally, include an entry in the SRTT:

```
#SECRTT   TYPE=ENTRY,
    RESTYPE=USER,                        X
    SECBY=EXTERNAL,                        X
    Additional parameters required
```

Note: For more information, see #SECRTT (see page 204).

How to Define Users

You define a user with a CREATE USER statement. For example, this statement creates user RKN:

```
create user rkn
  group mis
  name 'Randall K. Nelken'
  password ranken
  profile misprof;
```

Note: For more information, see CREATE USER (see page 239).

Maintaining User Definitions

You can alter the definition of a user with an ALTER USER statement. You can drop the definition of a user with a DROP USER statement.

Note: For more information, see the following sections:

- ALTER USER (see page 233)
- DROP USER (see page 247)

Granting Definition Privileges on Users

You can delegate the authority to define and maintain users by granting definition privileges on users. You can specify any combination of CREATE, ALTER, DROP, and DISPLAY privileges, or you can specify all definition privileges (DEFINE). You can specify WITH GRANT OPTION when you grant these privileges to allow the user to grant the same privileges to another user.

In this example, user mis1 is given the privilege to create or alter the definition of users whose user IDs begin with 'mis':

```
grant alter, create
  on user mis*
  to mis1;
```

Note: For more information, see GRANT Definition Privileges (see page 252).

Securing Groups

About Groups

You define groups for administrative efficiency. You group users according to the privileges that they require. Then you grant the privileges to the group rather than to individual users.

If you create a group of 10 users, you can grant each user the same five privileges by issuing five GRANT statements to the group. You would issue 50 statements to accomplish the same task if you did not first create the group.

Until you secure the group resource, any user can maintain definitions of groups in the user catalog.

How to Secure Groups

To secure the groups internally, include an entry in the SRTT:

```
#SECRTT   TYPE=ENTRY,
    RESTYPE=GROU,                         X
    SECBY=INTERNAL                        X
```

To secure the groups externally, include an entry in the SRTT:

```
#SECRTT   TYPE=ENTRY,
    RESTYPE=GROU,                         X
    SECBY=EXTERNAL,                        X
    Additional parameters required
```

Note: For more information, see #SECRTT (see page 204).

How to Define Groups

You define a group by specifying the users that comprise the group with the CREATE GROUP statement.

Note: A group cannot be a member of another group.

For example, this statement creates mis_group:

```
create group mis_group
  description 'Management Information Services'
  add user mis1, mis2, mis3, mis4, mis5;
```

Note: For more information about defining and maintaining group definitions, see the following sections:

- CREATE GROUP (see page 238)
- ALTER GROUP (see page 231)
- DROP GROUP (see page 245)

Granting Privileges to a Group

When you grant privileges to a group, each member of the group is implicitly granted the specified privileges.

For example, this statement grants mis_group the privilege of retrieving data from SYSTEM tables:

```
grant select
  on table system.*
  to mis_group;
```

A member of a group can hold additional privileges as an individual user or as a member of a different group. For example, mis1 might hold an administrative privilege that other members of mis_group do not hold.

Dropping Users and Groups

When a user is dropped from a group, all privileges inherited from the group are implicitly revoked from the user.

When a group is dropped, all privileges granted to that group are automatically revoked.

Granting Definition Privileges on Groups

You can delegate the authority to define and maintain groups by granting definition privileges on groups. You can specify any combination of CREATE, ALTER, DROP, and DISPLAY privileges, or you can specify all definition privileges (DEFINE). You can specify WITH GRANT OPTION when you grant these privileges to allow the user to grant the same privileges to another user.

Note: For more information, see GRANT Definition Privileges (see page 252).

Securing User Profiles

About User Profiles

You define a user profile to specify attributes for a user session in the domain (that is, irrespective of the system to which the user is signed on) whether the execution mode is online or batch.

Until you secure the user profile resource, any user can maintain user profile definitions in the user catalog.

How to Secure User Profiles

To secure user profiles internally, include an entry in the SRTT:

```
#SECRTT   TYPE=ENTRY,
    RESTYPE=UPRF,                    X
    SECBY=INTERNAL                    X
```

To secure the user profiles externally, include an entry in the SRTT:

```
#SECRTT   TYPE=ENTRY,
    RESTYPE=UPRF,                    X
    SECBY=EXTERNAL,                    X
    Additional parameters required
```

Note: For more information, see #SECRTT (see page 204).

Attributes

An attribute is the combination of a keyword and a value associated with the keyword. A user profile can contain multiple attributes.

Attributes are used by CA IDMS software to control the user session. Attributes can also be retrieved by application programs for additional application security and other purposes.

When you specify an attribute in a user profile definition, you have the option of marking it to indicate that the user is not permitted to override the attribute value at runtime with a DCUF SET PROFILE statement or, for attribute keywords with meaning to CA IDMS, with a SYSIDMS parameter.

Note: For more information about DCUF SET PROFILE, see the *CA IDMS System Tasks and Operator Commands Guide*.

For more information about SYSIDMS parameters, see the *CA IDMS Common Facilities Guide*.

How to Define User Profiles

You define a user profile with the CREATE USER PROFILE statement.

In this example, the first statement creates a user profile called MISPROF and the second statement associates the profile with user RKN:

```
create user profile misprof
  attributes
    dept='0056' override no,
    jobcode='42' override no,
    schema='&user'.,
    prtdest='gdnc005';

alter user rkn
  profile misprof;
```

Note: For more information about creating and maintaining user profile definitions, see the following sections:

- CREATE USER PROFILE (see page 242)
- ALTER USER PROFILE (see page 235)
- DROP USER PROFILE (see page 249)

Granting Definition Privileges on User Profiles

You can delegate the authority to define and maintain user profiles by granting definition privileges on user profiles. You can specify any combination of CREATE, ALTER, DROP, and DISPLAY privileges, or you can specify all definition privileges (DEFINE). You can specify WITH GRANT OPTION when you grant these privileges to allow the user to grant the same privileges to another user.

Note: For more information, see GRANT Definition Privileges (see page 252).

Associating User Profiles with Users

You can associate a user profile with a user in one of the following ways:

- **Explicitly** in the PROFILE parameter of a CREATE USER or ALTER USER statement.

  Note: For more information, see the following sections:

  - CREATE USER (see page 239)
  - ALTER USER (see page 233)
- **Implicitly** by assigning the user profile a name matching the user ID.

  This user profile is located at signon if no user profile has been specified in the user definition.

User Attributes in a System Profile

A system profile allows you to set the attributes of a user's session for a specific system. The system profile associated with the user is determined in one of the following ways:

- By the specification, if any, made in the GRANT SIGNON statement for the user.
- By the specification of default system profile made on the initial #SECRTT macro.
- If not specified on the #SECRTT macro, the system profile DEFAULT, if it exists.

Even if there is no system profile specification in GRANT SIGNON or on the #SECRTT, you can tailor a system profile to a user or the user's default group by specifying INCLUDE='&USER'. or INCLUDE='&GROUP'. in a system profile named 'DEFAULT.' because the system will search for the system profile DEFAULT at signon time.

If you have created a system profile with a name that matches the ID of the signed-on *user* and system profile DEFAULT contains INCLUDE='&USER'., the attributes of the nested system profile with a name matching &USER. are set for the session profile.

If you have created a system profile with a name that matches the name of the signed-on user's default *group* and system profile DEFAULT contains INCLUDE='&GROUP'., the attributes of the nested system profile with a name matching &GROUP. are set for session profile.

Scope of Profiles

The scope of system profile DEFAULT is the set of systems that share the SYSTEM.DDLDML area in which DEFAULT is defined. The scope of a user profile is the CA IDMS domain, which is the set of systems that share the SYSUSER.DDLSEC area. System profile attributes take precedence over matching user profile attributes unless the user profile attribute is defined with the OVERRIDE NO parameter.

Note: For more information about system profiles, see the *CA IDMS System Tasks and Operator Commands Guide*.

Chapter 8: Securing System Resources

This section contains the following topics:

Securing Definitions In The System Dictionary

Why You Do It

Security information about system resources (and some database resources) is stored in the system dictionary. To secure the security information itself, secure the system dictionary.

How You Do It

Since the system dictionary is a CA IDMS database, you secure the system dictionary by activating security for resource type DB. You can specify an SRTT entry to secure all databases in the system, or you can specify an individual entry for each database you choose to secure.

Important! Before securing any database in the system, become familiar with all considerations related to database security, as described in Securing Database Resources (see page 117).

Securing DCADMIN

About the DCADMIN Privilege

A holder of DCADMIN privilege can perform system administration functions. The DCADMIN user holds definition and access privileges on /DC system resources. CREATE, ALTER, DROP, and DISPLAY privileges on a system allow the user to maintain the system configuration using CA IDMS system generation.

The holder of DCADMIN privilege can grant all system privileges to one or more users.

Until you secure the DCADMIN resource, any user can administer security on system resources.

How to Secure DCADMIN

To secure DCADMIN internally, include an entry in the SRTT:

```
#SECRTT   TYPE=ENTRY,                        X
     RESTYPE=DCA,                         X
     SECBY=INTERNAL
```

To secure DCADMIN externally, include an entry in the SRTT:

```
#SECRTT   TYPE=ENTRY,                        X
     RESTYPE=DCA,                         X
     SECBY=EXTERNAL,                       X
     Additional parameters required
```

Note: For more information, see #SECRTT (see page 204).

How to Grant the DCADMIN Privilege

You can grant DCADMIN privilege to one or more users with a GRANT DCADMIN statement. To issue this statement, you must hold either SYSADMIN privilege or DCADMIN privilege.

Note: For more information, see GRANT Administration Privilege (see page 275) in the chapter Syntax for Securing System Resources.

Securing Systems

About Systems

The system resource represents DC systems in the domain.

Until you secure systems, any user can create and maintain a system using CA IDMS system generation.

How to Secure Systems

To secure systems internally, include an entry in the SRTT:

```
#SECRTT   TYPE=ENTRY,                     X
    RESTYPE=SYST,                     X
    SECBY=INTERNAL
```

To secure systems externally, include an entry in the SRTT:

```
#SECRTT   TYPE=ENTRY,                     X
    RESTYPE=SYST,                     X
    SECBY=EXTERNAL,                     X
    Additional parameters required
```

Note: For more information, see #SECRTT (see page 204).

How to Define a System

You define a system as a secured resource with a CREATE RESOURCE SYSTEM statement, specifying the identifier of the system to match the ID parameter of the SYSTEM statement in system generation.

Note: For more information about defining and maintaining system resources, see the following sections in the chapter Syntax for Securing System Resources:

- CREATE RESOURCE (see page 267)
- DROP RESOURCE (see page 273)

How to Grant Definition Privileges on Systems

You can delegate the authority to define and maintain systems by granting definition privileges on systems. You can specify any combination of CREATE, ALTER, DROP, and DISPLAY privileges, or you can specify all definition privileges (DEFINE). As a holder of DCADMIN privilege, you can specify WITH GRANT OPTION when you grant definition privileges to allow the recipient to grant the same privileges to another user.

Note: For more information about administering privileges on systems, see the following sections in the chapter Syntax for Securing System Resources:

- GRANT System Definition Privileges (see page 282)
- REVOKE System Definition Privileges (see page 291)

Securing Signon

About Signon

The signon resource controls access to DC systems, whether the signon is explicit or implicit. The security option you specify for signon determines whether password validation is enforced by the external system, internally, or not at all.

Note: For more information about signon processing, see the chapter Signon Processing (see page 57).

Until you secure the signon resource, any user can sign on to any system in the domain.

How to Secure Signon

To secure signon internally, include an entry in the SRTT:

```
#SECRTT  TYPE=ENTRY,                X
  RESTYPE=SGON,                   X
  SECBY=INTERNAL
```

To secure signon externally, include an entry in the SRTT:

```
#SECRTT  TYPE=ENTRY,                X
  RESTYPE=SGON,                   X
  SECBY=EXTERNAL,                  X
  Additional parameters required
```

Note: For more information, see #SECRTT (see page 204).

Granting System Signon Privilege

To control access to a system, you grant signon privilege to a user. You cannot grant signon to a group.

Note: For more information, see the following sections in the chapter Syntax for Securing System Resources:

- CREATE RESOURCE (see page 267)
- DROP RESOURCE (see page 273)

When you grant signon, you can associate a system profile with the user.

Note: For more information about system profiles, see the *CA IDMS System Tasks and Operator Commands Guide*.

Securing System Profiles

About System Profiles

The system profile resource represents a profile that is defined for a given system and can be associated with one or more users.

Although system profile is a resource that you can protect with CA IDMS centralized security, system profiles are not considered part of the security architecture because the scope of their influence on user session attributes is system-wide, not domain-wide.

Until you secure system profiles, any user can create and maintain a system profile.

How to Secure System Profiles

To secure system profiles internally, include an entry in the SRTT:

```
#SECRTT   TYPE=ENTRY,                          X
   RESTYPE=SPRF,                             X
   SECBY=INTERNAL
```

To secure system profiles externally, include an entry in the SRTT:

```
#SECRTT   TYPE=ENTRY,                          X
   RESTYPE=SPRF,                             X
   SECBY=EXTERNAL,                             X
   Additional parameters required
```

Note: For more information, see #SECRTT (see page 204).

Defining a System Profile

You define a system profile with a CREATE SYSTEM PROFILE statement, specifying the profile name and profile attributes.

Note: For more information about defining and maintaining system profiles, see the *CA IDMS System Tasks and Operator Commands Guide*.

Granting Definition Privileges on System Profiles

You grant definition privileges on systems profiles with a GRANT statement. You can specify any combination of CREATE, ALTER, DROP, and DISPLAY privileges, or you can specify all definition privileges (DEFINE). As a holder of DCADMIN privilege, you can specify WITH GRANT OPTION when you grant definition privileges to allow the recipient to grant the same privileges to another user.

Note: For more information about administering privileges on system profiles, see the following sections in the chapter Syntax for Securing System Resources:

- GRANT System Definition Privileges (see page 282)
- REVOKE System Definition Privileges (see page 291)

Securing Resources That Can Be Categorized

About Categories

If you secure certain system resources internally, you must group occurrences of these resources in categories and grant execution privilege on the categories to allow access.

You create categories using CREATE RESOURCE CATEGORY statements. You authorize access with GRANT EXECUTE ON CATEGORY statements.

External Security

Categories are not meaningful to external security enforcement. However, you can choose to specify external security for any resource type that can be categorized.

If you specify external security for a resource type that can be categorized, you must also specify in the external security system rules for all occurrences of the resource type.

Resource Types That Can Be Categorized

This table shows resource types that can be categorized and the resource type keywords that you specify in the SRTT to secure them.

Note: Run units and access modules are secured internally by specifying resource type 'DB'.

Resource	**SRTT**	**keyword**
	Internal security	External security
Task	TASK	TASK
Load module	SLOD	SLOD
Access module(1)	DB(1)	SACC
Program	SPGM	SPGM
Run unit	DB(1)	NRU
Queue	QUEU	QUEU

(1) For more information about securing the DB resource type, see Securing Database Resources (see page 117).

Wildcards

To simplify the process of category management, you can use wildcards when you specify the resource occurrences to add to a category.

In this example, load modules, tasks, and queues associated with an accounts receivable application are added to a category:

```
create resource category ar
  add load module appldict.v0001.car*
  add task car
  add queue car*, ap* ;
```

You can also wildcard the category name when you grant privilege. For example, if you create several categories for the accounts receivable application and assign names that begin 'AR', you can grant privilege on all of the accounts receivable categories in this way:

```
grant execute on category ar*
  to ar_sys_admin;
```

Implementing Security by Category

The following are the steps to implement security using categories:

1. Create categories for groups of resource occurrences.
2. Grant users execution privilege on categories.
3. Activate internal security for each categorized resource type.

Securing Programs

About Programs

When you secure programs, you can control who can execute programs maintained in an operating system load library.

Until you secure programs, any user can execute a program in the operating system load library.

How to Secure Programs

To secure programs internally, include an entry in the SRTT:

```
#SECRTT   TYPE=ENTRY,                              X
    RESTYPE=SPGM,                              X
    SECBY=INTERNAL
```

To secure programs externally, include an entry in the SRTT:

```
#SECRTT   TYPE=ENTRY,                              X
    RESTYPE=SPGM,                              X
    SECBY=EXTERNAL,                             X
    Additional parameters required
```

Whether to Secure Programs

If you secure programs, only an authorized user can execute a user-mode program, including any CA IDMS user-mode program. Therefore, you should carefully weigh the requirements for administering program security.

For example, if you secure programs externally, you must identify to the external system all user-mode programs supplied by CA IDMS and site-specific application programs that users need to execute and specify the rules for securing these programs.

If you secure programs internally, you can take advantage of categories, wildcards, and groups to simplify this process.

Note: To identify CA IDMS user-mode programs, view the DLODSECR member of the installation source library.

Alternative to Program Security

The purpose of securing programs is to control access to data. An approach to protecting data at the program level that may be easier to administer is to secure databases or database occurrences:

- For program access to non-SQL defined databases, you categorize run units and grant execution privilege on the categories. The task of categorizing run units is simplified by the inclusion of database name and subschema name and program name in the run unit identifier. This approach is comparable to program registration in Release 10.2.

  Note: For run unit information about CA IDMS user-mode programs, view the DLODSECR member of the installation source library.

- For program access to SQL-defined databases, you grant applicable privileges on access modules.

Program Occurrence Overrides

You can specify occurrence overrides in the SRTT for the SPGM resource type. If you secure programs externally, you *must* add to the SRTT an occurrence override to unsecure the signon program (RHDCSNON). Without this override, any attempt to signon will fail.

In this example, security for programs is external, but occurrence overrides makes RHDCSNON and RHDCBYE unsecured:

```
#SECRTT   TYPE=ENTRY,              X
   RESTYPE=SPGM,                 X
   SECBY=EXTERNAL,                X
   Additional parameters required

#SECRTT   TYPE=OCCURRENCE,             X
   RESTYPE=SPGM,                 X
   RESNAME='RHDCSNON',              X
   SECBY=OFF

#SECRTT   TYPE=OCCURRENCE,             X
   RESTYPE=SPGM,                 X
   RESNAME='RHDCBYE',              X
   SECBY=OFF
```

Note: For more information, see the following sections:

- #SECRTT (see page 204), in the chapter "Syntax for Assembler Macros"
- CREATE RESOURCE (see page 267), in the chapter "Syntax for Securing System Resources"
- Category Security Processing (see page 108), in the chapter "Securing System Resources"

Securing Load Modules

About Load Modules

When you secure load modules, you control who can execute load modules that reside in the DDLDCLOD area of the dictionary.

Until you secure load modules, any user can execute a load module.

How to Secure Load Modules

To secure load modules internally, include an entry in the SRTT:

```
#SECRTT   TYPE=ENTRY,                           X
          RESTYPE=SLOD,                         X
          SECBY=INTERNAL
```

To secure load modules externally, include an entry in the SRTT:

```
#SECRTT   TYPE=ENTRY,                           X
          RESTYPE=SLOD,                         X
          SECBY=EXTERNAL,                       X
          Additional parameters required
```

Note: For more information, see the following sections:

- #SECRTT (see page 204), in the chapter "Syntax for Assembler Macros"
- CREATE RESOURCE (see page 267), in the chapter "Syntax for Securing System Resources"
- Category Security Processing (see page 108), in the chapter "Securing System Resources"

Securing Queues

About Queues

When you secure queues, you control who can access a queue. Until you secure queues, any user can access queues.

How to Secure Queues

To secure queues internally, include an entry in the SRTT:

```
#SECRTT   TYPE=ENTRY,                    X
    RESTYPE=QUEU,                        X
    SECBY=INTERNAL
```

To secure queues externally, include an entry in the SRTT:

```
#SECRTT   TYPE=ENTRY,                    X
    RESTYPE=QUEU,                        X
    SECBY=EXTERNAL,                      X
    Additional parameters required
```

Note: For more information, see the following sections:

- #SECRTT (see page 204), in the chapter "Syntax for Assembler Macros"
- CREATE RESOURCE (see page 267), in the chapter "Syntax for Securing System Resources"
- Category Security Processing (see page 108), in the chapter "Securing System Resources"

Queue Ownership

For runtime efficiency, queues are protected by ownership and categories. The user who creates the queue owns the queue.

Shared Queues

A queue can be shared if it is assigned to a category. Users with execution privilege on the category can share the queue. An unshared queue should not be assigned to a category. Ownership is used to protect unshared queues.

How Queue Security Works

When a user attempts to create a queue at runtime, the queue manager calls the security system to determine if the queue is assigned to a category.

If the queue is in a category:

- The queue is created if the user has execution privilege on the category, and the user ID is recorded as the owner of the queue.
- For subsequent access to the queue, the system compares the requester's user ID to the queue owner's user ID and does the following:
  - Grants access if the IDs match.
  - calls the security system, which attempts to verify that the requester user ID has execution privilege on the queue.

If the queue is *not* in a category:

- The queue is created and the user ID is recorded as the owner of the queue.
- For subsequent access to the queue, the system compares the requester's user ID to the queue owner's user ID and allows access only if they match.

The ownership mechanism allows security for unshared queues to be managed efficiently regardless of whether security for categorized queues is handled by CA IDMS or an external security facility.

Note: For more information about queues, see the following sections:

- CREATE RESOURCE (see page 267), in the chapter "Syntax for Securing System Resources"
- Category Security Processing (see page 108), in the chapter "Securing System Resources"

Securing Tasks

About Tasks

When you secure tasks, you control who can invoke a task.

Until you secure tasks, any user can invoke any task.

How to Secure Tasks

To secure tasks internally, include an entry in the SRTT:

```
#SECRTT   TYPE=ENTRY,                 X
   RESTYPE=TASK,                      X
   SECBY=INTERNAL
```

To secure tasks externally, include an entry in the SRTT:

```
#SECRTT   TYPE=ENTRY,                 X
   RESTYPE=TASK,                      X
   SECBY=EXTERNAL,                    X
   Additional parameters required
```

Note: For more information, see the following sections:

- #SECRTT (see page 204), in the chapter "Syntax for Assembler Macros"
- CREATE RESOURCE (see page 267), in the chapter "Syntax for Securing System Resources"
- Category Security Processing (see page 108), in the chapter "Securing System Resources"

Task Occurrence Overrides

In the SRTT you can specify occurrence overrides for the task resource. If you secure tasks externally, you *must* specify an occurrence override to unsecure the SIGNON task. Without this override, any attempt to sign on will fail.

Unsecured Tasks

After task security is activated, you can allow a user to execute certain tasks, whether or not the user is signed on, by creating a category of those tasks and granting execution privilege on the category to group PUBLIC.

If you create a category of tasks which all users can execute if they are signed on, grant the privilege to a group of all users that you explicitly create, not to group PUBLIC.

For external run units such as local utility jobs and access through client/server technology, you must grant execute authority on tasks RHDCNP3S and RHDCNP3J to group PUBLIC or all groups. Alternatively, you can turn off security for tasks RHDCNP3S and RHDCNP3J by including an entry in the SRTT.

Tasks Started by the System

If signon and tasks are secured, the identifier of the user who submitted the job to start the system must be authorized to execute tasks that are created directly or indirectly by startup and shutdown autotasks. An autotask is defined in the AUTOTASK statement of system generation.

No security checking is performed for an autotask. However, any task invoked as a result of autotask execution will cause a security check for the task.

If signon and tasks are secured internally, take these steps to ensure that the tasks can be executed:

1. If not yet done, define the user who submits the job to start the system.
2. If not yet done, grant signon privilege to the user.
3. Define a category and add the tasks invoked by autotasks.
4. Grant execution privilege on the category to the user.

Category Security Processing

Internal Category Numbers

When you create a category, the system represents the category internally by a halfword number. The system maintains category bit assignments to ensure that category numbers are assigned in ascending sequence. The number corresponds to a bit position in a Category bit map.

Category Bit Map

The runtime system maintains a category bit map for each user. The category bit map indicates the categories the user may access. When a user is granted access to a category, the corresponding bit in the category bit map is turned on.

The bit map is loaded on the first security request for a categorized resource which is internally secured. If none of the categorized resource types is internally secured, no category bit map will exist.

Security Check

When the user attempts to access a resource that is protected by category, the resource category is checked against the user's category bit map. Access is denied if the corresponding bit in the user's category bit map is off.

Implementing Application Security

About Activities

An activity is an application function defined as a resource to CA IDMS security. Activity security is an enhancement of the security class mechanism of Release 10.2.

You assign activity names and activity numbers to application functions with the CREATE RESOURCE STATEMENT. If you secure the ACTI resource internally, you grant users execution privilege on activities.

You can define up to 256 discrete activities for an application. You can choose to associate more than one function of the application with a given activity name and number.

Note: If an application needs more than 256 activities, you can use multiple application names. The limit of 256 per application name is designed to provide upward compatibility with Release 10.2 systems and to keep the scheme simple and efficient.

Securing Activities

Why You Secure Activities

When you secure activities, you control who can execute a given application function.

Until you secure activities, any user can execute any application function.

How to Secure Activities

To secure activities internally, include an entry in the SRTT:

```
#SECRTT   TYPE=ENTRY,                              X
    RESTYPE=ACTI,                            X
    SECBY=INTERNAL
```

To secure activities externally, include an entry in the SRTT:

```
#SECRTT   TYPE=ENTRY,                              X
    RESTYPE=ACTI,                            X
    SECBY=EXTERNAL,                             X
    Additional parameters required
```

Note: For more information, see #SECRTT (see page 204) and CREATE RESOURCES.

About External Activity Security

The EXTNAME values that you can specify for RESTYPE ACTI are RESNAME, APPLNAME, and ACTIVITY. For the purposes of security checking, the value of both RESNAME and APPLNAME is *application-name*. Only the ACTIVITY value contains the application function number. Thus, to secure individual activities externally, you must include ACTIVITY in the SRTT entry for RESTYPE ACTI.

Defining an Activity Resource

If you assign activity numbers to functions within the application, you can define each activity as a system resource with a CREATE RESOURCE statement. In a CREATE RESOURCE statement, you associate the application activity number with an external activity name. The activity name can be up to 18 characters and must be qualified with the application name, as in this example:

```
create resource
     activity dcmt.vary_terminals
     number 14;
```

Granting Execution Privilege on the Activity

After you have defined an activity, you can give users the privilege of executing the application functions represented by the activity, using a GRANT statement as in this example:

```
grant  execute
     on  activity  dcmt.vary_terminals
     to  support;
```

Activity Security Processing

Activity Bit Map

For each application that the user can access, the system maintains a bit map that indicates the activities the user is allowed to execute. The activity number corresponds to a bit position; for example, bit 43 corresponds to activity 43.

If the ACTI resource is secured internally, the application activity access bit map is brought into memory when the application issues its first security check request. The application bit map is chained to the signon block, and user and group activity access authorities are merged into the bit map.

When a user attempts to execute an activity, the application issues a call for a security check to the central security interface, specifying the application name and activity number. The application is expected to enforce the activity security based on the return code from the central security interface.

Internal Security Check on an Activity

If the ACTI resource is secured internally, the system searches for the activity bit map for the application and the user when the first security check for an application is processed. If the bit map is found, the system checks the activity number against the corresponding bit position in the activity access bit map, and returns a YES or NO answer to the application.

If the bit map is not found, the system then looks for an activity bit map for the user and the application named 'DEFAULT.' If found, the bit map for the DEFAULT activity is used in the security check.

Application DEFAULT

The DEFAULT application is a mechanism that allows execution of existing applications that use Release 10.2 security classes without having to explicitly define activities for each application. The security system will allow execution of an application function if:

- The security class assigned to that function matches the activity number of an activity in the DEFAULT application.
- The user has been granted execution privilege on that activity.

You can use the RHDCSMIG program to generate statement syntax to create 255 DEFAULT application activities (activity numbers 1 through 255) and to grant execution privilege on the activities to users who have the matching security classes in the Release 10.2 dictionary.

Note: For more information about RHDCSMIG, see the *CA IDMS Conversion Guide*.

External Security Check on Activity

If the ACTI resource is secured externally, no activity bit map is involved in security checking. The central security interface creates the identifier of the activity that it passes to the external system by concatenating the shorter of the application name or the first five characters of the application name with the three-digit function number supplied by the application.

CA ADS Security

Application Name

The *application-name* you specify when defining a CA ADS activity must be the ADB name; that is, the name specified when the application was defined with the CA ADS application compiler (ADSA).

Note: The ADB name may differ from the task code or codes of the application. If necessary, you can determine the ADB name by viewing the application name by displaying the TAT table with DCMT DISPLAY MEMORY PROGRAM $ACF@TAT.

CA ADS Security Classes

CA ADS allows you to associate a security class with the application and with any one or more application responses.

If the application has been assigned a security class, a security check is requested when a user attempts to execute the application. The user must be authorized to execute the activity whose activity number matches the security class of the application.

Similarly, if a response has been secured with a security class, a security check is requested when a user attempts to execute the response. The user must be authorized to execute the activity whose activity number matches the security class of the response.

Note: For more information about assigning security classes in CA ADS, see the *CA ADS Reference Guide*.

DCMT Security

Application Name

When you create an activity name for the DCMT application, you specify DCMT for *application-name*. You associate the activity number with the DCMT commands you are grouping in the activity as follows.

Assigning DCMT Activity Numbers

DCMT provides the #CTABGEN macro for assigning activity numbers to DCMT commands. In the #CTABGEN macro, you associate an activity number with a DCMT command code.

#CTABGEN Example

In this example, #CTABGEN assigns the activity number of 14 to the DCMT commands as represented by their command codes—N028 (VARY LTERM), N029 (VARY PTERM), and N030 (VARY LINE):

```
#CTABGEN (N028,14,N029,14,N030,14)
```

Note: For #CTABGEN syntax, DCMT command codes, and information about generating the #CTABGEN module, see Syntax for Assembler Macros (see page 161).

Release 10.2 DCMT Security

If you have implemented security classes for DCMT commands in Release 10.2, you need only reassemble #CTABGEN under Release 16.0 and define activities either specifically for DCMT or for the DEFAULT application, using RHDCSMIG output to generate CREATE RESOURCE ACTIVITY statements.

OCF/BCF Security

Application Name

When you create an activity name for a utility command, specify an application name of OCF for activities that are to be secured when running under a CV. To secure batch local mode utility activities, specify BCF for an application name. If the same command is to be secured in the same manner in both CV and batch, then two activity resources must be created.

Assigning OCF/BCF Activity Numbers

OCF/BCF security provides the #UTABGEN macro for assigning activity numbers to OCF/BCF utility commands. In the #UTABGEN macro, you associate an activity number with an OCF/BCF command code.

#UTABGEN Example

In this example, #UTABGEN assigns the activity number of 14 to the OCF/BCF commands FORMAT and PRINTPAGE as represented by their command codes:

```
#UTABGEN (FORMAT,14,PRINTPAGE,14)
```

Note: For #UTABGEN syntax, OCF/BCF command codes, and information about generating the #UTABGEN module, see Syntax for Assembler Macros (see page 161).

Online Debugger Security

Application Name

When you create an activity name for the online debugger, you specify DBUG for *application-name*. You associate the activity number with the online debugger commands you are grouping in the activity as follows.

Assigning Online Debugger Activity Numbers

The CA IDMS/DC online debugger provides the #GTABGEN macro for assigning activity numbers to online debugger functions. In the #GTABGEN macro, you associate an activity number with an online debugger security Category.

#GTABGEN Example

Assume that you wish to assign the activity number of 20 to online debugger functions as represented by two online debugger security categories:

- AUPGMR (CA ADS user programs can be retrieved)
- USTGR (User storage can be retrieved)

To make this assignment, you would issue this #GTABGEN macro:

```
#GTABGEN(AUPGMR,20,USTGR,20)
```

Note: For #GTABGEN (see page 193) syntax and online debugger security categories, see the chapter Syntax for Assembler Macros.

Implementing Multi-level Application Security

Multi-level Security

You can secure applications at several levels.

For example, an online application with embedded SQL can be secured by the following:

- Task—The ability to invoke the application
- Load module—The ability to execute an application program
- Access module—The ability to execute embedded SQL that accesses a database

DCMT Example

When you analyze your site's DCMT security requirements, keep in mind that you can implement security for DCMT commands at these levels:

- **At the task level**, you secure the DCMT task by assigning the task a Category using security administration statements.
- **At the program level**, you secure programs invoked for the DCMT task by assigning the programs a Category using security administration statements. Programs invoked for DCMT requests all have names that begin with RHDCMT (for example, RHDCMTPT or RHDCMTTI).
- **At the DCMT command level**, you secure DCMT commands by means of the #CTABGEN macro. This macro is assembled into the program IDMSCTAB and allows you to apply discrete security to specific DCMT commands (such as, DCMT VARY PROGRAM) and also to individual command options (such as, DCMT VARY PROGRAM STORAGE PROTECT). The macro is used in conjunction with activity security in the security system to control access to specific DCMT functions.

Chapter 9: Securing Database Resources

This section contains the following topics:

About Database Security

What is Required

Any user has the ability to access any database until it is secured. You can secure a database only by securing database resources.

You can secure online access to databases at the task level, although this would involve securing the OCF task (online Command Facility) and other tasks. However, task security does not secure databases from batch access.

Even if the configuration of your external security system protects the database against local mode access, the database would not be protected from batch access through the central version unless the database is secured through CA IDMS centralized security.

Dictionaries and User catalog

The system dictionary, application dictionaries, and the user catalog are databases. If internal security is specified for one or more global resources, the user catalog contains security definitions. If internal security is specified for one or more database resources, the system dictionary contains security definitions, as does the application dictionary if the SQL Option is installed.

To secure dictionaries and the user catalog, you must secure database resources as described in this chapter.

About Database Resources

What You Should Know

Before you implement a scheme for database security using information in this chapter, become familiar with the concepts discussed in this section.

Securing Database Resources in the SRTT

Database Resources

The security option in the SRTT for the DB resource type determines whether database resources other than DMCLs and database name tables (DBTB) are secured externally, internally, or not at all.

SRTT entries for resource types other than DB, DMCL, and DBTB are used in runtime security processing only if security for DB is external; in that case, the SRTT entry is used only to determine the external resource class and resource name to send with the security check request to the external system.

The following table lists CA IDMS database resources and their keyword equivalents for the #SECRTT RESTYPE parameter:

Database resource	RESTYPE keyword to secure resource	RESTYPE keyword for external information
Database	DB	DB
DBADMIN privilege	DB	Not applicable
Access module	DB	DACC
Area	DB	AREA
Run unit	DB	NRU
SQL-defined schema	DB	QSCH
Non-SQL-defined schema	DB	NSCH
Table	DB	TABL
Database name table	DBTB	DBTB
DMCL	DMCL	DMCL

DB Occurrence Overrides

You can specify DB occurrence overrides in the SRTT. For example, if security for databases is off but you add an internal security occurrence override for database PROD, the runtime system will route a security check on a database resource to internal security if the database name on the current security request begins with 'PROD'.

You *cannot* override the automatic assignment of the DB security option to the other database resources. For example, if security for DB is off, security for the AREA resource type is also off and the security option specified on the SRTT entry for AREA is ignored. However, external resource class and name information in an SRTT entry for a database resource type such as AREA is used if external security is specified on the entry for DB or on a DB occurrence override.

Database Security and Database Names

Segment Names and Database Names

The database name specified on a BIND RUN-UNIT statement or a CONNECT statement can be either a segment name or a database name defined in the database name table. If you secure all databases, a security check will be routed to the enforcing system on BIND RUN-UNIT statements and on database definition and access statements issued following a CONNECT.

Note: To issue the CONNECT statement itself under the central version, the user must have signon authority for the system with which the dictionary named in the statement is associated and authority to invoke the task or application from which the CONNECT is issued.

However, if you plan to leave some databases unsecured, you must consider how CA IDMS processes a database name before you build database security in the SRTT.

Role of the Database Name Table

If an application requests a bind to a database or a connection to a dictionary, CA IDMS searches the database name table for the name specified on the BIND or CONNECT. If it finds a match, CA IDMS determines the areas and files to be accessed based on the segments that are included in the database name. If it does not find a match in the database name table, CA IDMS searches for a matching segment name in the DMCL. If no match is found, an error results.

Securing Access to Individual Segments

To understand how access to segments is secured, consider this sample database name table:

Database name	Segments
SYSTEM	SYSTEM CATSYS SYSMSG
DIRLDICT	DIRLNWK CATSYS SYSMSG

If the entry for DB is security '*OFF*', you would obtain these results using occurrence overrides:

- **If you secure 'SYSTEM',** access to the SYSTEM segment is secured. Access to CATSYS and SYSMSG through dbname SYSTEM is secured, but access to these segments directly or through dbname DIRLDICT is not secured.
- **If you secure 'SYSTEM' and 'DIRLDICT',** access to the SYSTEM segment is secured. Access to CATSYS, SYSMSG, and DIRLNWK through dbnames is secured, but direct access to these segments is not secured.
- **If you secure 'SYSTEM', 'DIRLDICT', 'CATSYS', and 'SYSMSG',** access to all segments but DIRLNWK is secured.

Therefore, to achieve complete database security using occurrence overrides, you must secure all segments to be protected and all dbnames that include one or more of those segments.

Securing the Database Name Table

To maintain database security that is based on occurrence overrides, you must secure database name tables that are included in DMCLs. If a database name table is not secure, a knowledgeable user could create or modify the definition of a database name that is not secured to include otherwise secure segments.

Note: For more information, see Securing Database Name Tables (see page 128).

Internal Security for Database Resources

Privileges on Common Database Resources

The following table presents the privileges in CA IDMS internal security that apply to use of database resources common to both SQL-defined and non-SQL-defined databases:

Privilege	DB	AREA	DMCL	DBTABLE
CREATE	X		X	X
ALTER	X		X	X
DROP	X		X	X
DISPLAY	X		X	X
USE	(1)	(1)	X	X
DBAREAD		X		
DBAWRITE		X		
DBADMIN	X			

(1) Privilege applicable only to non-SQL-defined databases.

DBADMIN can be granted to any other user by a holder of SYSADMIN or DBADMIN. All other privileges are grantable if a holder of SYSADMIN or DBADMIN grants them using the WITH GRANT OPTION parameter. A grantable privilege means that the recipient of the privilege can grant it to another user.

Definition Privileges

CREATE, ALTER, DROP, and DISPLAY control the user's ability to manipulate the definition of an object. To issue any definition statement other than DISPLAY on the common database resources, the user must also hold DBADMIN authority on the dictionary to which the session is connected when the statement is issued, if DB security is enabled for the dictionary.

USE Privilege

The following table explains the type of access that the USE privilege authorizes:

Resource	What USE privilege permits the user to do
DB	Associate a secured segment with an SQL schema
NSCH (1)	Associate a secured non-SQL-defined schema with an SQL schema

Resource	What USE privilege permits the user to do
AREA	Create an SQL table or index in a secured area
DMCL	Punch the load module of a secured DMCL and execute utilities on the journal files defined by that DMCL
DBTABLE	Punch the load module of a secured database name table and associate a database name table with a DMCL.

(1) NSCH is a common database resource in the sense that it represents a non-SQL-defined entity and is meaningful in SQL processing.

DBAREAD and DBAWRITE Privileges

The DBAREAD and DBAWRITE privileges are granted to permit users to execute utility functions on areas of the database. DBAREAD privilege allows the user to execute utilities that require read-only access to an area. DBAWRITE privilege allows the user to execute utilities that require read-write access to an area.

Securing Common Database Resources

About the DB Resource

Security on the DB resource type in the SRTT automatically activates security on the following resources:

- Database
- DBADMIN
- Area
- Run unit
- SQL schema
- Non-SQL-defined schema
- Table
- Access module

Internal Security for Databases

If you secure databases internally, you grant privileges to allow users access to database resources.

External Security for Databases

If you secure databases externally, you specify rules in the external security system for accessing database resources. You must also provide external resource class and name information in the SRTT entry for the database resources represented by these resource type keywords:

- DB
- AREA
- NRU
- QSCH
- NSCH
- TABL
- DACC
- SACC

Note: For more information, see #SECRTT (see page 204).

Securing Databases

About Databases

When you secure resource type DB, you control who can issue DDL SEGMENT statements and who can specify a segment in the DBNAME parameter of a CREATE SCHEMA statement. Until you secure resource type DB, any user can issue DDL SEGMENT statements and can specify a segment in the DBNAME parameter of a CREATE SCHEMA statement.

How to Secure Databases

To secure the DB resource internally, include an entry in the SRTT:

```
#SECRTT   TYPE=ENTRY,                        X
   RESTYPE=DB,                          X
   SECBY=INTERNAL
```

To secure the DB resource externally, include an entry in the SRTT:

```
#SECRTT   TYPE=ENTRY,                        X
   RESTYPE=DB,                          X
   SECBY=EXTERNAL,                         X
   Additional parameters required
```

Note: For more information about #SECRTT, see #SECRTT (see page 204).

Database Occurrence Overrides

You can specify a security option for a particular occurrence of a database that differs from the option specified for DB in the SRTT. This allows you, for example, to secure databases internally but to leave security 'OFF' for specific databases.

In this example, internal security is activated in the SRTT for all databases in the system (including the system dictionary and the user catalog), but security is turned off for any databases with names that begin with 'TEST' or 'DEMO'.

```
#SECRTT  TYPE=ENTRY,               X
   RESTYPE=DB,                 X
   SECBY=INTERNAL

#SECRTT  TYPE=OCCURRENCE,               X
   RESTYPE=DB,                 X
   RESNAME='TEST',               X
   SECBY=OFF

#SECRTT  TYPE=OCCURRENCE,               X
   RESTYPE=DB,                 X
   RESNAME='DEMO',                X
   SECBY=OFF
```

How to Grant Database Definition Privilege

To give physical database definition privileges, you issue a GRANT statement on the DB resource type, specifying the privilege or privileges and the name of the database. You can specify any combination of CREATE, ALTER, DROP, DISPLAY, and USE privileges, or you can specify all definition privileges (DEFINE). You must be connected to the system dictionary.

As a holder of SYSADMIN or DBADMIN privilege, you can specify WITH GRANT OPTION when you grant definition privileges to allow the recipient to grant the same privileges to another user.

Note: For more information, see the following sections:

- GRANT Physical Database Definition Privileges (see page 305)
- REVOKE Physical Database Definition Privileges (see page 326)

Performance Advantage

You may gain a performance advantage by using an override to turn off security for an occurrence of a secured resource type. Runtime security processing checks for an occurrence override in the SRTT before checking resource authorizations in the security database.

Securing DBADMIN

About DBADMIN

When you secure resource type DB, you control who can manipulate database definitions and database-related objects. Until you secure DBADMIN, any user can manipulate database definitions and database-related objects.

How to Secure DBADMIN

You secure DBADMIN by securing the DB resource.

Note: For more information, see Securing Databases (see page 123).

How to Grant DBADMIN Privilege

You give DBADMIN privilege on a named database to a user or group with a GRANT DBADMIN statement. You must hold SYSADMIN or the appropriate DBADMIN privilege to grant DBADMIN privilege. You must be connected to the system dictionary.

Note: For more information, see the following sections:

- GRANT Administration Privilege (see page 298)
- REVOKE Administration Privilege (see page 320)

Securing Areas

About Areas

When you secure resource type DB, you can control who can access an area through a CA IDMS utility and who can create tables and indexes to be stored in the area. Until you secure resource type DB, any user can access an area through a CA IDMS utility and create tables and indexes to be stored in the area.

How to Secure Areas

You secure areas by securing the DB resource.

Note: For more information, see Securing Databases (see page 123).

If you secure areas externally, you must also include an entry in the SRTT with external security information for resource type AREA.

How to Grant Area Access and Use Privileges

To give area access privileges, you issue a GRANT statement on the area resource type, specifying the privilege or privileges and identifying the area. You can specify any combination of DBAREAD, DBAWRITE, and USE privileges.

DBAREAD and DBAWRITE privileges allow read-only and read-write access to an area using CA IDMS Utilities.USE privilege allows creation of a table or index in the area.

As a holder of SYSADMIN or DBADMIN privilege, you can specify WITH GRANT OPTION when you grant these privileges to allow the user to grant the same privileges to another user. You must be connected to the system dictionary.

More Information:

- For more information, see GRANT Area Access Privileges (see page 301).
- For more information, see REVOKE Area Access Privileges (see page 323).
- For more information about utilities, see the *CA IDMS Utilities Guide*.
- For more information about creating tables and indexes, see the *CA IDMS SQL Reference Guide*.

Securing DMCLs

About DMCLs

When you secure the DMCL resource type, you can control who can do the following:

- Issue DDL DMCL statements
- Display or punch the DMCL load module
- Execute utilities which operate against DMCL journal files

Until you secure the DMCL resource type, any user can issue DDL DMCL statements if database security is not in effect, and any user perform the other previously listed functions.

How to Secure DMCLs

If the system dictionary is secured, DBADMIN privilege on the system dictionary is required to manipulate DMCL definitions.

Note: For more information, see Securing the Dictionaries and the User catalog (see page 149).

The following discussion applies to securing DMCLs explicitly.

To secure DMCLs internally, include an entry in the SRTT:

```
#SECRTT   TYPE=ENTRY,                    X
   RESTYPE=DMCL,                        X
   SECBY=INTERNAL
```

To secure DMCLs externally, include an entry in the SRTT:

```
#SECRTT   TYPE=ENTRY,                    X
   RESTYPE=DMCL,                        X
   SECBY=EXTERNAL,                       X
   Additional parameters required.
```

Note: For more information, see #SECRTT (see page 204).

How to Grant DMCL Definition and Use Privileges

To allow a user to create and maintain a DMCL definition, you issue a GRANT statement on the DMCL resource type, specifying the privilege or privileges and identifying the DMCL.

You can specify any combination of CREATE, ALTER, DROP, DISPLAY, and USE privileges, or you can specify all privileges (DEFINE).

Note: The USE privilege allows the user to punch the DMCL load module and format journal files defined in the DMCL.

As a holder of the applicable SYSADMIN or DBADMIN privilege, you can specify WITH GRANT OPTION when you grant these privileges to allow the recipient to grant the same privileges to another user. You must be connected to the system dictionary.

More Information:

- For more information, see GRANT Physical Database Definition Privileges (see page 305).
- For more information, see REVOKE Physical Database Definition Privileges (see page 326).
- For more information about DMCL statements, see the *CA IDMS Database Administration Guide*.

Securing Database Name Tables

About Database Name Tables

When you secure the database name table resource type, you control who can issue DBTABLE definition statements and who can display or punch the DBTABLE load module. Until you secure the database name table resource type, any user can issue DBTABLE definition statements if database security is not in effect, and any user can display or punch the DBTABLE load module.

How to Secure Database Name Tables

If the system dictionary is secured, DBADMIN privilege on the system dictionary is required to manipulate database name table definitions.

Note: For more information, see Securing the Dictionaries and the User catalog (see page 149).

The following discussion applies to securing database name tables explicitly.

To secure database name tables internally, include an entry in the SRTT:

```
#SECRTT  TYPE=ENTRY,                X
  RESTYPE=DBTB,                  X
  SECBY=INTERNAL
```

To secure database name tables externally, include an entry in the SRTT:

```
#SECRTT  TYPE=ENTRY,                X
  RESTYPE=DBTB,                  X
  SECBY=EXTERNAL,                  X
  Additional parameters required
```

Note: For more information, see #SECRTT (see page 204).

How to Grant Database Name Table Definition Privilege

To allow a user to create and maintain a database name table definition, you issue a GRANT statement on the database name table resource type, specifying the privilege or privileges and identifying the database name table. You can specify any combination of CREATE, ALTER, DROP, DISPLAY, and USE privileges, or you can specify all privileges (DEFINE).

Note: The USE privilege allows the user to punch the database name table load module and to associate the database name table with a DMCL.

As a holder of the applicable SYSADMIN or DBADMIN privilege, you can specify WITH GRANT OPTION when you grant these privileges to allow the recipient to grant the same privileges to another user. You must be connected to the system dictionary.

More Information:

- For more information, see GRANT Physical Database Definition Privileges (see page 305).
- For more information, see REVOKE Physical Database Definition Privileges (see page 326).
- For more information about DBTABLE statements, see the *CA IDMS Database Administration Guide*.

Securing Access to Non-SQL-Defined Databases

How to Do It

To secure access to a non-SQL-defined database, these resource types must be secured:

- DB
- DBADMIN* (internal security)
- AREA*
- NRU*

Depending on your database definitions and runtime environment, you may also need to secure these resource types:

- DBTB
- DMCL
- NSCH*

* Resource type automatically secured if security for DB is activated.

Note: For more information about securing DB, DBADMIN, AREA, DBTB, and DMCL, see Securing Common Database Resources (see page 122).

Other Security Techniques

CA IDMS also supports security techniques such as compiler security and database procedures that were supported prior to Release 12.0.

Securing Run Units

About Run Units

The primary access to data stored in non-SQL-defined databases is through a subschema using navigational DML. This type of database transaction is a run unit.

A run unit is started by a BIND RUN-UNIT command, which effectively opens a logical view of the database as defined by the subschema. Within the subschema, privacy locks and LRF path logic restrict the operations which can be performed on the data accessible through the subschema.

When you secure run units, you control who can access a non-SQL-defined database through navigational DML. Until you secure run units, any user can access a non-SQL-defined database through navigational DML.

About the Run Unit Resource

At bind time, the following elements are available for security checking purposes:

- The name of the database being accessed
- The name of the subschema
- The original compilation name of the program issuing the BIND

The combination of these three elements—*database-name.subschema-name.program-name*—identify the run unit resource on which authority is checked when access to a non-SQL-defined database using navigational DML is requested.

How to Secure Run Units

You secure run units by securing the DB resource.

Note: For more information, see Securing Databases (see page 123).

If you secure run units externally, you must also include an entry in the SRTT with external security information for resource type NRU.

How to Grant Execution Privilege on a Run Unit

Run units must be categorized before you can grant execution privilege on them. For example, this statement assigns run units for a group of programs that access the PRODSCHM database to the Category EMPINFO:

```
create resource
  Category empinfo
```

Note: For more information, see CREATE RESOURCE (see page 267).

To allow a user to execute a run unit, you issue a GRANT EXECUTE statement on the Category that contains the run unit. For example, this statement gives privilege on the EMPINFO Category to two groups of users:

```
grant execute on
  Category empinfo
  to hr_mgrs, corp_execs;
```

Note: For more information, see GRANT Execution Privilege (see page 277).

Runtime Run Unit Checking

If the security option for DB is internal, BIND processing requests a check of privilege on the Category to which the run unit is assigned. The internal security system uses dynamic table support to maintain a sorted cache of run units and their associated Category.

If the security option for DB is external, the check request is routed to external security with the class and resource name information specified on the SRTT entry for NRU.

Schema and Subschema Security

You can control who can use CA IDMS compilers to define non-SQL-defined schemas and subschemas by implementing internal dictionary security.

Note: For more information about dictionary security, see Securing Application Dictionary Resources (see page 153).

SQL Access to a Non-SQL-Defined Database

With the CA IDMS SQL Option, a user can access a non-SQL-defined database using SQL. If the database is secured, you allow SQL access to a non-SQL database by creating an SQL schema for a non-SQL schema (this requires CREATE privilege on the SQL schema, USE privilege on the non-SQL schema) and granting table access privileges to other users.

Note: For more information, see the following sections:

- Securing SQL schemas (see page 143)
- Securing Non-SQL-defined Schemas (see page 144)

Using Database Procedures

What You can Do

You can use a database procedure to enforce user-defined security rules for non-SQL-defined databases. The procedure can make use of centralized security to actually do the checking which in turn will invoke an external package if appropriate.

What is a Database Procedure

A database procedure is a user-written exit specified as part of the schema definition of the non-SQL-defined database. Both area- and record-level procedures are supported. They allow you to specify which non-SQL DML verbs will cause invocation of the procedure.

Advantages of Database Procedures

- You can choose resource names for records which are meaningful in your environment.
- You can selectively check privileges based on the record or area and the type of access being performed, limiting the additional overhead incurred to only those resources which require a higher degree of security checking.
- Within the procedure you can apply additional rules based on such information as the content of the record (the equivalent of row-level security), the program name being used, and the contents of the BIND PROCEDURE block as passed from the application program.
- Facilities for caching information are provided.

SQL Security Enforcement

Overview

The security scheme for SQL processing is both based upon, and compliant with, the ANSI SQL standard.

The major secured resource in SQL processing is the table resource type. CA IDMS internal security validates a user's right to access a table according to the rules specified in the ANSI SQL standard. Additional facilities control table creation and alteration.

If resources accessed in SQL processing are secured externally, rules for accessing each resource type must be defined to the external security system. Reference to external security in the following discussion is limited to information about how CA IDMS centralized security processes a request for an externally secured resource.

SQL Resources

Table

The table resource type represents base tables, functions, procedures, table procedures and views.

A view is a logical table derived from information in one or more base tables, table procedures or views. CA IDMS centralized security does not distinguish between table-like objects, but special security considerations pertain to view access.

Note: For more information, see Securing Views (see page 139).

Access Module

The access module resource represents a set of precompiled SQL statements. The owner of the access module must hold all necessary table access privileges in order to execute the access module. The access module owner can grant execution privilege to other users if the owner holds grantable table access privileges.

Note: For a more detailed discussion of access module security, see Runtime Security for Access Modules (see page 137).

Schema

The schema resource is equivalent to the ANSI SCHEMA construct. All tables, functions, procedures, table procedures, views, and access modules are contained within a schema. The schema name becomes the high-level name qualifier for all subordinate entities.

The schema also designates ownership of resources. The owner of a schema owns all resources within the schema. Ownership is established when the schema is created.

CA IDMS Privileges

Privileges and Resources

The following tables summarizes the resources and CA IDMS privileges that apply to those resources in SQL processing if the resources are secured internally:

Privilege	TABLE	SCHEMA	ACCESS MODULE
SELECT	X		
INSERT	X		
UPDATE	X		

Privilege	TABLE	SCHEMA	ACCESS MODULE
DELETE	X		
EXECUTE			X
CREATE	X	X	X
ALTER	X	X	X
DROP	X	X	X
DISPLAY	(1)	(1)	(2)
REFERENCES	X		

(1) Privilege to display the resource and privileges on it.

(2) Privilege to issue the EXPLAIN statement on the module and to display the resource and privileges on it.

All privileges are grantable when a holder of SYSADMIN or DBADMIN privilege grants them using the WITH GRANT OPTION parameter. This allows the recipient of the privilege to grant it to another user.

CA IDMS internal security specifically checks for grantability of privileges when it processes a security check on view and access module resource types.

Note: For more information about runtime security checks on views and access modules, see the following sections:

- Securing Views (see page 139)
- Runtime Security for Access Modules (see page 137)

Access Privileges

SELECT, INSERT, UPDATE, and DELETE privileges control a user's ability to access data. These privileges are defined according to the ANSI SQL standard.

Definition Privileges

CREATE, ALTER, DROP, DISPLAY, and REFERENCES control the user's ability to manipulate the definition of an object or, in the case of REFERENCES, control a user's ability to reference a table in a referential constraint definition.

Access Module Execution Privilege

The EXECUTE privilege allows the user to execute an access module. The privilege to execute an access module can also be held through the Category mechanism.

Note: For more information, see Securing Resources That can Be categorized (see page 100).

If an access module has been assigned to a Category, a user must hold privilege on the Category to execute the access module. In this situation, an individual grant of execution privilege on the access module is ignored by the security system as long as the Category exists and the access module remains in it.

Security Checking for Interactive and Dynamic SQL

Dynamic Checking

When a user executes a tool which allows SQL statements to be entered explicitly (or implicitly as a result of information provided on a form), security checking is performed as each statement is processed. This is also true if a user-written program issues dynamic SQL statements.

The authorization ID against which the privileges are checked is the authorization ID of the executing user, except in certain instances associated with view access, as discussed in Securing Views (see page 139). Every privilege required to execute a given SQL statement is checked. The results of the check are cached for the life of the database transaction, to avoid repetitive authorization checks for similar access to the same table-like object.

Securing the Dynamic SQL Statement cache

If dynamic SQL statement caching is in effect, users and administrators should keep in mind that the SQL cache contains images of the source of dynamic SQL statements. These images can possibly contain confidential information. The source of the cached SQL statements is accessible through the table procedure SYSCA.DSCCACHE and any view (that is SYSCA.DSCCACHEV delivered during installation), that projects the STATEMENT column of this table procedure. It is recommended to secure access to the SYSCA.DSCCAHE table procedure and any view exposing the column STATEMENT of SYSCA.DSCCACHE.

External Security

If security on the database being accessed is controlled externally, the security checks are issued by CA IDMS as it executes the commands and the authorization permissions are cached for the life of the transaction or task whichever ends first.

CA IDMS Internal Security

If security on the database being accessed is controlled by CA IDMS internal security, security checks are issued by the access module compiler (AMC) as it compiles the dynamic SQL command.

Security Checking for Precompiled SQL Statements

Precompiled SQL Statements

User-written programs or SQL routines may contain embedded SQL statements that are precompiled and included in an access module prior to runtime. Security checking for embedded SQL statements is performed in one of two ways, depending on how security on the database being accessed is controlled.

External Security

If external security is in effect for the database, dynamic security checking is performed on all SQL statements, precompiled or not. When the SQL session is started, a security check determines if external security is in force for the database to which the session is connected. If so, this information is cached for the duration of the session.

CA IDMS issues the security checks as it executes each statement and caches the information for the life of the database transaction or task. The name of the access module is passed as part of the security check and is used as an authorized program filter.

This method of security checking for SQL statements complies with government requirements.

CA IDMS Internal Security

If CA IDMS internal security is in effect for the database, security checking for precompiled SQL statements takes a pre-authorized approach that requires the owner of the access module to hold all privileges necessary to execute every SQL statement in the module. For example, the owner must hold the appropriate table access privilege for each table accessed by an SQL statement in the module.

If this condition is met, then the owner of the access module can execute it. The owner can give execution privilege on the access module to other users if the owner holds the necessary grantable privileges.

Advantages of the Pre-authorized Approach

The pre-authorized security approach for SQL statements minimizes the overhead of security checking at runtime.

It also eliminates the need to grant all users the privileges needed to execute the SQL statements in the access module. Only the owner must have those privileges; other users simply require execution privilege on the access module. This means that executing the program is the only way the users can access the resources because they hold no privileges independent of the access module.

The pre-authorized security approach for SQL statements complies with the ANSI SQL standard.

Runtime Security for Access Modules

Overview

Each time a new copy of the access module is physically loaded by the runtime system, the privileges of the access module owner are checked.

The result of the security check performed on a new copy of an access module is the status of the access module. This information is cached in the PDE until a new copy is loaded (or until the system is recycled).

The status of the access module is one of the following:

- Not runnable—No one can execute the access module.
- Runnable—The owner holds all necessary privileges to execute the access module.
- Runnable/grantable—The owner holds all necessary privileges to execute the access module and to grant execution privilege on the access module to other users.

Runtime checking is required because a grantable privilege needed to pass execution privilege to users could be removed from the owner. Revocation of a privilege occurs independently of both CA IDMS and CA IDMS inter security.

A detailed description of runtime procedures for access module security follows.

On a Load of an Access Module

On a load of *any* access module, the CA IDMS program load function issues two security check requests to verify that the user has execution privilege for the access module. The first security check is for load privilege on the access module. An access module is just like any other load module. When loaded, you need to check to see if the user has the authority to load the program. The second security check is issued to see if the requestor has authority to access the database using the SQL statements that are stored in the access module.

- If the checks fail, an error is returned.
- If the checks succeed, normal load processing occurs.

If a *new* copy of the access module is being loaded and the security check on the user succeeds, CA IDMS program load processing:

1. Calls a database routine to scan the module and return this information about the access module:

   - Status (Not runnable, runnable, runnable/grantable).
   - Authorization ID of the owner.

   This information is cached.

2. The CA IDMS program load function requests a check to determine if the user is allowed to execute the access module based on its status:

   - If status is not runnable, then the check fails.
   - If status is runnable/grantable, then the check succeeds.
   - If status is runnable and the user (or a group to which the user belongs) is the owner of the access module, then the check succeeds; if status is runnable but the user is not the owner, the check fails.

3. Depending on the result of the preceding step, the program requesting the load receives a return code indicating one of these conditions:

   - Security violation—The user does not have execution privilege on the access module.
   - Runnable—The user has execution privilege on the access module.
   - Not grantable—The user cannot execute the access module because the owner does not hold all required privileges.

On a CREATE or ALTER ACCESS MODULE Command

The access module compiler issues a security check to determine whether the user has the CREATE or ALTER privilege on the access module. If not, an error is issued and no further processing is done.

If the security check succeeds, the access module compiler creates the access module. It then calls the database routine used by the CA IDMS program loader function to check the access module owner's privileges. The owner of the access module is the owner of the associated schema. The user who submits the CREATE or ALTER ACCESS MODULE statement is notified if the access module owner lacks any required privileges.

Note: The access module is stored whether or not the owner holds all required privileges. If you subsequently grant the owner any missing privileges, the access module will be runnable.

On Dynamic Compilation of an SQL Statement

There are three situations that require dynamic compilation of SQL statements:

- The statement is submitted through the Command Facility.
- A user-written program submits a dynamic SQL statement.
- CA IDMS determines that database changes require an access module to be recompiled.

In these situations, the access module compiler checks privileges as the statements are being compiled. The result of this checking is based on the same criteria used for a load:

- If status is not runnable, then the check fails.
- If status is runnable/grantable, then the check succeeds.
- If status is runnable and the user (or a group to which the user belongs) is the owner of the access module, then the check succeeds; if status is runnable but the user is not the owner, the check fails.

Securing Views

Views in Security Strategy

There are special security considerations associated with creating and accessing views. A view is a logical table derived from one or more base tables, table procedures or views. You can use a view to restrict a user's access to specific columns and rows of the underlying tables.

The benefit of views from a security perspective is that you can give users access to the view without giving them equivalent privileges on the underlying tables.

Checking of view privileges is performed at runtime when either of these cases occurs:

- A dynamic SQL statement is executed (the result of the check is cached until end of transaction).
- A new copy of an access module is physically loaded (the result of the check is cached until another copy is loaded or the system is recycled).

In either case, the actual checks made are identical except for the authorization ID used for checking access to the view:

- The executing user, when statements are executed dynamically.
- The owner of the access module, when statements are precompiled.

View ownership

The method used by CA IDMS internal security to secure views employs information about view ownership. A table-like object, such as a view, is owned by the owner of the schema with which it is associated.

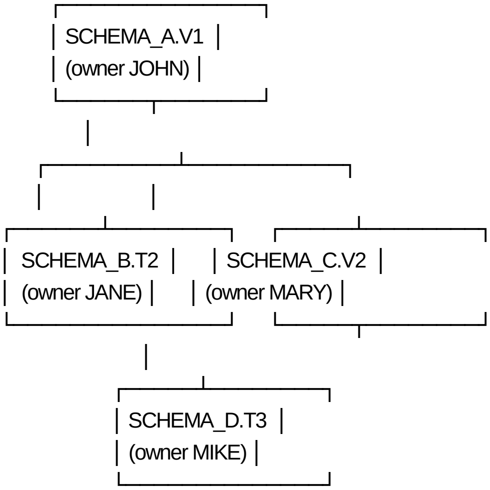
When CA IDMS internal security is in effect for views, a user gains access to the view only if:

- The user holds SELECT privilege on the view.
- The owner of the view holds grantable SELECT privilege on each table-like object referenced by the owner's view.

This approach to view security complies with ANSI standards for SQL.

View example

In the view depicted by the following chart, SCHEMA_A.V1 is owned by user JOHN and references table SCHEMA_B.T2 owned by JANE and view SCHEMA_C.V2 owned by MARY. View V2 in turn references table SCHEMA_D.T3 owned by MIKE.



What CA IDMS Internal Security Checks

Using the preceding view example, assume that user FRED is issuing a dynamic SELECT against view V1. The security system will check to see the following:

- FRED, the executing user, holds SELECT privilege on view V1
- JOHN, the owner of view V1 holds:
  - Grantable SELECT privilege on table T2
  - Grantable SELECT privilege on view V2
- MARY, the owner of view V2 holds grantable SELECT privilege on table T3

The security system uses the schema name qualifier of the table-like object to determine the authorization ID to be checked. The authorization ID of the schema owner is stored in a row of the SYSTEM RESOURCEGROUP table. For example, to determine security requirements on view SCHEMA_A.V1, CA IDMS in effect asks the security system to check whether the owner of SCHEMA_A has grantable SELECT privilege on table T2 and view V2.

External Security Enforcement for Views

If external security is in effect, only the executing user's privilege to access the view is checked. Neither owner privileges nor authorities to access base tables are checked.

Securing SQL routines

SQL routines are SQL-invoked procedures or functions written in the SQL procedural language, that require a number of different objects to be built, processed and executed in the dictionary. The following table gives an overview of these objects showing the dictionary area containing the object, the dictionary entity, the entity subtype, and the name and description. The objects are created while processing the create command for an SQL routine.

Dictionary Area	Entity	Type	Name and Description
DDLCAT DDLCATX	TABLE	PROCEDURE or FUNCTION	A table-like object (procedure or function) contained in the routine schema of the SQL catalog
DDLCAT DDLCATX	TABLE	LOCAL VARIABLES OWNER	One to many table-like objects contained in the routine schema. The names are constructed from ExternalName and nnnn, a serial number starting with 0000 up to n, the number of compound statements in the routine + 1, as follows: SQLLOCnnnnExternalName (1)

Dictionary Area	Entity	Type	Name and Description
DDLDML	MODULE	PROCESS	An IDD process module that contains the CA ADS source code, generated for the SQL routine. The name is constructed from ExternalName, as follows: PREMAP-ExternalName (1)
DDLDCLOD	LOAD MODULE	RCM	The RCM (Resource Control Module) module associated with the SQL routine. The name of the RCM is given by External Name (2)
DDLDCLOD	LOAD MODULE	DIALOG	The dialog module associated with the SQL routine. The name of the dialog is given by ExternalName
DDLCAT DDLCATX	AM	AM	The AM (Access Module) contained in the routine schema and associated with the SQL routine. The name of the AM is given by ExternalName (2)
DDLCATLOD	LOAD MODULE	AM	The AM (Access Module) load module associated with the SQL routine. The name of the AM is given by ExternalName.

(1) The object is not accessed when the SQL routine is invoked.

(2) The object is only accessed when the AM needs to be recompiled.

To invoke an SQL routine a user must have the SELECT privilege on the SQL procedure or function and the EXECUTE privilege on the AM associated with the SQL routine. If CA ADS run time security is implemented then additional privileges might be required to execute the CA ADS dialog associated with the SQL routine.

To create an SQL routine the user must have the CREATE and SELECT privileges on the SQL procedure or function and the CREATE, ALTER and DROP privileges on the AM associated with the SQL routine.

To drop an SQL routine the user must have the DROP privileges on the SQL procedure or function and the DROP privilege on the AM associated with the SQL routine.

Note: Dropping an SQL routine will remove all the associated dictionary objects. The optionally granted privileges will also have been removed, except for the privileges granted on the access module, which are preserved. A REVOKE command can be used to remove the privileges if needed.

How to Grant Privileges to Invoke an SQL Procedure

Assume the SQL procedure GET_EMPLOYEE contained in the SQL schema HR_APPL with an external name GETEMPL. needs to be called by users of the group HR_ DEP.

```
grant select on HR_APPL.GET_EMPLOYEE to HR_DEP;
grant execute on access module HR_APPL.GETEMPL to HR_DEP;
```

How to Grant Privileges to Define and Maintain an SQL Procedure

Assume the SQL procedure GET_EMPLOYEE contained in the SQL schema HR_APPL with an external name GETEMPL. needs to be defined and maintained by users of the group HR_DEV.

```
grant define on HR_DEP.GET_EMPLOYEE to HR_DEV;
grant select on HR_DEP.GET_EMPLOYEE to HR_DEV;
grant define on access module HR_DEP.GETEMPL to HR_DEV;
grant execute on access module HR_APPL.GETEMPL to HR_DEV;
```

Securing SQL Access to Databases

About SQL Access

Users have the capability of accessing both SQL-defined and non-SQL-defined databases with SQL DML. Granting SQL access to a secured non-SQL-defined database is the same as granting access to a secured SQL-defined database with the additional step of granting USE privilege on non-SQL-defined schemas.

Securing SQL schemas

About SQL Schemas

When you secure resource type DB, you control who can create an SQL schema. Until you secure resource type DB, any user can create an SQL schema.

How to Secure SQL Schemas

You secure SQL schemas by securing the DB resource.

Note: For more information, see Securing Databases (see page 123).

If you secure SQL schemas externally, you must also include an entry in the SRTT with external security information for resource type QSCH.

How to Grant Definition Privileges on an SQL Schema

To allow a user to create an SQL schema, you issue a GRANT statement on the SQL schema resource type, specifying the privilege or privileges and identifying the SQL schema. You can specify any combination of CREATE, ALTER, DROP, and DISPLAY privileges, or you can specify all definition privileges (DEFINE).

As a holder of SYSADMIN or DBADMIN privilege, or as owner of the schema, you can specify WITH GRANT OPTION when you grant definition privileges to allow the recipient to grant the same privileges to another user.

More Information:

- For more information, see GRANT SQL Definition Privileges (see page 311).
- For more information, see REVOKE SQL Definition Privileges (see page 330).
- For more information about creating SQL schemas and transferring schema ownership, see the *CA IDMS SQL Reference Guide.*

Securing Non-SQL-defined Schemas

About Non-SQL-Defined Schemas

When you secure resource type DB, you control who can create an SQL schema for a non-SQL-defined schema. Until you secure resource type DB, any user can create an SQL schema for a non-SQL-defined schema.

How to Secure Non-SQL-defined Schemas

You secure non-SQL-defined schemas by securing the DB resource.

Note: For more information, see Securing Databases (see page 123).

If you secure non-SQL-defined schemas externally, you must also include an entry in the SRTT with external security information for resource type NSCH.

How to Grant USE Privilege on a Non-SQL-defined Schema

To allow a user to specify a non-SQL-defined schema when creating an SQL schema, you issue a GRANT statement on the non-SQL-defined schema specifying the USE privilege.

As a holder of SYSADMIN or DBADMIN privilege, you can specify WITH GRANT OPTION when you grant this privilege to allow the recipient to grant the same privilege to another user.

More Information:

- For more information, see GRANT Non-SQL Definition Privilege (see page 303).
- For more information, see REVOKE Non-SQL Definition Privilege (see page 325).
- For more information about creating SQL schemas, see the *CA IDMS SQL Reference Guide*.

Granting SQL Access to Non-SQL-defined Databases

You allow SQL access to non-SQL-defined databases by creating an SQL schema for a non-SQL schema. This requires the CREATE privilege on the SQL schema and the USE privilege on the non-SQL-defined schema, as in this example:

```
grant create on
 schema qschtest
 to dba;

grant use on
 nonsql schema v0001.nschtest
 to dba;
```

The owner of the SQL schema for non-SQL schema has the authority to access the non-SQL-defined database using SQL. Other users require table access privileges.

Securing Tables

About tables

When you secure resource type DB, you can control who can create and access a table-like object. Until you secure resource type DB, any user can create and access a table-like object.

How to Secure Tables

You secure tables by securing the DB resource.

Note: For more information, see Securing Databases (see page 123).

If you secure tables externally, you must also include an entry in the SRTT with external security information for resource type TABL.

How to Grant Table Definition Privileges

To allow a user to create a table-like object, you issue a GRANT statement on the table-like object, specifying the privilege or privileges and identifying the object. You can specify any combination of CREATE, ALTER, DISPLAY, and DROP privileges, or you can specify all definition privileges (DEFINE). You can also specify the REFERENCES privilege.

Note: REFERENCES privilege allows the user to create a constraint that names the table as the referenced table in the constraint.

As a holder of SYSADMIN or DBADMIN privilege or as owner of the table-like object, you can specify WITH GRANT OPTION when you grant definition privileges to allow the recipient to grant the same privileges to another user.

Note: For more information, see GRANT SQL Definition Privileges (see page 311) and see REVOKE SQL Definition Privileges"

For more information about creating tables and constraints, see the *CA IDMS SQL Reference Guide*.

How to Grant Table Access Privilege

To allow a user to access a table-like object, you issue a GRANT statement on the table-like object, specifying the privilege or privileges. You can specify any combination of DELETE, INSERT, SELECT, and UPDATE privileges.

As a holder of SYSADMIN or DBADMIN privilege, or as owner of the table-like object, you can specify WITH GRANT OPTION when you grant access privileges to allow the recipient to grant the same privileges to another user.

Note: For more information, see the following sections:

- GRANT Table Access Privileges (see page 315)
- REVOKE Table Access Privileges (see page 333)

How to Grant All Table Privileges

You can grant all definition and access privileges on a table-like object with the GRANT ALL PRIVILEGES statement.

As a holder of SYSADMIN or DBADMIN privilege, or as owner of the table-like object, you can specify WITH GRANT OPTION when you grant all table privileges to allow the recipient to grant the same privileges to another user.

Note: For more information, see the following sections:

- GRANT All Table Privileges (see page 299)
- REVOKE All Table Privileges (see page 321)

Securing Access to Table Definitions

You can allow limited access to table definitions by granting users privilege on SYSCA views.

SYSCA views restrict access to information in the SYSTEM tables to viewing definitions of only those tables on which the executing user holds SELECT privilege. SYSCA information about tables on which the executing user does not hold SELECT privilege. Therefore, a user who holds privilege on SYSCA views and not on SYSTEM tables must have the authority to retrieve data from a table in order to be able to view the definition of the table.

Note: For more information about SYSCA views, see the *CA IDMS SQL Reference Guide*.

Securing Access Modules

About Access Modules

An access module is a set of compiled and optimized SQL statements. Certain characteristics of the access module are contained in its dictionary definition. The actual module to be loaded at runtime may be regenerated because of changed database characteristics and aspects of the access module definition.

When you secure resource type DB, you can control who can create and maintain an access module definition. Until you secure resource type DB, any user can create and maintain an access module definition.

How to Secure Access Modules

You secure access modules by securing the DB resource.

Note: For more information, see Securing Databases (see page 123).

If you secure access modules externally, you must also include an entry in the SRTT with external security information for resource types DACC (access module definition) and SACC (loadable entity).

How to Grant Access Module Definition Privilege

To allow a user to create and maintain an access module definition, you issue a GRANT statement on the access module resource type, specifying the privilege or privileges and identifying the access module. You can specify any combination of CREATE, ALTER, DROP, and DISPLAY privileges, or you can specify all definition privileges (DEFINE).

As a holder of SYSADMIN or DBADMIN privilege, or as owner of the access module, you can specify WITH GRANT OPTION when you grant definition privileges to allow the recipient to grant the same privileges to another user.

More Information:

- For more information, see GRANT SQL Definition Privileges (see page 311).
- For more information, see REVOKE SQL Definition Privileges (see page 330).
- For more information about creating access modules, see the *CA IDMS SQL Reference Guide*.

How to Grant Access Module Execution Privilege

Two security checks are involved in granting access module execution privilege. The categorization of access modules controls who can load an access module, it does not control who can execute the SQL statements in the access module. You have to think of an access module as any other load module. You need both the authority to load the module and the authority to perform the database access to the program.

As mentioned before, there are two levels of authority that must be granted to implement access module and DB security:

- A Category must be created and an access module added to the Category and execute authority granted. This controls the loading of the access module.
- Execute must be granted on the access module. This controls the database access for statements within the access module.

To allow a user to execute an individual access module, you issue a GRANT statement on the access module resource type, specifying the EXECUTE privilege and identifying the access module.

As holder of SYSADMIN or DBADMIN privilege, or as owner of the access module, you can specify WITH GRANT OPTION when you grant execution privilege to allow the recipient to grant the same privilege to another user.

Note: For more information, see the following sections:

- GRANT Access Module Execution Privilege (see page 295)
- REVOKE Access Module Execution Privilege (see page 318)

Securing the Dictionaries and the User Catalog

Activating Database Security

The dictionaries and the user catalog are CA IDMS databases. To secure these entities, you must activate database security for them.

The following discussion explains what to do when you secure dictionaries by defining occurrence overrides.

Note: You can secure all dictionaries by specifying security for all databases in an SRTT entry for the DB resource type.

For the purposes of discussion, the authorities that you give to allow access to a dictionary are the CA IDMS privileges you would grant if the dictionary is secured internally.

Securing the System Dictionary

To secure the system dictionary using occurrence overrides, you must secure the DB resource type for database name 'SYSTEM' and the names of the three segments that comprise the system dictionary.

In the following example, the first entry secures the name 'SYSTEM' which prevents access to the system dictionary through the database name defined for it at installation. This entry also prevents access to the SYSTEM segment. The next entries secure the SYSMSG segment, which contains messages, and the CATSYS segment, which is the catalog component of the dictionary.

```
#SECRTT TYPE=OCCURRENCE,              X
   RESTYPE=DB,                     X
   RESNAME='SYSTEM',                  X
   SECBY=INTERNAL

#SECRTT TYPE=OCCURRENCE,              X
   RESTYPE=DB,                     X
   RESNAME='SYSMSG',                  X
   SECBY=INTERNAL

#SECRTT TYPE=OCCURRENCE,              X
   RESTYPE=DB,                     X
   RESNAME='CATSYS',                  X
   SECBY=INTERNAL
```

Securing the User catalog

To secure the user catalog with an occurrence override, specify the SYSUSER segment, as in this example:

```
#SECRTT TYPE=OCCURRENCE,                X
   RESNAME='SYSUSER',                   X
   RESTYPE=DB,                          X
   SECBY=INTERNAL
```

If a database name has been defined for this segment, you must also include an entry specifying the database name.

Securing Application Dictionaries

If you activate database security with occurrence overrides, you must individually secure every segment in the application dictionary and every database name that includes a dictionary segment.

Privileges for Secured Dictionaries

After you have secured dictionaries and the user catalog, you grant privileges that permit appropriate access.

If security for the dictionary databases is *internal*, you grant CA IDMS privileges on the database resource types associated with the dictionary, including privileges on resources such as run units and areas that allow users to access the dictionary according to their needs.

If security for the dictionary databases is *external*, you define rules for each dictionary database and each of its associated database resources in the external system. You add SRTT entries with external class and resource name information for the dictionary database resources to be sent with security checks to the external security system.

For example, users who must execute the CA IDMS compilers such as the schema compiler require execute privilege on a Category containing compiler run unit resources. Users who must execute CA IDMS utilities require the appropriate privileges for area access.

Granting Privileges on Run Units

To grant blanket run unit access to an internally secured system dictionary, you first categorize all run units and then grant privilege on the category, as in this example:

```
create resource Category sysdict_general
  add rununit sysdict.* ;

grant execute
  on Category sysdict_general
  to general ;
```

To categorize specific run units for CA IDMS compilers and tools that access the dictionary, you can specify as appropriate run units listed in the installation source library member DLODSECR.

For the purpose of using the CA IDMS Command Facility, there is no need to grant privileges on run units that access the SYSUSER segment.

Example

In this example, the system and application dictionaries have been secured. The first statement creates a Category of run units that access these dictionaries, and the second statement grants EXECUTE privilege on the Category:

```
create resource Category rununit_category
    add rununit appldict.idmsnwka.idmschem
    add rununit appldict.idmsnwka.idmsdddl
    add rununit appldict.idmsnwka.idmsubsc
    add rununit appldict.idmsnwkg.idmsrpts
    add rununit system.idmsnwka.idmsdddl
    add rununit system.idmsnwka.rhdcsgen
    add rununit system.idmsnwkg.idmsrpts
    ;

grant execute on Category rununit_category
    to rununit_group
    ;
```

Granting Privileges on Areas

To allow execution of certain CA IDMS utilities against a secured database, you grant DBAREAD or DBAWRITE privilege on the area or areas to be accessed. The following table presents the installation names of the areas of the system dictionary and the user catalog:

Database	Area (segment-name.area-name)
System dictionary	SYSTEM.DDLDML SYSTEM.DDLDCRUN SYSTEM.DDLDCLOG SYSTEM.DDLDCSCR SYSTEM.DDLDCLOD SYSMSG.DDLDCMSG CATSYS.DDLCAT CATSYS.DDLCATX CATSYS.DDLCATLOD
User catalog	SYSUSER.DDLSEC

Granting Privileges on Non-SQL-defined Schemas

When you secure the dictionaries and the user catalog, you control SQL access to these databases. You allow SQL access by creating SQL schemas for the non-SQL-defined schemas that describe these databases (IDMSNTWK for the dictionary and IDMSSECU for the user catalog) and granting table access privileges.

Similarly, security definitions for system and non-SQL-defined database resources are inaccessible through SQL unless you create an SQL schema for IDMSSECS, the non-SQL-defined schema for system resources security database.

Granting Access to SYSTEM Tables

When you secure a dictionary as a database, you secure tables associated with the SYSTEM schema in the catalog component of the dictionary. To allow access to the SYSTEM tables, you have these options:

- Grant access privileges on SYSTEM tables, individually or collectively.
- Grant access privileges on SYSCA views, which allow the executing user to view data in SYSTEM tables about only those tables on which the user holds SELECT privilege.

Note: For more information about SYSTEM tables and SYSCA views, see the *CA IDMS SQL Reference Guide*.

Chapter 10: Securing Application Dictionary Resources

This section contains the following topics:

What Is an Application Dictionary?

An application dictionary contains definitions for application development objects such as dialogs and maps. An application dictionary may also contain non-SQL defined schemas and subschemas and, in its catalog component, SQL-defined entities.

Securing the Dictionary As a Database

Why You Do It

An application dictionary consists of one segment that is a non-SQL defined database and, if the SQL Option is installed, another segment (the catalog component) that is an SQL-defined database. A user can access the dictionary as a database, and, therefore, to secure the application dictionary, you should secure it as a database in CA IDMS centralized security.

How You Do It

To secure the application dictionary as a database, you secure the DB resource type or the occurrence of the DB resource that the dictionary represents. Then you categorize run units for the compilers and tools that access the dictionary and grant execution privilege on the Category.

Note: For more information, see Securing the Dictionaries and the User catalog (see page 149).

Signon To the Dictionary

Using a Compiler or Tool

When a user invokes a compiler or tool, signon to the application dictionary is automatically initiated using the ID with which the requesting user is signed on to the system.

If the user of the compiler or tool is not signed on to the system, an actual system signon is attempted internally, and if it is successful, dictionary signon proceeds.

Thus, under centralized security, a user who is authorized to sign on to the system is authorized to access the dictionary that is current for the user session.

Current dictionary: You can enforce the specification of the current dictionary for a user's session by including the DICTNAME attribute in the user profile with the OVERRIDE=NO parameter. This prevents the user from accessing a dictionary other than the one you specify in the DICTNAME attribute. For more information, see Securing User Profiles (see page 90).

Securing Secondary Signons

You can secure signon to a particular application dictionary by using DDDL to specify SECURITY FOR IDD SIGNON IS ON for the dictionary. In this situation, the user must be defined in the application dictionary with the ADD USER statement and authorized to sign on to the dictionary with the inclusion of IDD SIGNON IS ALLOWED in the USER statement.

This measure provides additional security only for IDD and does not affect security for other compilers that access the dictionary. Therefore, it is not a substitute for securing signon through centralized security.

Secondary Signon Processing

If a user issues an IDD signon statement that specifies the same ID as the user's system signon ID, no password validation is done for signon to the dictionary. If a user issues an IDD signon statement that specifies a different ID from the user's system signon ID, then the ID and password entered on the signon statement must match an ID and password defined in the dictionary with the ADD USER statement.

If the user is either not defined or not authorized, the secondary signon is rejected.

Compiler Security Within the Dictionary

What is Compiler Security?

The compiler security described in this section is part of the IDD architecture and is not part of CA IDMS centralized security. Security checks access the definition of the user in the dictionary, not the user catalog.

When do Compilers Check Security?

The compilers perform security checking operations when any of the following is true of a DDL statement:

- The verb is SIGNON, VALIDATE, or GENERATE.
- The SET OPTIONS statement contains REGISTRATION OVERRIDE.
- The component type is SCHEMA.
- The component type is SUBSCHEMA.
- The statement is the first statement of the session.

Checks User's Dictionary Description

In any of the preceding cases, the compiler determines whether the requested operation is secured within the dictionary.

If the operation is not secured, the compiler bypasses the security check and begins processing the statement.

If the operation is secured, the compiler checks the user's description in the dictionary to determine whether the user is authorized to perform an operation.

If the user is authorized, the compiler processes the input statement; if not, the compiler issues an error message. All levels of security checking follow this procedure.

Types of Security

The compilers check four kinds of security:

- *Compiler* security
- *Registration override* security
- *Verb* security
- *Component* security

What Follows

Each kind of security is presented separately as follows; each topic includes the following information:

- When security is checked
- How security is turned on or off
- How the compiler determines who the issuing user is
- What constitutes an authorized user

Checking Compiler Security

Turning on Compiler Security

Compiler security is turned on or off through the IDD DDDL statement, SET OPTIONS FOR DICTIONARY SECURITY FOR IDMS IS ON/OFF. (Note that this IDD DDDL statement also turns verb security on or off: compiler security and verb security cannot be set independently.)

How the Compiler Checks the User

To determine who is issuing the statement, the compiler looks at the user name specified in the SIGNON statement. If the SIGNON statement is not issued or does not include the USER clause, the user name defaults to the following:

- The system signon ID of the user
- The ID in SET OPTIONS DEFAULT PREPARED BY/REVISED BY *user-name*

Definition of an Authorized User

An authorized user, for this function, is one whose description in the dictionary includes authority to use the compiler. Compiler authority is assigned through one of the following IDD DDDL USER statements (use MODIFY for existing user descriptions):

```
ADD USER NAME IS user-ID
  AUTHORITY FOR any verb    assigns authority to use both
    IS ALL.            compilers

ADD USER NAME IS user-ID
  AUTHORITY FOR any verb    assigns authority to use both
    IS IDMS.            compilers
```

```
ADD USER NAME IS user-ID
   AUTHORITY FOR any verb     assigns authority to use the
      IS SCHEMA.              schema compiler only

ADD USER NAME IS user-ID
   AUTHORITY FOR any verb     assigns authority to use the
      IS SUBSCHEMA.           subschema compiler only
```

Checking Registration Override Security

When Compilers Check Registration Override

The schema and subschema compilers check registration override security when they encounter a SET OPTIONS statement containing a REGISTRATION OVERRIDE clause.

Turned on by REGISTRATION OVERRIDE

Unlike the other kinds of security, this one cannot be turned on or off; that is, the compiler always checks for an authorized user when it encounters a REGISTRATION OVERRIDE clause.

How the Compilers Check the User

To determine who is issuing the REGISTRATION OVERRIDE clause, the compiler looks at user names specified in the **user-specification** clause of the SET OPTIONS statement. If the SET OPTIONS statement does not include this clause, the user name defaults as described in the SET OPTIONS statement.

Note: For more information about the SET OPTIONS statement, see the *CA IDMS Database Administration Guide*.

Description of an Authorized User

An authorized user for the REGISTRATION OVERRIDE clause is one who has been defined in the dictionary and whose description includes all authorities. All authorities are assigned through the following IDD DDDL USER statement (use MODIFY for existing user descriptions):

```
ADD USER NAME IS user-id
   AUTHORITY IS ALL.
```

Checking Verb Security

When Compilers Check Verb Security

The schema and subschema compilers check verb security whenever a SCHEMA statement (schema compiler only) or SUBSCHEMA statement (subschema compiler only) is issued. Note that verb security is not checked for each component of a schema or subschema. Once a user passes security for a schema or subschema, all of its components are available to the user.

Turning on Verb Security

Verb security is turned on or off through the IDD DDDL statement, SET OPTIONS FOR DICTIONARY SECURITY FOR IDMS IS ON/OFF. (Note that this IDD DDDL statement also turns compiler security on or off: verb security and compiler security cannot be set independently.)

How the Compilers Check the User

To determine who is issuing the SCHEMA or SUBSCHEMA statement, the compiler looks at the following:

- The ID of user signed on to the dictionary
- The **user-specification** on the SCHEMA or SUBSCHEMA statement
- The **user-specification** on the SET OPTIONS statement

If any of these IDs is that of an authorized user, security is satisfied and the compiler processes the request.

Description of an Authorized User

An authorized user, for this function, is one who is defined in dictionary and whose description includes authority to issue the verb specified in the SCHEMA or SUBSCHEMA statement, *in conjunction with* the authority to use the compiler. Verb authority is assigned through IDD DDDL USER statements, such as those in the following examples:

```
ADD USER NAME IS KCO          assigns authority to use all
  AUTHORITY FOR UPDATE          verbs in each DDL compiler
    IS IDMS.

ADD USER NAME IS GKD          assigns authority to use MODIFY,
  AUTHORITY FOR MODIFY          DISPLAY, and PUNCH in each
    IS IDMS.              DDL compiler

ADD USER NAME IS TWG          assigns authority to use DELETE,
  AUTHORITY FOR DELETE          DISPLAY, and PUNCH in the
    IS SCHEMA.              schema compiler only
```

Implicit Subschema Updates Allowed

While schema authority only allows the user to access the schema compiler, any subschema updates resulting from authorized schema updates are allowed (for example, deleting a set from the schema causes the set to be deleted from the schema's subschemas).

Note: For more information about assigning verb authority, see the *CA IDMS IDD DDDL Reference Guide*.

Checking Component Security

When Compilers Check Component Security

The schema compiler checks the security of a specific schema whenever a SCHEMA statement (other than ADD SCHEMA) is issued for that schema; the subschema compiler checks security of a specific subschema whenever a SUBSCHEMA statement (other than ADD SUBSCHEMA) is issued for that subschema. Note that this security is not checked for each component of a schema or subschema; once a user passes security for a schema or a subschema, all of its components are available to the user. Component security applies to every existing schema and subschema, regardless of whether compiler security is on.

Security Maintained Through PUBLIC ACCESS Clause

Security for a specific schema or subschema is set through the PUBLIC ACCESS clause of the SCHEMA or SUBSCHEMA statement. A schema or subschema is said to be unsecured if PUBLIC ACCESS IS ALLOWED FOR ALL is in effect; any other public access specification places some level of security on the schema or subschema. The following examples show how component security is set:

```
MOD SCHEMA EMPSCHM          turns off security for EMPSCHM
  PUBLIC ACCESS IS ALLOWED
    FOR ALL.

MOD SUBSCHEMA EMPSS01        turns on security for all verbs
  OF SCHEMA EMPSCHM         issued against EMPSS01
  USER IS NET
     REGISTERED FOR ALL
  PUBLIC ACCESS IS ALLOWED
    FOR NONE.

MOD SUBSCHEMA EMPSS02        turns off security for DISPLAY
  OF SCHEMA EMPSCHM         EMPSS02 and PUNCH EMPSS02;
  USER IS NET          turns on security for all other
     REGISTERED FOR ALL     verbs issued against EMPSS02
  PUBLIC ACCESS IS ALLOWED
    FOR DISPLAY.
```

Description of an Authorized User

An authorized user for a specific schema or subschema is one who is defined in the dictionary and whose association with the schema or subschema includes the verb used in the SCHEMA or SUBSCHEMA statement being processed. This authority is assigned through the REGISTERED FOR subclause (in the **user-options-specification**) of the USER clause in a previously issued SCHEMA or SUBSCHEMA statement, as illustrated in the following examples:

```
ADD SUBSCHEMA NAME IS EMPSS01     assigns authority to KCO to
  USER NAME IS KCO                use all verbs against EMPSS01
    REGISTERED FOR ALL.

ADD SUBSCHEMA NAME IS EMPSS02     assigns authority to GKD to
  USER NAME IS GKD                access EMPSS02 with only
    REGISTERED FOR PUBLIC ACCESS.  those verbs specified in
                                  EMPSS02's PUBLIC ACCESS clause

ADD SCHEMA NAME IS EMPSCHM        assigns authority to TWG to
  USER NAME IS TWG                DISPLAY and PUNCH EMPSCHM
    REGISTERED FOR DISPLAY.
```

Note: For more information about PUBLIC ACCESS and USER clauses, see the SCHEMA and SUBSCHEMA statement documentation in the *CA IDMS Database Administration Guide*.

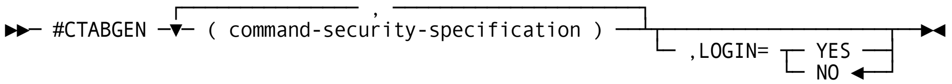
Chapter 11: Syntax for Assembler Macros

This section contains the following topics:

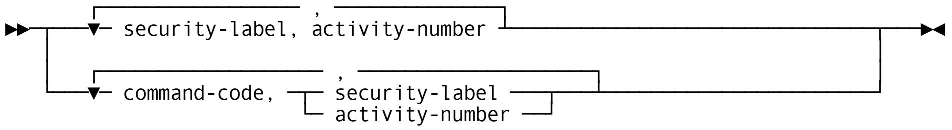
#CTABGEN

Assigns activity numbers to DCMT commands.

#CTABGEN Syntax



Expansion of Command-Security-Specification



#CTABGEN Parameters

command-security-specification

Specifies the information for assigning activity numbers to DCMT commands.

Expanded syntax for *command-security-specification* appears immediately following the command syntax.

LOGIN

Specifies whether DC/UCF writes input DCMT commands to the log file.

If you do not specify the LOGIN parameter, the effect is the same as specifying LOGIN=NO.

YES

Specifies that all input DCMT commands are written to the log file.

NO

Specifies that most input DCMT commands are not written to the log.

Some input DCMT commands are *always* written to the log because they provide important information about the activity of the command. For example, DCMT VARY MEMORY commands are always logged. The LOGIN option *does not* stop the system from logging these input DCMT commands.

security-label,activity-number

Associates a DCMT security label with an activity number.

Security-label specifies a label with which you can associate one or more DCMT commands.

Security-label must be one alphabetic character (A through Z). You can define at most 26 security labels in the #CTABGEN macro.

Activity-number specifies the activity number you are associating with *security-label*. *Activity-number* must be a numeric value in the range 1 through 256.

command-code

Specifies one of the following:

- The **4-character** command code that identifies a predefined group of DCMT commands. For example, N001 identifies the DCMT SHUTDOWN command group.
- The **7-character** command code that identifies a specific DCMT command option. For example, N001001 identifies the DCMT SHUTDOWN IMMEDIATE command.

You can specify any number of DCMT command codes in the #CTABGEN macro.

Note: For a list of valid DCMT command codes, see DCMT Command Codes (see page 165).

security-label

Specifies the DCMT security label that you are associating with *command-code*.

activity-number

Specifies the DCMT activity number that you are associating with *command-code*.

#CTABGEN Usage

Coding Considerations

All lines except the first one must start in column 16.

All lines except the last one must have a non-blank character in column 72.

General

When you use the #CTABGEN macro, you can assign a DCMT activity number to a DCMT command or to a group of commands:

- You can associate a specific **DCMT activity number** (1 through 256) with the DCMT command.
- You can associate a site-defined **security label** (A through Z) with the DCMT command.

  Use of security labels makes it easier to maintain security definitions when several commands and/or command groups are assigned the same DCMT activity number. You define a security label in the #CTABGEN macro itself. You need only change the security label definition in the #CTABGEN macro to modify the security for all associated DCMT commands.

Defining Labels

You make security label assignments in a parenthesized list. Use a comma to separate specifications and parenthesized lists. For example:

```
----+----1----+----2----+----3----+----4----+----5----+----6----+----7--

    #CTABGEN (A,3,B,10),                                              X
               (N001,A,N002,B,N003,6)
```

You must define all security labels before beginning the list of DCMT security assignments.

Specifying Command Groups and Command Codes

If a 4-character command code is given for a group of commands and a 7-character command code is given for a particular command within that group, then the DCMT activity number for the particular command overrides the security for the group.

No Effect of LOGIN on Response Messages

Response messages issued for DCMT commands are not affected by the LOGIN specification. Response messages are written to the system log if LOG is given as a destination in the message definition.

z/VSE Sites

At z/VSE sites, the LOGIN parameter, if coded, must be coded following all other #CTABGEN parameters.

Generating the #CTABGEN Macro

The source file that contains the #CTABGEN macro can contain only one macro. The resulting object must be link edited as a stand-alone module IDMSCTAB.

To assemble and link edit the #CTABGEN macro, use the JCL or commands appropriate for your operating system.

Note: For more information, see Appendix A: Security Macro JCL. (see page 379)

Example 1

Assign DCMT activity number 3 to DCMT SHUTDOWN (code N001) and ABORT (code N002) by means of security label A:

```
----+----1----+----2----+----3----+----4----+----5----+----6----+----7--

      #CTABGEN (A,3),                                         X
          (N001,A,N002,A)
```

Example 2

Assign different discrete security to several DCMT commands:

```
----+----1----+----2----+----3----+----4----+----5----+----6----+----7--

      #CTABGEN (A,3,B,10),                                       X
          (N001,A,N002,A,N033,A),                                  X
          (N011,B,N025,B),                                     X
          (N025007,50)
```

This #CTABGEN macro associates three different DCMT activity numbers (3, 10, and 50) with DCMT commands:

- **Number 3 is assigned indirectly**, by means of security label A, to the following:
  - SHUTDOWN (N001)
  - ABORT (N002)
  - VARY MEMORY (N033)
- **Number 10 is assigned indirectly**, by means of security label B, to the following:
  - VARY DATABASE PROGRAM (N011)
  - VARY PROGRAM (N025)
- **Number 50 is assigned directly** to one DCMT command: DCMT VARY PROGRAM STORAGE PROTECT ON (N025007). To use this DCMT command, a user would require DCMT activity number 50. To use any other DCMT VARY PROGRAM command, the user would require DCMT activity number 10.

#CTABGEN DCMT Command Codes

Code	DCMT Command
N001	SHUTDOWN
N001000	SHUTDOWN
N001001	SHUTDOWN IMMEDIATE
N002	ABORT
N002000	ABORT
N002001	ABORT DUMP
N003	DISPLAY DATABASE
N003000	DISPLAY DATABASE
N004	DISPLAY TRANSACTIONS
N004000	DISPLAY TRANSACTIONS
N004001	DISPLAY TRANSACTION *run-unit-id*
N005	DISPLAY AREAS
N005000	DISPLAY AREAS
N005001	DISPLAY AREA *area-name*
N005002	DISPLAY AREA *area-name* FILE
N005003	DISPLAY AREA *area-name* BUFFER
N005004	DISPLAY AREA *area-name* ALL
N005128	DISPLAY AREA *area-name* LOC

Code	DCMT Command
N006	DISPLAY BUFFERS
N006000	DISPLAY BUFFERS
N006001	DISPLAY BUFFER *buffer-name* AREA
N006002	DISPLAY BUFFER *buffer-name* FILE
N006003	DISPLAY BUFFER *buffer-name*
N006004	DISPLAY BUFFER *buffer-name* ALL
N006128	DISPLAY BUFFER *buffer-name* LOC
N006130	DISPLAY BUFFER *buffer-name* FILE BUFNO
N007	DISPLAY CENTRAL VERSION
N007000	DISPLAY CENTRAL VERSION
N008	DISPLAY DATABASE PROGRAMS
N008000	DISPLAY DATABASE PROGRAMS
N008001	DISPLAY DATABASE PROGRAM *program-name*
N009	VARY AREA
N009000	ACTIVE
N009001	RETRIEVAL *<additional parameters>*
N009002	UPDATE (ONLINE) *<additional parameters>*
N009003	OFFLINE *<additional parameters>*
N009004	TRANSIENT RETRIEVAL *<additional parameters>*
N009005	UPDATE (ONLINE) LOCKED *<additional parameters>*
N009006	PREFETCH ON
N009007	PREFETCH OFF
N009008	QUIESCE WAIT
N009011	OPEN UPDATE
N009012	OPEN
N009013	CLOSE
N009014	PURGE
N009015	QUIESCE <NOWAIT>
N009016	DEALLOCATE
N009017	SHARED CACHE ca*che-name*
N009018	SHARED CACHE NO
N009020	DATA SHARING ON
N009021	DATA SHARING OFF
N009022	ALLOCATE

Code	DCMT Command
N010	VARY BUFFER
N010000	PAGES
N010001	OPEN
N010002	CLOSE
N010003	MAXIMUM PAGES
N010004	INITIAL PAGES
N010005	ADDITIONAL PAGES
N010006	OPSYS
N010007	DC
N010010	PREFETCH ON
N010011	PREFETCH OFF
N010012	PREFETCH *nnn*
N011	VARY DATABASE PROGRAM
N011000	PRIORITY
N011001	LOCKS
N011002	ONLINE
N011003	OFFLINE
N011004	STORAGE/LOCK/CALL/DBIO LIMIT
N012	VARY CENTRAL VERSION
N012000	OFFLINE
N012001	ONLINE
N013	VARY TRANSACTION
N013000	ABORT
N013001	PRIORITY
N015	DISPLAY TASKS
N015001	DISPLAY TASKS
N015002	DISPLAY TASK *task-name*
N016	DISPLAY PROGRAMS
N016001	DISPLAY PROGRAMS
N016002	DISPLAY PROGRAM *program-name*
N017	DISPLAY TIME
N017001	DISPLAY TIME
N017002	DISPLAY TIME TASKS
N018	DISPLAY QUEUES
N018001	DISPLAY QUEUES
N018002	DISPLAY QUEUE *queue-name*

Code	DCMT Command
N019	DISPLAY DESTINATIONS
N019001	DISPLAY DESTINATIONS
N019002	DISPLAY DESTINATION *destination-id*
N020	DISPLAY LTERMINALS
N020001	DISPLAY LTERMINALS
N020002	DISPLAY LTERMINAL *logical-terminal-id*
N020003	DISPLAY LTERMINAL RESOURCES
N021	DISPLAY PTERMINALS/SNA PTERMINALS
N021001	DISPLAY PTERMINALS/SNA PTERMINALS
N021002	DISPLAY PTERMINAL *physical-terminal-id*
	DISPLAY SNA PTERMINAL *physical-terminal-id*
N021004	DISPLAY UCF FETID

Code	DCMT Command
N022	DISPLAY MEMORY
N022001	*hex-address*
N022002	CSA
N022003	TCA
N022004	RCA
N022005	SCT
N022006	OPT
N022007	MAP
N022008	TDT
N022009	TDE
N022010	PDT
N022011	PDE
N022012	QDT
N022013	QDE
N022014	DDT
N022015	DDE
N022016	LTT
N022017	LTE
N022018	PLE
N022019	PTE
N022020	ID
N022021	PROGRAM
N022022	STR
N022023	MOD/EP
N022024	NUCLEUS
N022025	ESE
N022026	ERES
N022027	ACTIVE ERES
N022028	SVC
N022029	CCE
N022030	DMCL
N022031	BCR/BUFFER
N022032	JCB/JOURNAL
N022033	SEGMENT
N022034	DPR/AREA
N022035	FCB/FILE

Code	DCMT Command
N023	DISPLAY ACTIVE
N023001	TASKS
N023002	PROGRAMS
N023010	TASKS WAITING
N023013	STORAGE
N023020	TASKS HOLDING
N023027	REENTRANT PROGRAMS
N023039	XA PROGRAMS
N023040	XA REENTRANT PROGRAMS
N024	VARY TASK
N024001	ENABLE
N024002	DISABLE
N024003	SECURITY
N024004	PRIORITY
N024005	STALL
N024006	PROGRAM
N024007	RESOURCE INTERVAL *timeout-interval*/OFF
N024008	RESOURCE INTERVAL SYSTEM
N024009	RESOURCE PROGRAM
N024010	STALL OFF
N024011	STALL SYSTEM
N024012	SAVE
N024013	NOSAVE
N024014	LOCATION ANY
N024015	LOCATION BELOW
N024016	MAXIMUM CONCURRENT OFF
N024017	MAXIMUM CONCURRENT *task-count*
N024018	STORAGE/LOCK/CALL/DBIO LIMIT
N024019	EXTERNAL WAIT *nnn*
N024020	QUIESE WAIT *nnn*
N024021	TRANSACTION SHARING ON/OFF
N024022	ON COMMIT ..
N024023	ON ROLLBACK ..
N024024	SNAP *snap-options*

Code	DCMT Command
N025	VARY PROGRAM
N025001	ENABLE
N025002	DISABLE
N025003	SECURITY
N025004	NEW COPY
N025005	PROGRAM CHECK THRESHOLD
N025006	DUMP THRESHOLD
N025007	STORAGE PROTECT ON
N025008	STORAGE PROTECT OFF
N025009	NEW COPY QUIESCE
N025010	NEW COPY IMMEDIATE
N025011	ADSO STATISTICS ON
N025012	ADSO STATISTICS OFF
N025013	MULTIPLE ENCLAVE ON
N025014	MULTIPLE ENCLAVE OFF
N025015	DEFINE <keyword>
N025016	DEFINE LANGUAGE
N025017	DEFINE ISA SIZE
N025018	DEFINE TYPE
N025019	DEFINE MPMODE
N025020	SNAP *snap-options*
N026	VARY TIME
N026001	STALL
N026002	RUNAWAY
N026003	TIMER
N026004	RESOURCE INTERVAL
N026005	RESOURCE PROGRAM
N026006	RECOVERY WAIT *nnn*
N026007	QUIESCE WAIT *nnn*
N027	VARY STORAGE POOL
N027001	VARY STORAGE POOL CUSHION
N027002	VARY STORAGE POOL RELOCATABLE THRESHOLD

Code	DCMT Command
N028	VARY LTERM
N028001	DESTINATION ONLINE
N028002	DESTINATION OFFLINE
N028003	ONLINE
N028004	OFFLINE
N028005	TO
N028006	DISCONNECT
N028008	USERTRACE OFF
N028009	USERTRACE ON SAVE
N028010	USERTRACE ON WRAP
N028011	RESOURCES DELETE
N028012	COMMAND
N028013	TERMINAL
N028014	TCP/IP TRACE
N029	VARY PTERM
N029000	DEFAULT PRINT CLASS
N029001	ONLINE
N029002	OFFLINE
N029003	TRACE OFF
N029004	TRACE *hh*
N029005	QUIESCE
N029006	CONNECT
N029007	DISCONNECT
N029008	ONLINE *<telephone-number>*
N029009	CONNECT *<telephone-number>*
N029010	*tcp/ip-parameters*

Code	DCMT Command
N030	VARY LINE
N030001	ONLINE
N030002	OFFLINE
N030003	CONTROL UNIT ONLINE
N030004	CONTROL UNIT OFFLINE
N030005	QUIESCE
N030006	CONNECT
N030007	DISCONNECT
N030008	RLN ONLINE
N030009	RLN OFFLINE
N030010	RLN QUIESCE
N030011	RLN CONNECT
N030012	RLN DISCONNECT
N030013	RLN CONTROL UNIT ONLINE
N030014	RLN CONTROL UNIT OFFLINE
N030015	MASTER
N030016	SLAVE
N031	VARY QUEUE
N031001	ONLINE
N031002	OFFLINE
N031003	THRESHOLD COUNT
N031004	MAX RECORDS
N031005	TASK CODE
N031006	DELETE
N032	VARY DESTINATION
N032001	ONLINE
N032002	OFFLINE
N032003	ADD TERMINAL
N032004	ADD OPERATOR
N032005	DELETE TERMINAL
N032006	DELETE OPERATOR
N033	VARY MEMORY
N033001	X '*hex-literal*'
N033002	C '*character-literal*'
N033003	PROGRAM

Code	DCMT Command
N034	VARY ACTIVE TASK
N034000	TERMINATE USERID *user-id* DUMP
N034001	MAX TASK
N034002	TERMINATE TASKID
N034003	TERMINATE TERMID
N034004	PRIORITY TASKID
N034005	PRIORITY TERMID
N034006	TERMINATE USERID
N034007	STORAGE/LOCK/CALL/DBIO LIMIT
N034008	TERMINATE TASKID *task-id* DUMP
N034009	TERMINATE TERMID *logical-terminal-id* DUMP

Code	DCMT Command
N035	HELP
N035000	HELP
N035001	TASKS
N035002	PROGRAMS
N035003	TIME
N035004	QUEUES
N035005	DESTINATIONS
N035006	TERMINALS
N035007	MEMORY
N035008	STORAGE
N035009	DATABASE
N035010	PRINTERS
N035011	MESSAGE
N035012	REPORTS
N035013	STATISTICS
N035014	LOADLIBS, LOADLISTS, and DICTIONARIES
N035015	SNAP
N035016	ADSO
N035017	LIMITS
N035018	LU
N035019	SNA
N035020	XA
N035021	JOURNALS
N035022	AREAS
N035023	MULTITASK
N035024	NUCLEUS
N035025	RUN UNITS
N035026	BUFFERS
N035027	SHUTDOWN and ABORT
N035028	DYNAMIC
N035029	FILES
N035030	SEGMENTS
N035031	DEADLOCKS
N035032	TRANSACTIONS
N035033	DBTABLE

Code	DCMT Command
N035034	DMCL
N035035	LOCKS
N035036	LOG
N035037	NODE
N035038	SYSGEN
N035039	DDS
N035040	DBGROUP
N035041	SHARED CACHE
N035042	RESOURCE
N035043	DATA SHARING
N035044	ID
N035045	DBNAME
N035046	SYSTRACE
N035047	SCRATCH
N035048	TCP/IP
N035049	CHANGE TRACKING
N035051	AUTOTUNE
N035052	MODID
N036	DISPLAY PRINTERS
N036001	DISPLAY PRINTERS
N036002	DISPLAY PRINTER *printer-id*
N037	DISPLAY CLASSES/REPORTS
N037001	DISPLAY CLASSES/REPORTS
N037002	DISPLAY CLASS/REPORTS CLASS *printer-class*
N037003	DISPLAY REPORTS DESTINATION *printer-destination*
N038	VARY PRINTER
N038001	DRAIN
N038002	REQUEUE
N038003	START
N038004	CANCEL
N038005	CLASSES
N038006	DESTINATION TO
N038007	DESTINATION ONLINE
N038008	DESTINATION OFFLINE

Code	DCMT Command
N039	VARY REPORT
N039001	DELETE
N039002	TO CLASS
N039003	TO DESTINATION
N039004	COPIES
N039005	FIRST
N039006	LAST
N039007	HOLD
N039008	RELEASE
N039009	KEEP
N040	DISPLAY REPLIES
N040000	DISPLAY REPLIES
N041	DISPLAY/VARY/WRITE STATISTICS
N041001	DISPLAY STATISTICS INTERVAL
N041002	VARY STATISTICS INTERVAL *interval*
N041003	VARY STATISTICS INTERVAL OFF
N041004	WRITE STATISTICS
N041005	DISPLAY STATISTICS SYSTEM
N041006	VARY STATISTICS TRANSACTION ON
N041007	VARY STATISTICS TRANSACTION OFF
N041008	DISPLAY STATISTICS ROLL
N041009	VARY STATISTICS ROLL TIME *HH:MM*
N041010	VARY STATISTICS ROLL FRE *ddd*
N041011	VARY STATISTICS ROLL TIME *HH:MM FRE ddd*
N041012	VARY STATISTICS NOROLL
N041013	WRITE STATISTICS ROLL
N042	DISPLAY JOURNAL
N042000	DISPLAY JOURNAL
N042001	DISPLAY JOURNAL *journal-name*
N043	VARY JOURNAL
N043000	VARY JOURNAL
N044	VARY UCF FETID
N044001	ONLINE
N044002	OFFLINE
N044005	QUIESCE

Code	DCMT Command
N045	VARY UCF SYSTEM
N045001	ONLINE
N045002	QUIESCE
N045003	OFFLINE
N045004	NEW COPY
N046	VARY DYNAMIC
N046001	PROGRAM
N046002	TASK
N047	DISPLAY LINES
N047001	DISPLAY LINES
N047002	DISPLAY LINE
N048	DISPLAY/VARY DDS
N048001	DISPLAY DDS LINE
N048002	DISPLAY DDS PTERM
N048003	DISPLAY DDS
N048004	VARY PTERM WEIGHT
N049	DISPLAY/VARY DBTABLE
N049000	DISPLAY DBTABLE
N049001	VARY DBTABLE
N050	VARY LOADLIB
N050000	OFFLINE
N050001	ONLINE
N051	DISPLAY LOADLIBS/LOADLISTS/DICTIONARIES
N051001	DISPLAY LOADLIBS
N051002	DISPLAY LOADLIB
N051003	DISPLAY DICTIONARIES
N051004	DISPLAY DICTIONARY
N051005	DISPLAY LOADLIST
N051006	DISPLAY LOADLISTS
N052	DISPLAY ALL STORAGE POOLS
N052001	DISPLAY ALL STORAGE POOLS

Code	DCMT Command
N053	VARY SNAP
N053000	SYSTEM ON
N053001	SYSTEM OFF
N053002	SYSTEM PHOTO
N053003	SYSTEM NOPHOTO
N053004	TASK ON
N053005	TASK OFF
N053006	TASK PHOTO
N053007	TASK NOPHOTO
N053008	TASK TRACE ON
N053009	TASK TRACE OFF
N053010	TASK TRACE TASK
N053011	TASK TRACE LIMIT nnn
N053012	TASK TRACE LIMIT OFF
N054	DISPLAY SNAP
N054000	DISPLAY SNAP
N055	DISPLAY ALL PROGRAM POOLS
N055001	DISPLAY ALL PROGRAM POOLS
N056	DISPLAY/VARY ADSO STATISTICS
N056001	VARY ADSO STATISTICS ON
N056002	VARY ADSO STATISTICS ON ALL
N056003	VARY ADSO STATISTICS ON SELECTED
N056004	VARY ADSO STATISTICS OFF
N056005	VARY ADSO STATISTICS CHECKPOINT
N056006	DISPLAY ADSO STATISTICS
N056007	VARY ADSO RECORD COMPRESSION ON
N056008	VARY ADSO RECORD COMPRESSION OFF
N058	DISPLAY MESSAGE
N058000	DISPLAY MESSAGE
N059	DISPLAY/VARY LIMITS
N059001	DISPLAY LIMITS
N059002	VARY LIMITS
N060	DISPLAY LU
N060001	DISPLAY LUS
N060002	DISPLAY LU *logical-unit-name*

Code	DCMT Command
N061	VARY LU
N061001	VARY LU (all except VARY LU RESET)
N061002	VARY LU RESET
N062	DISPLAY NUCLEUS MODULE RELOAD TABLE
N062000	DISPLAY NUCLEUS MODULE RELOAD TABLE
N063	VARY NUCLEUS
N063001	VARY NUCLEUS MODULE NEW COPY
N063002	VARY NUCLEUS MODULE CANCEL
N063004	VARY NUCLEUS MODULE RELOAD
N064	DISPLAY/VARY DISTRIBUTED
N064001	DISPLAY DISTRIBUTED TRANSACTION
N064002	DISPLAY DISTRIBUTED TRANSACTION (X)ID
N064006	DISPLAY DISTRIBUTED RESOURCE MANAGER
N064007	DISPLAY DISTRIBUTED RESOURCE MANAGER name
N064010	VARY DISTRIBUTED TRANSACTION (X)ID
N064011	VARY DISTRIBUTED RESOURCE MANAGER name
N065	VARY DB WRITE DRIVER ONLINE
N065000	VARY DB WRITE DRIVER ONLINE
N066	VARY JOURNAL DRIVER ONLINE
N066000	VARY JOURNAL DRIVER ONLINE
N067	VARY DB WRITE DRIVER OFFLINE
N067000	VARY DB WRITE DRIVER OFFLINE
N068	VARY JOURNAL DRIVER OFFLINE
N068000	VARY JOURNAL DRIVER OFFLINE
N069	VARY JOURNAL FRAGMENT NUMBER
N069000	VARY JOURNAL FRAGMENT NUMBER

Code	DCMT Command
N072	DISPLAY RUN UNITS
N072000	DISPLAY RUN UNIT *run-unit-n*
N072002	DISPLAY RUN UNIT QUEUE
N072003	DISPLAY RUN UNIT LOADER
N072004	DISPLAY RUN UNIT MSGDICT
N072005	DISPLAY RUN UNIT SIGNON
N072006	DISPLAY RUN UNIT SYSTEM/DEST
N072009	DISPLAY RUN UNIT SECURITY
N072012	DISPLAY RUN UNIT SQL LOADER
N072013	DISPLAY RUN UNIT SQL SECURITY
N073	VARY RUN UNIT
N073002	VARY RUN UNIT QUEUE
N073003	VARY RUN UNIT LOADER
N073004	VARY RUN UNIT MSGDICT
N073005	VARY RUN UNIT SIGNON
N073006	VARY RUN UNIT SYSTEM/DESTINATION
N073009	VARY RUN UNIT SECURITY
N073012	VARY RUN UNIT SQL LOADER
N073013	VARY RUN UNIT SQL SECURITY
N074	VARY DATABASE READ DRIVER ON
N074000	VARY DATABASE READ DRIVER ON
N075	VARY DATABASE READ DRIVER OFF
N075000	VARY DATABASE READ DRIVER OFF
N076	DISPLAY SUBTASK/MT
N076001	DISPLAY MPMODE TABLE
N076002	DISPLAY SUBTASK
N076003	DISPLAY SUBTASKS
N076004	DISPLAY MT QUEUE DEPTH
N076005	DISPLAY SUBTASK EFFECTIVENESS
N077	VARY SUBTASK/MT QUEUE
N077001	VARY SUBTASK N RRS ENABLED
N077002	VARY SUBTASK N RRS DISABLED
N077003	VARY MT QUEUE DEPTH *nnn*

Code	DCMT Command
N078	DISPLAY STATISTICS AREA
N078000	DISPLAY STATISTICS AREAS
N078001	DISPLAY STATISTICS AREA *area-name*
N078002	DISPLAY STATISTICS AREA *area-name* FILE
N078003	DISPLAY STATISTICS AREA *area-name* BUFFER
N078004	DISPLAY STATISTICS AREA *area-name* ALL
N079	DISPLAY STATISTICS BUFFER
N079000	DISPLAY STATISTICS BUFFERS
N079001	DISPLAY STATISTICS BUFFER *buffer-name* AREA
N079002	DISPLAY STATISTICS BUFFER *buffer-name* FILE
N079003	DISPLAY STATISTICS BUFFER *buffer-name*
N079004	DISPLAY STATISTICS BUFFER *buffer-name* ALL
N080	DISPLAY STATISTICS FILE
N080000	DISPLAY STATISTICS FILES
N080001	DISPLAY STATISTICS FILE *file-name* AREA
N080002	DISPLAY STATISTICS FILE *file-name*
N080003	DISPLAY STATISTICS FILE *file-name* BUFFER
N080004	DISPLAY STATISTICS FILE *file-name* ALL
N081	DISPLAY FILES
N081000	DISPLAY FILES
N081001	DISPLAY FILE *file-name* AREA
N081002	DISPLAY FILE *file-name*
N081003	DISPLAY FILE *file-name* BUFFER
N081004	DISPLAY FILE *file-name* ALL
N081128	DISPLAY FILE *file-name* LOC

Code	DCMT Command
N082	VARY FILE
N082000	VARY FILE *file-name* ACTIVE
N082001	VARY FILE *file-name* INACTIVE
N082002	VARY FILE *file-name* OPEN
N082003	VARY FILE *file-name* CLOSE
N082004	VARY FILE *file-name* OPEN UPDATE
N082005	VARY FILE *file-name* ALLOCATE
N082006	VARY FILE *file-name* DEALLOCATE
N082007	VARY FILE *file-name* DEALLOCATE FORCE
N082008	VARY FILE *file-name* DATASPACE/MEMORY CACHE YES
N082009	VARY FILE *file-name* DATASPACE/MEMORY CACHE NO
N082010	VARY FILE *file-name* DSNAME
N082011	VARY FILE *file-name* DISP SHR
N082012	VARY FILE *file-name* DISP OLD
N082015	VARY FILE *file-name* PREFETCH ON
N082016	VARY FILE *file-name* PREFETCH OFF
N082017	VARY FILE *file-name* SHARED CACHE ca*che-name*
N082018	VARY FILE *file-name* SHARED CACHE NO
N083	DISPLAY LOG
N083001	DISPLAY LOG
N083002	DISPLAY LOG DRIVERS
N084	VARY LOG DRIVER
N084001	VARY LOG DRIVER ONLINE
N084002	VARY LOG DRIVER OFFLINE
N085	VARY JOURNAL TRANSACTION LEVEL
N085000	VARY JOURNAL TRANSACTION LEVEL
N086	DISPLAY DEADLOCK
N086001	DISPLAY DEADLOCK DETECTION INTERVAL
N086002	DISPLAY DEADLOCK DETAILS
N087	VARY DEADLOCK
N087001	VARY DEADLOCK DETECTION INTERVAL
N087003	VARY DEADLOCK DETAILS OFF
N087004	VARY DEADLOCK DETAILS ON
N088	DISPLAY PHYSICAL DATABASE
N088000	DISPLAY PHYSICAL DATABASE
N088001	DISPLAY PHYSICAL DATABASE *database-name*

Code	DCMT Command
N089	VARY DMCL
N089000	VARY DMCL VALIDATE NEW COPY
N089001	VARY DMCL NEW COPY
N089002	VARY DMCL PREFETCH ON
N089003	VARY DMCL PREFETCH OFF
N089004	VARY DMCL PREFETCH TRACE ON
N089005	VARY DMCL PREFETCH TRACE OFF
N089006	VARY DMCL MEMORY CACHE STORAGE LIMIT *nnn x*B
N089007	VARY DMCL MEMORY CACHE STORAGE LIMIT OPSYS
N089008	VARY DMCL MEMORY CACHE LOCATION 64 BIT ONLY
N089009	VARY DMCL MEMORY CACHE LOCATION ANYWHERE
N090	DISPLAY SEGMENTS
N090000	DISPLAY SEGMENTS
N090001	DISPLAY SEGMENT *segment-name*
N091	VARY SEGMENT
N091000	ACTIVE
N091001	RETRIEVAL *<additional parameters>*
N091002	UPDATE (ONLINE) *<additional parameters>*
N091003	OFFLINE *<additional parameters>*
N091004	TRANSIENT RETRIEVAL *<additional parameters>*
N091011	OPEN UPDATE
N091012	OPEN
N091013	CLOSE
N091014	PURGE
N091015	QUIESCE
N091016	DEALLOCATE
N091017	SHARED CACHE ca*che-name*
N091018	SHARED CACHE NO
N091020	DATA SHARING ON
N091021	DATA SHARING OFF
N091022	ALLOCATE
N092	DISPLAY/VARY RESOURCE TABLE/NODE
N092001	DISPLAY RESOURCE TABLE
N092002	DISPLAY NODE
N092003	VARY RESOURCE TABLE NEW COPY

Code	DCMT Command
N093	DISPLAY LOCKS
N093001	DISPLAY LOCKS AREAS
N093002	DISPLAY LOCKS AREA *area-name*
N093003	DISPLAY LOCKS LTERMS
N093004	DISPLAY LOCKS LTERM *logical-terminal-id*
N093005	DISPLAY LOCK STATISTICS
N093006	DISPLAY LOCK RECORD DATA
N093007	DISPLAY LOCK RECORD DATA MEMBER *member-name*
N094	DISPLAY SYSGEN
N094001	DISPLAY SYSGEN REFRESH ALL
N094002	DISPLAY SYSGEN REFRESH LINES
N094003	DISPLAY SYSGEN REFRESH LINE *line-id*
N094004	DISPLAY SYSGEN REFRESH STORAGE POOLS
N094005	DISPLAY SYSGEN REFRESH STORAGE POOL *nnn*
N094006	DISPLAY SYSGEN REFRESH PROGRAM POOLS
N094007	DISPLAY SYSGEN REFRESH PROGRAM POOL XAPP/XARP
N095	VARY SYSGEN
N095001	VARY SYSGEN REFRESH ALL
N095002	VARY SYSGEN REFRESH LINES
N095003	VARY SYSGEN REFRESH LINE *line-id*
N095004	VARY SYSGEN REFRESH STORAGE POOLS
N095005	VARY SYSGEN REFRESH STORAGE POOL *nnn*
N095006	VARY SYSGEN REFRESH PROGRAM POOLS
N095007	VARY SYSGEN REFRESH PROGRAM POOL XAPP/XARP

Code	DCMT Command
N096	DISPLAY/VARY/TEST DBGROUP/SHARED CACHE/DATA SHARING
N096001	DISPLAY DBGROUP
N096002	DISPLAY DBGROUP DEBUG
N096005	DISPLAY SHARED CACHE
N096006	DISPLAY SHARED CACHE DEBUG
N096007	DISPLAY XES LIST &vbar. LOCK *strname*
N096008	DISPLAY XES LIST &vbar. LOCK *strname* DEBUG
N096010	VARY DBGROUP *group-name* ON
N096011	VARY DBGROUP *group name* ACTIVE
N096012	VARY DBGROUP *group-name* JOIN
N096015	VARY DBGROUP *group-name* OFF
N096016	VARY DBGROUP *group name* INACTIVE
N096017	VARY DBGROUP *group-name* LEAVE
N096020	VARY SHARED CACHE *cache-name* ON
N096021	VARY SHARED CACHE *cache-name* OFF
N096030	DISPLAY DATA SHARING SUMMARY
N096031	DISPLAY DATA SHARING XES LOCK
N096032	DISPLAY DATA SHARING XES LIST
N096033	DISPLAY DATA SHARING XCF GROUP
N096034	DISPLAY DATA SHARING ALL
N096035	DISPLAY DATA SHARING DEBUG
N096036	VARY DATA SHARING DEFAULT CACHE
N096037	VARY DATA SHARING ON CONNECTIVITY LOSS option
N096090	TEST DBGROUP count *member-name*
N096091	TEST SHARED CACHE *file-id*
N096092	TEST XCFGRP *group-name* option
N096093	TEST XCFMSG function *message-type* option
N096094	TEST XESLOCK option count
N096095	TEST XESLIST option count
N097	TEST LRBK
N097001	TEST LRBK *run-unit-id*
N098	QUIESCE
N098001	QUIESCE AREA
N098002	QUIESCE SEGMENT
N098003	QUIESCE DBNAME

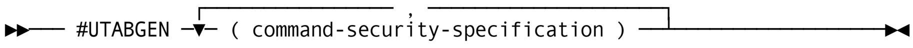
Code	DCMT Command
N099	DISPLAY ID
N099001	DISPLAY ID
N099002	DISPLAY ID NAME
N100	DISPLAY CSAFLAGS & VARY
N100000	CSATST/CSALMGR/CSAHPCS/CSADBIO/CSACFIM
N100001	DISPLAY CSAFLAGS
N100002	VARY CSATST *nn* ON/OFF
N100003	VARY CSALMGR *n* ON/OFF
N100004	VARY CSAHPCS *n* ON/OFF
N100005	VARY CSADBIO *n* ON/OFF
N100006	VARY CSACFIM *n* ON/OFF
	VARY CSATMGR *n* ON/OFF
N101	VARY ID
N101003	VARY ID *xxx* TERMINATE
N102	DISPLAY/VARY TRANSACTION SHARING
N102000	DISPLAY TRANSACTION SHARING
N102001	VARY TRANSACTION SHARING ON/OFF
N103	DISPLAY/VARY TRACE/SYSTRACE
N103001	DISPLAY SYSTRACE
N103002	VARY SYSTRACE OFF
N103003	VARY SYSTRACE ON
N103007	DISPLAY TRACE
N103008	VARY TRACE
N104	DISPLAY/VARY SCRATCH
N104001	DISPLAY SCRATCH
N104002	VARY SCRATCH
N105	VARY CHANGE TRACKING
N105002	FILE COUNT *nnn*/DELETE
	ON/OFF/REFRESH/ACTIVE/INACTIVE/DISABLE/ENABLE
N106	DISPLAY CHANGE TRACKING

Code	DCMT Command
N107	DISPLAY TCP/IP
N107001	DISPLAY TCP/IP ALL
N107002	DISPLAY TCP/IP SUMMARY
N107003	DISPLAY TCP/IP STATISTICS
N107004	DISPLAY TCP/IP STACK TABLE
N107005	DISPLAY TCP/IP SERVICES
N107006	DISPLAY TCP/IP SOCKETS
N108	VARY TCP/IP
N108001	VARY TCP/IP STATUS
N108002	VARY TCP/IP TCP_NODELAY
N108003	VARY TCP/IP DEFAULT STACK
N108004	VARY TCP/IP INCLUDE/EXCLUDE STACK
N108005	VARY TCP/IP MAXIMUM SOCKETS
N108006	VARY TCP/IP MAXIMUM SOCKETS PER TASK
N108007	VARY TCP/IP SERVICES FILE
N108008	VARY TCP/IP STACK TABLE
N109	VARY JOURNAL FILE
N10900	VARY JOURNAL FILE *journal-file-name* ACTIVE
N10901	VARY JOURNAL FILE *journal-file-name* INACTIVE
N10905	VARY JOURNAL FILE *journal-file-name* ALLOCATE
N10906	VARY JOURNAL FILE *journal-file-name* DEALLOCATE
N10907	VARY JOURNAL FILE *journal-file-name* DEALLOCATE FORCE
N10910	VARY JOURNAL FILE *journal-file-name* DSNAME
N10911	VARY JOURNAL FILE *journal-file-name* DISP SHR
N10912	VARY JOURNAL FILE *journal-file-name* DISP OLD
N10913	VARY JOURNAL FILE *journal-file-name* ONLINE
N10914	VARY JOURNAL FILE *journal-file-name* OFFLINE (PERmanent)
N110	DISPLAY AUTOTUNE
N110001	AUTOTUNE [parameter-name]
N110002	AUTOTUNE *
N111	VARY AUTOTUNE
N111001	AUTOTUNE … RESET
N111002	AUTOTUNE … OFF
N112	DISPLAY MODID
N112001	[FROM/TO]
N112002	module-name

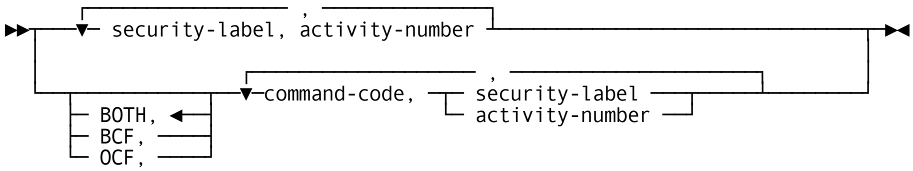
#UTABGEN

Assigns activity numbers to utility commands.

#UTABGEN Syntax



Expansion of Command-Security-Specification



#UTABGEN Parameters

security-label,activity-number

Defines a security label and associates it with a BCF or OCF activity number.

A security label can be used to classify utility commands by assigning a security label to one or more utility command codes. All commands with the same security label will then be associated with the security label's activity number.

A security label must be one alphabetic character (A through Z). You can define a maximum of 26 security labels in the #UTABGEN macro. Valid activity-numbers are in the range of 0-255.

Note: An activity number of zero means no security.

BOTH, BCF, OCF,

Specifies whether the current group of command code security assignments (within the parentheses), will apply to both BCF and OCF commands, just BCF commands, or just OCF commands. BOTH is the default.

Note: The terms BCF and OCF are used to distinguish between operations processed inside the CV from those processed in the batch address space. This means that the term BCF applies to local mode batch only, while the term OCF applies to both OCF and batch to CV (that is, central mode batch).

command-code

Defines a code from the predefined utility command code table that represents a utility command. For example, FORMAT identifies the Format utility. PRINTSPACE identifies the Print Space utility.

Note: A list of valid utility commands follows:.

security-label

Specifies a previously defined security label that you are associating with *command-code*.

activity-number

Specifies the BCF/OCF activity number you are associating with *command-code*. Valid activity-numbers are in the range of 0-255.

Note: An activity number of zero means no security.

#UTABGEN Usage

Coding Considerations

All lines except the first one must start in column 16.

All lines except the last one must have a non-blank character in column 72.

General

When you use the #UTABGEN macro, you can assign an OCF/BCF activity number to one or more Utility commands.

- You can associate a specific OCF/BCF *activity number* (0 through 255) with a utility command.
- You can associate a *security label* (A through Z) with a utility command.

An activity number of zero turns off security for that security-label or command-code. Coding zero is a useful way to turn off security, without deleting the command-code from the #UTABGEN source definition.

Only commands that are being secured need to be coded. If omitted, they default to an activity code of zero.

Security labels must be defined before they can be assigned to command codes.

Use of security labels makes it easier to maintain security definitions when several commands are assigned the same OCF/BCF activity number. You define a security label in the #UTABGEN macro itself. You need to only change the security label definition in the #UTABGEN macro to modify the security for all associated DCMT commands.

Generating the #UTABGEN Macro

The source file that specifies the #UTABGEN macro can only contain one macro. Once assembled, the resulting object must be link edited as the stand-alone module IDMSUTAB.

Example 1:

```
----+----1----+----2----+----3----+----4----+----5----+----6----+----7--

         #UTABGEN (A,3,B,10),                                          X
               (OCF,FORMAT,A,LOCK,A,UNLOCK,A),                         X
               (BCF,ARCHIVEJOURNAL,B,ARCHIVELOG,B),                    X
               (FIXPAGE,50)
         END
```

This example shows an activity code of 3 being assigned to security-label A and activity code 10 being assigned to security-level B.

OCF indicates the commands that follow (within the parentheses) will be assigned activity codes only when running in the online command facility OCF or as part of the batch command facility IDMSBCF running in central mode. Commands FORMAT, LOCK and UNLOCK are associated with security-label A. Since security-label A is currently assigned to the OCF/BCF activity code 3, the Format, Lock, and Unlock commands are assigned activity code 3.

BCF indicates that the commands within that group will only be secured when running as part of the batch command facility IDMSBCF running in local mode only. In this example, the Archive Journal and Archive Log commands will be assigned to the activity code 10, via the security-label B.

FIXPAGE is not qualified so the activity code 50 will be assigned to the Fix Page utility in both OCF and BCF.

Example 2:

```
----+----1----+----2----+----3----+----4----+----5----+----6----+----7--

         #UTABGEN (FORMAT,14,FIXPAGE,14)
         END
```

In this example, the activity number of 14 is assigned the utility command codes FORMAT and FIXPAGE. Because the codes are not identified as being either OCF or BCF, the commands associated with these codes will be secured in both online and batch, and both will use the same activity number.

#UTABGEN Utility Command Codes

Code	Utility Command
ARCHIVEJOURNAL	Archive Journal
ARCHIVETRACE	ARCHIVE TRACE

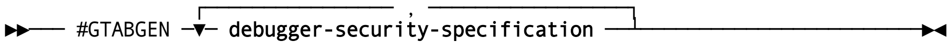
Code	Utility Command
ARCHIVELOG	Archive Log
BACKUP	Backup
BUILD	Build Indexes/Constraints
CLEANUP	Cleanup Segment/Area
CONVERTCATALOG	Convert catalog
CONVERTPAGE	Convert Page
EXPANDPAGE	Expand Page
FASTLOAD	Fastload
FIXARCHIVE	Fix Archive
FIXPAGE	Fix Page
FORMAT	Format Area/Segment/File
INSTALLSTAMPS	Install Stamps
LOAD	Load
LOCK	Lock Area/Segment
MAINTAINASF	Maintain ASF
MAINTAININDEX	Maintain Index
MERGEARCHIVE	Merge Archive
PRINTINDEX	Print Index
PRINTJOURNAL	Print Journal
PRINTLOG	Print Log
PRINTPAGE	Print Page
PRINTSPACE	Print Space
PRINTTRACE	PRINT TRACE
PUNCHLOADMODULE	Punch Load Module
RELOAD	Reload
REORG	Unload and reload all or part of a database
RESTORE	Restore
RESTRUCTURE	Restructure
RESTRUCTURECONNECT	RestructureConnect

Code	Utility Command
ROLLBACK	RollBack
ROLLFORWARD	RollForward Extract Journal
SETOPTIONS	Set BCF/OCF Options
SYNCHRONIZESTAMPS	Synchronize stamps
TUNEINDEX	Tune Index
UNLOAD	Unload
UNLOCK	Unlock Area/Segment
UPDATESTATISTICS	Update Statistics
VALIDATE	Validate

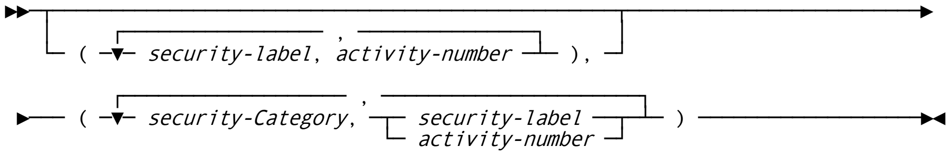
#GTABGEN

Assigns activity numbers to online debugger security categories.

#GTABGEN Syntax



Expansion of Debugger-Security-Specification



#GTABGEN Parameters

security-label,activity-number

Associates an online debugger security label with an activity number.

Security-label specifies a label with which you can associate one or more online debugger security categories.

Security-label must be one alphabetic character (A through Z). You can define at most 26 security labels in the #GTABGEN macro.

Activity-number specifies the activity number you are associating with *security-label*. *Activity-number* must be a numeric value in the range 1 through 256.

security-category

Specifies the online debugger security category with which you are associating the activity number or security label.

Security-category can be any value listed in the "Category" column of the following table, which presents each online debugger security category and its description:

Category	Description
ALLR	Any entities named in the following categories can be retrieved.
ALLU	Any entities named in the following categories can be updated.(1)
ASYSPGR	CA ADS runtime system programs (ADSOMAIN, ADSORUN1, and ADSODBUG) can be retrieved.
ASYSPGU	CA ADS runtime system programs can be updated.(1)
ASYSTGR	Storage acquired by the CA ADS runtime system can be retrieved. System storage includes the control block for the online work area (OWA), the online terminal block (OTB), the online terminal block extension (OTBX), and the variable dialog block (VDB).
ASYSTGU	Storage acquired by the CA ADS runtime system can be updated.(1)
AUPGMR	CA ADS user programs can be retrieved. These include the fixed dialog block (FDB), the application definition block (ADB), and the task application table (TAT).
AUPGMU	CA ADS user programs can be updated.(1)
SHSTGR	Shared storage can be retrieved.
SHSTGU	Shared storage can be updated.(1)
UPGMR	User programs, subschemas, maps, and tables can be retrieved.
UPGMU	User programs, subschemas, maps, and tables can be updated.(1)
USTGR	User storage can be retrieved.
USTGU	User storage can be updated.(1)

(1) Update mode also implies that programs can be retrieved. In retrieval mode you can list and set breakpoints.

security-label

Specifies the online debugger security label that you are associating with *security-category*.

This occurrence of *security-label* must match a security label that is specified in a prior parenthesized list in the macro.

activity-number

Specifies the online debugger activity number that you are associating with *security-category*.

#GTABGEN Usage

Generating the #GTABGEN Macro

To assemble and link edit the #GTABGEN macro, use the JCL or commands appropriate to your operating system.

Note: For more information, see Appendix A: Security Macro JCL. (see page 379)

In the following sample #GTABGEN macro, two security labels are defined:

- A, which is assigned activity number 20
- B, which is assigned activity number 30

UPGMU, the security category that allows updates of user programs, is assigned label A (and, therefore, activity number 20). USTGU, the category that allows updates of storage, is assigned security label B (and, therefore, activity number 30). ALLR, the category that allows retrieval only for all programs and storage is assigned activity number 50.

```
----+----1----+----2----+----3----+----4----+----5----+----6----+----7--
         #GTABGEN(A,20,B,30),                                     X
               (UPGMU,A,USTGU,B,ALLR,50)
```

#SECHECK

Checks a user's authority to access a resource.

#SECHECK Authorization

No authorization required.

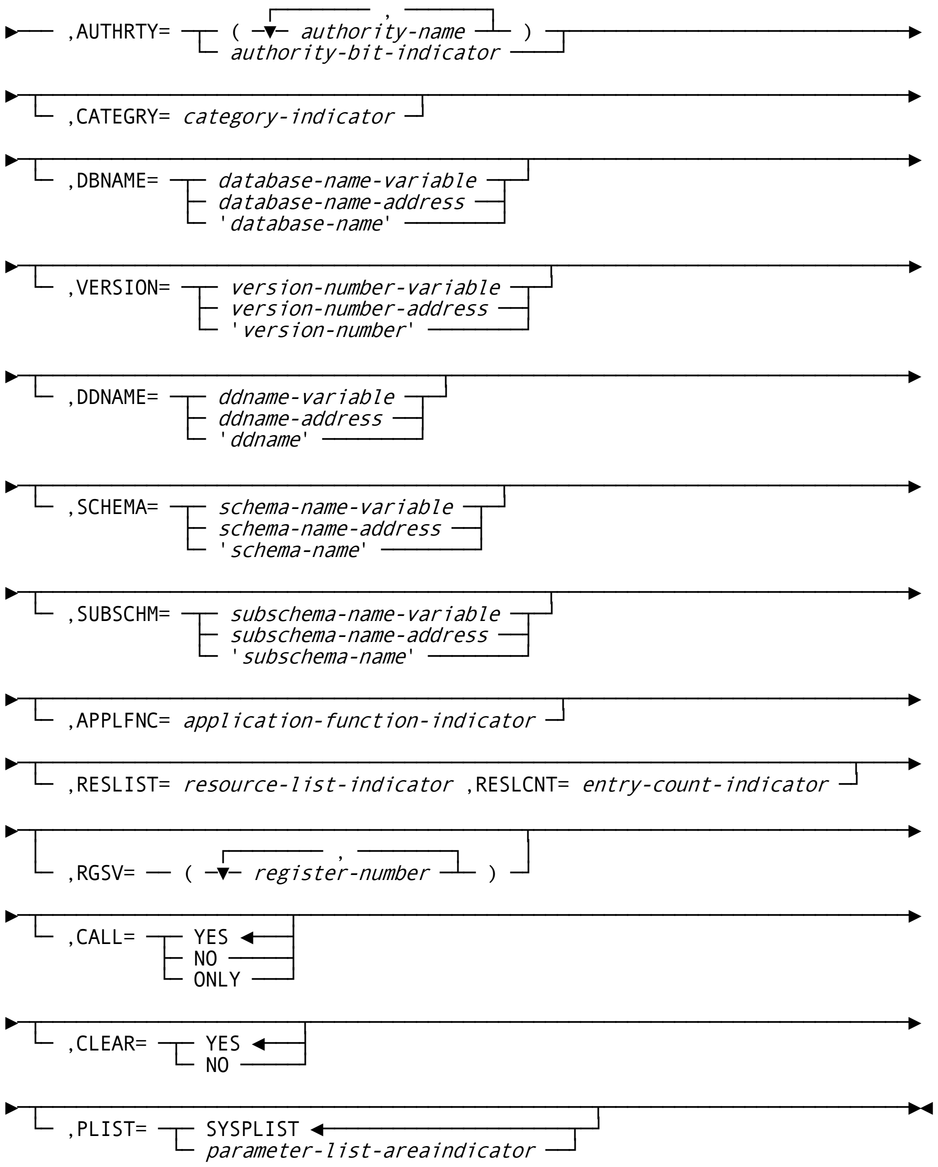
#SECHECK Syntax

```
►►── #SECHECK ──────────────────────────────────────────────────────────►

►── SRB= ─┬─ request-block-area-name ─┬─────────────────────────────────►
          └─ request-block-address ───┘

►── ,RESTYPE= ─┬─ resource-type-variable ─┬─────────────────────────────►
               ├─ resource-type-address ──┤
               └─ 'resource-type-name' ───┘

►── ,RESNAME= ─┬─ resource-name-variable ─┬─────────────────────────────►
               ├─ resource-name-address ──┤
               └─ 'resource-name' ────────┘
```



#SECHECK Parameters

SRB=

Specifies the area containing the security request block (SRB) associated with this request. This area is mapped by DSECT #SECRB.

The location of the SRB is required because most of the other parameters coded on the #SECHECK will be stored in the SRB.

request-block-area-name

Specifies the symbolic name of the area containing the SRB.

request-block-address

Specifies the register containing the address of the SRB.

,RESTYPE=

Specifies the type of the resource being checked. The name of the resource type must be one- to four-characters. At runtime the resource type must be defined to CA IDMS centralized security.

Note: For a list of valid CA IDMS resource types, see #SECRTT (see page 204).

#SECHECK does not validate the resource type. However, the security system will fail the request at runtime if the type is not valid.

resource-type-variable

Specifies a user-defined field containing the name of the resource type.

The field must be at least four bytes in length. The name of the resource type must be left-justified and padded with blanks.

resource-type-address

Specifies a register containing the address of the resource type name.

'*resource-type-name*'

Specifies a literal that is the name of the resource type.

,RESNAME=

Specifies the name of the resource being checked.

resource-name-variable

Specifies a user-defined field containing the name of the resource.

The resource name must be left-justified and padded with blanks or binary zeros.

resource-name-address

Specifies a register containing the address of the resource name.

'*resource-name*'

Specifies a literal that is the name of the resource.

,AUTHRTY=

Specifies the authorities the user must hold to gain access to the resource being checked.

authority-name

Specifies the name of the authority. If you specify only one authority, you may omit parentheses.

Valid authority names are as follows:

ALTER
CREATE
DBADMIN
DBAREAD
DBAWRITE
DCADMIN
DELETE
DISPLAY
DROP
EXECUTE
INSERT
REFERENCES
SELECT
SIGNON
SYSADMIN
UPDATE
USE

authority-bit-indicator

Specifies a user-defined six-byte field or a register containing the address of such a field. You must set the bits in the field corresponding to the required authorities.

,CATEGRY=*category-indicator*

Specifies whether the security system should update the category table from the catalog before proceeding with the security check.

If *category-indicator* is 0, the security system will unconditionally update the category table from the catalog before making the security check.

If *category-indicator* is a non-zero value, the security system will update the category table from the catalog before making the security check only if the category has not already been retrieved.

category-indicator must be one of the following:

- The name of a user-defined halfword field containing the indicator
- A register containing the indicator value
- A literal representing the indicator

If the category is zero, the security system will retrieve the assigned value from the security database and return it in the SRB.

CATEGRY is valid only for resource types which can be secured by categories. If you specify CATEGRY for the resource, do not specify the APPLFNC parameter.

,DBNAME=

Specifies the database name for the load area in which resource types such as database resources and load modules for CA ADS dialogs reside. The runtime security system ignores the contents of this field for other resource types.

Resource type DB: Use the RESNAME parameter to supply database name if the security check is for resource type DB.

database-name-variable

Specifies the name of a user-defined field that contains the database name.

The database name must be left-justified and padded with blanks or binary zeros.

database-name-address

Specifies the register with the address of the user-defined field that contains the database name.

'*database-name*'

Supplies the database name as a character string literal.

,VERSION=

Specifies the version for load modules (resource type SLOD) and non-SQL schemas (resource type NSCH). The value in this parameter must be in the form V*nnnn*, where *nnnn* is the version in character format.

If you specify VERSION= for the resource, do not specify the DDNAME parameter.

version-number-variable

Specifies the name of a user-defined field that contains the version number.

version-number-address

Specifies the register with the address of the user-defined field that contains the version number.

'*version-number*'

Supplies the version number as a character string literal.

DDNAME=

Specifies the ddname defining the operating system library in which a program (resource type SPGM) resides.

If you specify DDNAME for the resource, do not specify the VERSION= parameter.

ddname-variable

Specifies the name of a user-defined field that contains the ddname.

The ddname must be left-justified and padded with blanks.

ddname-address

Specifies the register with the address of the user-defined field that contains the ddname.

'*ddname*'

Supplies the ddname as a character string literal.

,SCHEMA=

Specifies the name of the schema for SQL tables (resource type TABL) and access modules (resource type DACC and SACC).

Resource type SCHEMA: Use the RESNAME parameter to specify schema name if the security check is for a schema (resource types QSCH and NSCH).

schema-name-variable

Specifies the name of a user-defined field that contains the schema name.

Schema name must be left-justified and padded with blanks.

schema-name-address

Specifies the register with the address of the user-defined field that contains the schema name.

'*schema-name*'

Supplies the schema name as a character string literal.

,SUBSCHM=

Specifies the subschema name for native run units (resource type NRU).

subschema-name-variable

Specifies the name of a user-defined field that contains the subschema name.

Subschema name must be left-justified and padded with blanks.

subschema-name-register

Specifies the register with the address of the user-defined field that contains the subschema name.

'*subschema-name*'

Supplies the subschema name as a character string literal.

,APPLFNC=

Specifies the number associated with an application function. Each application can have up to 256 functions, numbered 1 through 256. Function numbers must be unique within a given application but need not be unique across applications.

APPLFNC is valid for activities (resource type ACTI) only. If you specify APPLFNC for the resource, do not specify the CATEGRY parameter.

application-function-indicator

Supplies the application function number.

Application-function-indicator can be one of the following:

- User-supplied halfword field
- Register containing the function number
- Numeric literal

,RESLIST=

Specifies a list of resources to be checked on this call.

The resource list contains one entry for each resource being checked. All resources in the list must be of the same resource type.

Note: For more information about the format of the entries in a resource list, see DSECT #SECRLST (see page 432) in Security Database Information and DSECTs.

resource-list-indicator

Specifies one of the following:

- A user-defined field that contains the name associated with the first entry in the resource list.
- The register with the address of the first entry in the resource list.

,RESLCNT=

Specifies the number of entries in the list specified by RESLIST.

entry-count-indicator

Specifies one of the following:

- A user-defined halfword field that contains the entry count.
- The register that contains the entry count.
- The entry count represented by a numeric literal.

,RGSV=

Specifies that one or more registers are to be saved across the call. This parameter is valid in system mode only.

,register-number

Specifies a register.

Register-number must be a numeric literal.

,CALL=

Controls the expansion of the #SECHECK macro.

If you omit the CALL parameter, the effect is the same as specifying CALL=YES.

YES

Causes #SECHECK to generate both the code to complete the SRB and invoke the security system.

NO

Causes #SECHECK to generate the code to fill in the SRB fields, but not to build the parameter list or the call.

ONLY

Causes #SECHECK to generate only the code needed to invoke the security system.

,CLEAR=

Specifies whether you want the SRB to be initialized.

If you omit the CLEAR= parameter, the effect is the same as specifying CLEAR=YES.

YES

Causes #SECHECK to clear the SRB to binary zeros before the macro expansion begins to assign values.

NO

Indicates that the SRB should not be initialized.

,PLIST=

Specifies the address of the area in which to build the parameter list.

SYSPLIST

Supplies the default name for the area that contains the parameter list.

parameter-list-area-indicator

Overrides the default area name SYSPLIST by specifying one of these:

- The name of a user-defined fullword-aligned field that contains the address of the parameter list
- A register with the address of the parameter list

#SECHECK Usage

Copying the Security Request Block

To issue the #SECHECK macro, you must copy the Security Request Block. This block is mapped by the #SECRB DSECT.

Note: For more information, see the documentation of #SECRB (see page 428).

Multiple Security Checks in a Single Request

The #SECHECK function supports multiple security checks in a single request if all the resources are of the same type. For example, all authorities needed by an access module can be validated in a single request.

An application can request that authorization for a list of resources be checked. This reduces the number of calls to the security manager.

Validating Parameters

#SECHECK does not validate parameters based on resource type. All parameters supplied on the call are stored in the SRB. The security system ignores information which is irrelevant to the resource type.

#SECHECK Return Codes

The return code for a security check is stored in register 15 and the SRBXR15 field of the SRB. The following table lists the possible return codes provided by the internal security system in response to a #SECHECK macro:

Code Resource grouping	**Meaning Resource**
00	Request was successful; access allowed
04	Resource occurrence or object (user, group, etc.) not found
08	User not authorized; access denied
12	Interface/parameter list error
16	Resource access threshold violation

When using the multiple security check option, register R15 contains return code 0 only if the user has access to all the resources. The security system places the return codes for the individual resources in the area (RTNADDR) supplied by the caller.

Return Codes for a List of Resources

To check the return code for each entry in a list of resources supplied with a security request, you must copy #SECRLST. When using this option, determine the results from all list entries if the return code is not 12 (invalid request). R15=0 does not necessarily mean the user is authorized for all entries in the list.

External Return Codes

If the external security system return code is 4, the issuer of #SECHECK will receive a return code of 8. Other non-zero codes from the external system will be passed through as is. The CA IDMS internal security system is not used as a fallback system for external security.

#SECRTT

Generates the table used to route security check requests for each resource type to the internal or external security system.

#SECRTT Authorization

Authorization to assemble the #SECRTT macro, if any, is defined in an external security system.

#SECRTT Syntax

Syntax for TYPE=INITIAL

```
►►─── #SECRTT TYPE=INITIAL ────────────────────────────────────────────►

►─┬──────────────────────────────────────┬─────────────────────────────►
  └─ ,ENVNAME= ─┬─ environment-name ─┬─┘
                └─ NULL ◄────────────┘

►─┬─────────────────────────────────┬──────────────────────────────────►
  └─ ,SGNRETN= ─┬─ time-interval─┬─┘
                └─ OFF ◄─────────┘

►─┬─────────────────────────────────────────────────────────────────┬──►
  └─ ,SYSPROF= ── ( ─┬─ OFF ──────────┬─┬──────────────────┬─ ) ─┘
                     ├─ NULL ─────────┤ └─ , ─┬─ ON ───┬─┘
                     ├─ USER ─────────┤       └─ OFF ◄─┘
                     ├─ GROUP ────────┤
                     ├─ SYSTEM ───────┤
                     ├─ DEFAULT ◄─────┤
                     └─ profile-name ─┘
```

```
►─┬────────────────────────────────────────────────────────────────────┬─►
  └─ ,USRPROF= ── ( ─┬─ OFF ──────────┬─┬─────────────────────┬─ ) ─┘
                     ├─ NULL ─────────┤ └─ , ─┬─ ON ─────┬─┘
                     ├─ USER ◄────────┤       └─ OFF ◄───┘
                     ├─ GROUP ────────┤
                     ├─ SYSTEM ───────┤
                     └─ profile-name ─┘

►─┬────────────────────────────────────────────────────────────────────┬─►
  └─ ,DFLTSGN= ─┬─ YES ──┬─┘
                └─ NO ◄──┘

►─┬────────────────────────────────────────────────────────────────────┬─►
  └─ ,DFLTUID= ─┬─────────── user-identifier ──────────┬─┘
                └─ ( ─▼─┬──────── , ────────┬──── ) ───┘
                        ├─ VTAMNODE ────────┤
                        ├─ PTERMID ─────────┤
                        └─ LTERMID ─────────┘

►─┬────────────────────────────────────────────────────────────────────┬─►
  └─ ,EXTRUID= user-identifier ──┘

►─┬────────────────────────────────────────────────────────────────────┬─►
  └─ ,MAXRESN= ─┬─ resource-entries ─┬─┘
                └─ 150 ◄─────────────┘

►─┬────────────────────────────────────────────────────────────────────┬─►◄
  └─ ,SVCNUM= ─┬─ svc-number ─┬─┘
               └─ 175 ◄───────┘
```

Syntax for TYPE=ENTRY and TYPE=OCCURRENCE

```
►►── #SECRTT TYPE= ─┬─ ENTRY ──────────────────────────────────────┬─►
                    └─ OCCURrence,RESNAME= 'resource-name', ───────┘

►── RESTYPE= resource-type-name ───────────────────────────────────────►

►─┬────────────────────────────────────────────────────────────────────┬─►
  └─ ,SECBY= ─┬─ EXTernal ─┬─┘
              ├─ INTernal ─┤
              └─ OFF ◄─────┘

►─┬────────────────────────────────────────────────────────────────────┬─►
  └─ ,EXTCLS= ─┬─ resource-class-variable ──┬─┘
               └─ 'resource-class-name' ────┘

►─┬────────────────────────────────────────────────────────────────────┬─►
  └─ ,EXTNAME= ( ─▼─┬────── , ──────┬─── ) ─┘
                    ├─ ACTIvity ────┤
                    ├─ APPLname ────┤
                    ├─ DBNAme ──────┤
                    ├─ DDNAme ──────┤
                    ├─ ENVIr ───────┤
                    ├─ RESName ◄────┤
                    ├─ RESTYPE ─────┤
                    ├─ SCHEma ──────┤
                    ├─ SSNAme ──────┤
                    ├─ SYSTem ──────┤
                    └─ VERSion ─────┘
```

Syntax for TYPE=FINAL

```
►►── #SECRTT TYPE=FINAL ───────────────────────────────────────────────►◄
```

#SECRTT Parameters

TYPE=

Specifies the type of action to result from assembling the macro.

In a series of #SECRTT macros, the first of the series must specify TYPE=INITIAL and the last must specify TYPE=FINAL.

INITIAL

Specifies that entries in the SRTT for all CA IDMS-defined resources are to be initialized:

For each resource type, the initial values are the following:

- SECBY=OFF
- EXTNAME=(RESNAME)
- EXTCLS=blanks

ENVNAME=*environment-name*

Specifies a name for the environment that uses the SRTT. *Environment-name* can be used in external resource name construction.

Environment-name must be one to eight characters in length.

NULL

Specifies that there is no name for the environment that uses the SRTT.

SGNRETN

Specifies whether CA IDMS should retain signon information originating from external request units (ERUs). This option will provide performance improvements in environments which process large numbers of short-lived ERUs and external security systems.

time-interval

Specifies the time in minutes that CA IDMS should retain signon information for external request units after the last session has been ended by signoff.

You can specify the CA IDMS command, DCUF SHOW USERS ALL, to show the retained users signons with an LTERMID of *NONE*.

Note: If a user signs on to the CA IDMS CV through a VTAM or TSO UCF connection and this is the last (or only) session, a FULL signoff will be performed and the retained signon information and control blocks will be freed from the CA IDMS CV.

OFF

Specifies that a full signoff, which frees all retained control blocks, will be performed at the end of the last (or only) session for the user. OFF is the default.

SYSPROF=

Specifies the default SYSTEM profile and whether SYSTEM profiles should be processed for external run units.

OFF

Specifies that no SYSTEM profile should be processed.

Note: If SYSTEM profiles are OFF, they will be off for all tasks including external run units, regardless of the setting of the second subparameter.

NULL

Specifies that there is no default SYSTEM profile.

USER

Specifies that the default SYSTEM profile name is the user-id.

GROUP

Specifies that the default SYSTEM profile name is the name of the user's default group.

SYSTEM

Specifies that the default SYSTEM profile name is the SYSTEM ID defined in SYSGEN.

***profile-name*/DEFAULT**

Specifies the name of the default profile. The profile name must be 1 to 18 characters.

ON

Indicates that profiles should be processed for external run units. The default profile, if any, is specified by the first subparameter.

OFF

Indicates that profiles should not be processed for external run units. The default is OFF.

USRPROF=

Specifies the default USER profile and whether USER profiles should be processed for external run units.

OFF

Specifies that no USER profile should be processed.

Note: If USER profiles are OFF, they will be off for all tasks including external run units, regardless of the setting of the second subparameter.

NULL

Specifies that there is no default USER profile.

USER

Specifies that the default USER profile name is the user-id.

GROUP

Specifies that the default USER profile name is the name of the user's default group.

SYSTEM

Specifies that the default USER profile name is the SYSTEM ID defined in SYSGEN.

***profile-name*/DEFAULT**

Specifies the name of the default profile. The profile name must be 1 to 18 characters.

ON

Indicates that profiles should be processed for external run units. The default profile, if any, is specified by the first subparameter.

OFF

Indicates that profiles should not be processed for external run units. The default if OFF.

DFLTSGN=

Specifies whether CA IDMS should perform a signon using a specific name if a security check is issued and the terminal operator has not signed on. The name to use for the default signon is defined by the DFLTUID parameter.

YES

Enables default signon.

NO

Disables this option.

DFLTUID=

Specifies the default signon CA IDMS is to use when the DFLTSGN parameter is enabled, a security check is issued, and the terminal operator has not signed on. Specify a *user-identifier* or a list of up to three ID options in parentheses. If DFLTSGN=YES, and you don't specify DFLTUID parameters, the default is as follows: (VTAMNODE,PTERMID,LTERMID).

user-identifier

Specifies the default signon as an unquoted literal from 1 to 18 characters in length.

VTAMNODE

Specifies that for VTAM terminals, the VTAM node name is used as the default signon.

PTERMID

Specifies that the PTERM ID is used as the default signon, if the PTERM is available and the option has not been satisfied by the VTAMNODE parameter (non-VTAM terminals, or VTAMNODE not specified for VTAM terminals).

LTERMID

Specifies that the LTERM ID is used as the default signon, if the option has not been satisfied by the VTAMNODE or PTERMID parameters.

EXTRUID=

Specifies the extract user ID that can be used at sites that do not have an external security system. *User-identifier* is an unquoted literal from 1- to 18-characters.

MAXRESN=*max-resource-entries*

Specifies maximum number of entries in the #SECRTT global table.

If the default of 150 entries is exceeded, the assembly of the #SECRTT fails with condition code of 12 and an assembler error message displays:

"12, SRTT GLOBAL TABLE OVERFLOW. GENERATION ABORTED".

When this error message is received, review the #SECRTT entries. Check the wildcards to ensure they are valid and used properly. When wildcards are used properly, they reduce the number of entries in #SECRTT global table.

Important! Excessive entries require CPU time to resolve each security check.

150

This is the default. It can be increased if necessary.

SVCNUM=*svc-number*

Specifies the installed SVC number. This parameter is required. If *svc-number* is not specified, the system defaults to 175.

ENTRY

Specifies that the user-supplied values apply to all occurrences of the resource type identified in the RESTYPE parameter.

For each resource type whose default values you want to replace in SRTT, you must issue a #SECRTT macro with TYPE=ENTRY.

OCCURRENCE

Specifies that the user-supplied values apply to one occurrence of the resource type identified in the RESTYPE parameter.

Note: TYPE=OCCURRENCE is valid only for resource types DB, SPGM, and TASK.

EXTCLS= and EXTNAME= specifications are ignored if TYPE=OCCURRENCE. Therefore, if you specify TYPE=OCCURRENCE and SECBY=EXTERNAL to secure an occurrence override externally, be sure to specify EXTCLS= and EXTNAME= on the TYPE=ENTRY macro for the resource type. This information will be used for checks on the occurrence override.

RESNAME='*resource-name*'

Names the occurrence of the resource to which the user-supplied values in the macro apply. You must enclose the resource name in quotes.

If TYPE=OCCURRENCE, the value in *resource-name* is treated as a wildcarded name. Thus, if RESTYPE=SPGM and RESNAME='RHDC', the scope of the override is all program names that begin with 'RHDC'.

If you do not want wildcarding to take effect—that is, you want to limit the scope of the override to only one resource-name—then include a blank character at the end of the resource-name. Thus, if RESTYPE=SPGM and RESNAME='TEST01 ', the scope of the override is the program 'TEST01' only.

RESTYPE=*resource-type-name*

Specifies the resource type you are defining in the SRTT.

Resource-type-name must be 1 to 4 characters in length and may identify a resource type defined by CA IDMS or a user-defined resource type.

Note: For more information about user-defined resource types, see Using External Security (see page 45).

This table lists valid resource type names for CA IDMS resources:

Global resources	SYSADMIN privilege	SYSA
	User	USER
	Group	GROU
	User profile	UPRF

System resources	DCADMIN privilege	DCA
	System	SYST
	System profile	SPRF
	Signon	SGON
	Activity	ACTI
	Task	TASK
	Load module	SLOD
	Queue	QUEU
	Access module	SACC
	Program	SPGM
Database resources	DBADMIN	DB
	Database	DB
	Area	DB (AREA)(1)
	Rununit	DB (NRU)(1)
	Schema (SQL)	DB (QSCH)(1)
	Non-SQL schema	DB (NSCH)(1)
	Access module	DB (DACC)(1)
	Table	DB (TABL)(1)
	DMCL	DMCL
	Database name table	DBTB

(1) Resource type is secured when DB is secured.

Note: DBADMIN privilege is secured when you activate security for DB.

SECBY=

Specifies the security option for the resource type identified in the RESTYPE parameter.

EXTERNAL

Specifies that security-checking for the resource type is performed using definitions in an external security system.

If you specify SECBY=EXTERNAL, you must include the EXTCLS and EXTNAME parameters in the macro.

INTERNAL

Specifies that security-checking for the resource type is performed using security definitions in CA IDMS.

SECBY=INTERNAL is valid for any CA IDMS resource type (see the following table). It is not valid for a user-defined resource type.

OFF

Specifies that no security-checking is performed for the resource type; the resource type is unsecured.

EXTCLS=

Maps the CA IDMS resource type specified in the RESTYPE parameter to the resource class you have defined for this type in the external security system.

EXTCLS is required when TYPE=ENTRY and SECBY=EXTERNAL for the entry or for any occurrence override of the entry.

If EXTCLS is specified, the information is recorded in the SRTT but used only when security enforcement is external.

resource-class-variable

Specifies a variable containing the name of the external resource class.

***resource-class-name*'**

Specifies the name of the external resource class.

EXTNAME=

Using a set of predefined keywords, specifies the fields to be included in the external resource name. The order in which you specify the keywords is the order in which the fields will be included in the external resource name.

Since EXTNAME defines the format of the resource name for external security requests, the format you specify here must match the naming conventions for the corresponding resource class in the external security system.

Note: For more information about constructing external resource names, see Using External Security (see page 45).

EXTNAME is required when TYPE=ENTRY and SECBY=EXTERNAL for the entry or for any occurrence override of the entry.

If EXTNAME is specified, the information is recorded in the SRTT but used only when security enforcement is external.

ACTIvity

Includes in the external resource name the activity number supplied by the application.

When formatted for an external security request, this field will be a 4- to 8-character string that is the concatenation of the following:

- Either the application name or the first 5 characters of the application name (if the full name exceeds 5 characters).
- The 3-digit activity number in displayable format.

APPLname

Includes the full application name, as supplied on the current security request, in the external resource name.

DBNAme

Includes the database name, as supplied on the current security request in the external resource name.

DDNAme

Includes the ddname, as supplied on the current security request, in the external resource name. The ddname defines the operating system library in which the program (resource type SPGM) resides.

ENVIr

Includes the environment name in the external resource name.

RESName

Includes the resource name as specified on the current security request in the external resource name.

RESType

Includes the resource type, as supplied on the RESTYPE= parameter for this SRTT entry, in the external resource name.

SCHEma

Includes the schema name, as supplied on the current security request, in the external resource name. The schema name qualifies the names of SQL tables (resource type TABL) and access modules (resource types DACC and SACC).

SSNAme

Includes the subschema name, as supplied on the current security request, in the external resource name.

SYSTem

Includes the name of the CA IDMS system in the external resource name.

VERSion

Includes the version number for load modules (resource type SLOD) and non-SQL schemas (resource type NSCH), as supplied on the current security request, in the external resource name.

FINAL

Indicates the end of SRTT specifications.

You can specify TYPE=FINAL only once. SRTT entries will be generated from the series of #SECRTT macros beginning with the one that specifies TYPE=INITIAL.

#SECRTT Usage

User-Defined Resource Types

Resource-type-name can be a user-defined resource type. The valid SECBY specifications for a user-defined resource type are EXTERNAL or OFF.

The following short resource type names are reserved by CA IDMS for future use. If you specify one of these short resource type names in a #SECRTT assembly, an error message will be returned.

DDA

NSUB

DMSG

DATT

DCLA

DUDE

DAPP

DIAL

DELE

DFIL

DLOD

DMAP

DMOD

DPAN

DPGM

DREC

DSYS

DUSR

DDES

DLIN

DLTE

DPTE

DQUE

DTSK

DACT

Order of EXTNAME Specification

The order of keywords that you specify in the EXTNAME parameter determines the order of fields in the external resource name format. For example, suppose that you specify for RESTYPE=TASK the following parameter:

```
EXTNAME=(RESTYPE,ENVIR,SYSTEM,RESNAME)
```

The external resource name format for a task will be the following:

TASK.*environment-name*.*system-name*.*task-identifier*

Generating the #SECRTT Macro

To assemble and link edit the #SECRTT macro, you can use the appropriate JCL or commands.

Note: For more information, see the Security Macro JCL (see page 379) appendix.

However, it is recommended that you use SMP to assemble this macro.

Note: For more information, see "System Modification" in the *CA IDMS Installation Manual* for your operating system.

Using a Single Resource Class

In the following example, each resource type is assigned to the class IDMS. In each case, the resource type is part of the external resource name format:

```
----+----1----+----2----+----3----+----4----+----5----+----6----+----7--

   #SECRTT RESTYPE=TASK,SECBY=EXTERNAL,                X
         EXTCLS='IDMS',EXTNAME=(RESTYPE,RESNAME)
   #SECRTT RESTYPE=SPGM,SECBY=EXTERNAL,                X
         EXTCLS='IDMS',EXTNAME=(RESTYPE,DDNAME,RESNAME)
   #SECRTT RESTYPE=ACTI,SECBY=EXTERNAL,                X
         EXTCLS='IDMS',EXTNAME=(RESTYPE,RESNAME)
```

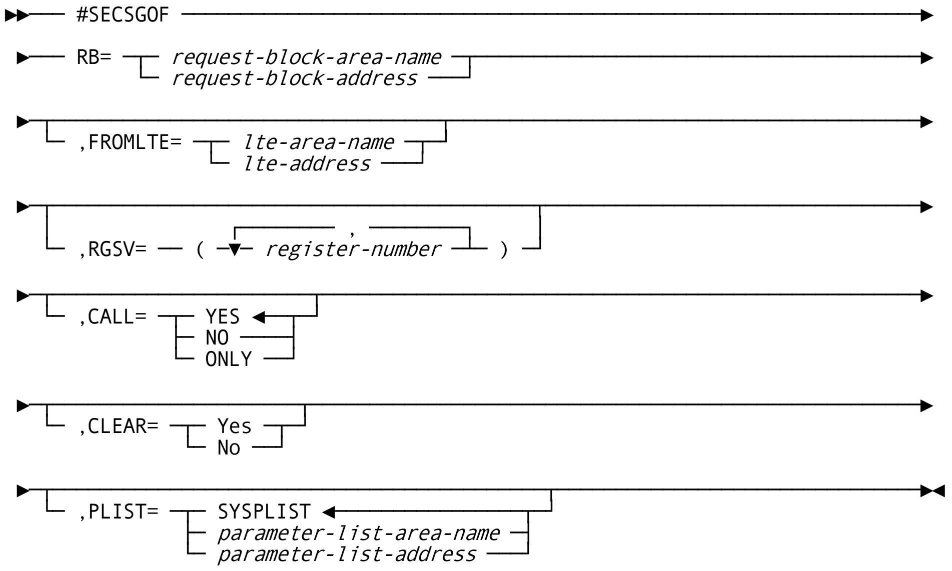
#SECSGOF

#SECSGOF signs the user off the DC system.

#SECSGOF Authorization

Authorization requirements for use of #SECSGOF, if any, are maintained in the external security system.

#SECSGOF Syntax



#SECSGOF Parameters

RB=

Specifies the area containing the Security Request Block (SRB) associated with this request. This area is mapped by #SECRB.

The location of the SRB is required because most of the other parameters coded on #SECSGOF will be stored in the SRB.

This parameter is not required if CALL=ONLY.

request-block-area-name

Identifies the symbolic name of the area containing the SRB.

request-block-address

Identifies the register containing the address of the SRB.

,FROMLTE=

Specifies the address of the Logical Terminal Element (LTE) for which signoff will be done.

If you do not specify FROMLTE=, signoff processing will use the task's primary LTE (that is, TCELTEA).

lte-area-name

Identifies the symbolic name of a user-defined area containing the LTE address.

lte-address

Identifies the register containing the LTE address.

,RGSV=

Specifies that one or more registers are to be saved across the call. This parameter is valid in system mode only.

register-number

Specifies a register.

Register-number must be a numeric literal.

Register-number can be specified in the form R*n*, where *n* is a numeric literal. This assumes that the symbol R*n* has been equated to the corresponding register number; for example:

```
R0  EQU  0
R1  EQU  1
R2  EQU  2
 .
 .
 .
R15 EQU  15
```

These assignments can be made with the macro #REGEQU or explicitly coded in the program.

,CALL=

Controls the expansion of the #SECSGOF macro.

If you omit the CALL parameter, the effect is the same as specifying CALL=YES.

YES

Causes #SECSGOF to generate both the code to complete the SRB and the code to invoke the security system.

NO

Causes #SECSGOF to generate the code to fill in the SRB fields, but not to build the parameter list or the call.

ONLY

Causes #SECSGOF to generate only the code needed to invoke the security system.

,CLEAR=

Specifies whether you want the SRB to be initialized.

If you omit the CLEAR= parameter, the effect is the same as specifying CLEAR=YES. Exception: If CALL=ONLY, the CLEAR parameter defaults to NO.

Yes

Causes #SECSGOF to clear the SRB to binary zeros before the macro expansion begins to assign values.

No

Indicates that the SRB should not be initialized.

,PLIST=

Specifies the area in which to build the parameter list. The parameter list is a three-fullword storage area.

SYSPLIST

Supplies the default name for the area that contains the parameter list.

parameter-list-area-name

Identifies the symbolic name of a user-defined fullword-aligned field that contains the parameter list.

parameter-list-address

Identifies the register containing the address of the parameter list.

#SECSGOF Usage

Substituting for RHDCSNOF

You can issue a #SECSGOF request instead of linking to RHDCSNOF. However, only RHDCSNOF frees all resources associated with the user session.

Copying the Security Request Block

To issue the #SECSGOF macro, you must copy the Security Request Block. This block is mapped by the #SECRB DSECT.

Note: For more information, see the documentation of #SECRB (see page 428).

#SECSGOF Return Codes

The return code for a security check is stored in register 15 and the SRBXR15 field of the SRB. The following table lists the possible return codes provided by the security system in response to a #SECSGOF macro:

Code Code	Meaning Meaning
00	Request was successful
12	Interface/parameter list error

#SECSGON

#SECSGON initiates a signon to the CA IDMS system and attempts to authenticate the user and validate the password.

#SECSGON Authorization

Authorization requirements for use of #SECSGON, if any, are maintained in the external security system.

#SECSGON Syntax

```
►►── #SECSGON ─────────────────────────────────────────────────────────────►

►── RB= ─┬─ request-block-area-name ─┬─────────────────────────────────────►
         └─ request-block-address ───┘

►── ,USERID= ─┬─ user-identifier-area-name ─┬──────────────────────────────►
              ├─ user-identifier-address ───┤
              └─ 'user-identifier' ─────────┘

►── ,PASSWRD= ─┬─ password-area-name ─┬────────────────────────────────────►
               ├─ password-address ───┤
               └─ 'password' ─────────┘

►─┬──────────────────────────────────────────────┬─────────────────────────►
  └─ ,NEWPASS= ─┬─ new-password-area-name ─┬─────┘
                ├─ new-password-address ───┤
                └─ 'new-password' ─────────┘

►─┬─────────────────────────────────┬──────────────────────────────────────►
  └─ ,PASSTYP= ─┬─ CLEAR ◄──┬───────┘
                └─ ENCRYPT ─┘

►─┬────────────────────────────┬───────────────────────────────────────────►
  └─ ,SUPOFFM= ─┬─ YES ──┬─────┘
                └─ NO ◄──┘

►─┬────────────────────────────┬───────────────────────────────────────────►
  └─ ,PASSCHK= ─┬─ YES ◄─┬─────┘
                └─ NO ───┘

►─┬───────────────────────────────────┬────────────────────────────────────►
  └─ ,TOLTE= ─┬─ lte-area-name ─┬─────┘
              └─ lte-address ───┘

►─┬─────────────────────────────────────────────┬──────────────────────────►
  │                 ┌────────── , ─────────┐     │
  └─ ,RGSV= ── ( ─▼─ register-number ─┴─ ) ─┘

►─┬──────────────────────────┬─────────────────────────────────────────────►
  └─ ,CALL= ─┬─ YES ◄─┬──────┘
             ├─ NO ───┤
             └─ ONLY ─┘

►─┬──────────────────────────┬─────────────────────────────────────────────►
  └─ ,CLEAR= ─┬─ Yes ─┬──────┘
              └─ No ──┘

►─┬──────────────────────────────────────────────┬────────────────────────►◄
  └─ ,PLIST= ─┬─ SYSPLIST ◄──────────────────────┤
              ├─ parameter-list-area-name ─┤
              └─ parameter-list-address ───┘
```

#SECSGON Parameters

RB=

Specifies the area containing the Security Request Block (SRB) associated with this request. This area is mapped by #SECRB.

The location of the SRB is required because most of the other parameters coded on #SECSGON will be stored in the SRB.

This parameter is not required if CALL=ONLY.

request-block-area-name

Identifies the symbolic name of the area containing the SRB.

request-block-address

Identifies the register containing the address of the SRB.

,USERID=

Specifies the area containing the user ID being signed on.

The user ID must be 1 to 18 characters. If less than 18 characters, it must be left-justified and padded with spaces or binary zeros.

user-identifier-area-name

Identifies the symbolic name of the area containing the user ID.

user-identifier-address

Identifies the register containing the address of the user ID.

'*user-identifier*'

Supplies the user ID as a literal character string.

,PASSWRD=

Specifies the area containing the password.

The password must be one to eight characters. If less than eight characters, it must be left-justified and padded with spaces or binary zeros.

PASSWRD is ignored if PASSCHK=NO is specified.

password-area-name

Identifies the symbolic name of the area containing the password.

password-address

Identifies the register containing the address of the password.

'*password*'

Supplies the password as a literal character string.

,NEWPASS=

Specifies the area containing the new password.

The password must be one to eight characters. If less than eight characters, it must be left-justified and padded with spaces or binary zeros.

NEWPASS is ignored if PASSCHK=NO is specified.

new-password-area-name

Identifies the symbolic name of the area containing the password.

new-password-address

Identifies the register containing the address of the password.

'*new-password*'

Supplies the password as a literal character string.

,PASSTYP=

Specifies whether the value supplied for PASSWRD is encrypted or in clear text.

If you omit PASSTYP and specify CLEAR=YES, the effect is the same as coding PASSTYP=CLEAR. If you omit PASSTYP and specify CLEAR=NO, PASSTYP retains its prior setting.

CLEAR

Specifies that the password is not encrypted. CA IDMS will encrypt the password if signon security is internal.

ENCRYPT

Specifies that the password supplied on this request is already encrypted.

,SUPOFFM=

Specifies whether a message should be issued if a user is currently signed on to the terminal associated with this request and must be signed off.

If you omit SUPOFFM and specify CLEAR=YES, the effect is the same as specifying SUPOFFM=NO. If you omit SUPOFFM and specify CLEAR=NO, SUPOFFM retains its prior setting.

YES

Specifies that the signoff message should be suppressed.

NO

Specifies that the signoff message should be issued.

,PASSCHK=

Indicates whether password validation should be performed as part of this signon.

Password validation may be bypassed only if the requestor is in system mode. If a user-mode program issues a #SECSGON with PASSCHK=NO, the request will be rejected as invalid (return code 12). If you omit PASSCHK and specify CLEAR=YES, the effect is the same as specifying PASSCHK=YES. If you omit PASSCHK and specify CLEAR=NO, PASSCHK retains its prior setting.

YES

Specifies that password validation should be performed as part of this request.

NO

Specifies that no password validation should be performed as part of this request.

,TOLTE=

Specifies the address of the Logical Terminal Element (LTE) for which signon will be done.

If you do not specify TOLTE=, signon processing will use the task's primary LTE (that is, TCELTEA).

lte-area-name

Identifies the symbolic name of a user-defined area containing the LTE address.

lte-address

Identifies the register containing the LTE address.

,RGSV=

Specifies that one or more registers are to be saved across the call. This parameter is valid in system mode only.

register-number

Specifies a register.

Register-number must be a numeric literal.

Register-number can be specified in the form R*n*, where *n* is a numeric literal. This assumes that the symbol R*n* has been equated to the corresponding register number; for example:

```
R0  EQU  0
R1  EQU  1
R2  EQU  2
 .
 .
 .
R15 EQU  15
```

These assignments can be made with the macro #REGEQU or explicitly coded in the program.

,CALL=

Controls the expansion of the #SECSGON macro.

If you omit the CALL parameter, the effect is the same as specifying CALL=YES.

YES

Causes #SECSGON to generate both the code to complete the SRB and the code to invoke the security system.

NO

Causes #SECSGON to generate the code to fill in the SRB fields, but not to build the parameter list or the call.

ONLY

Causes #SECSGON to generate only the code needed to invoke the security system.

,CLEAR=

Specifies whether you want the SRB to be initialized.

If you omit the CLEAR= parameter, the effect is the same as specifying CLEAR=YES. Exception: If CALL=ONLY, the CLEAR parameter defaults to NO.

Yes

Causes #SECSGON to clear the SRB to binary zeros before the macro expansion begins to assign values.

No

Indicates that the SRB should not be initialized.

,PLIST=

Specifies the area in which to build the parameter list. The parameter list is a three-fullword storage area.

SYSPLIST

Supplies the default name for the area that contains the parameter list.

parameter-list-area-name

Identifies the symbolic name of a user-defined fullword-aligned field that contains the parameter list.

parameter-list-address

Identifies the register containing the address of the parameter list.

#SECSGON Usage

Substituting for RHDCSNON

All processing performed by RHDCSNON, including user profile processing, is also performed by #SECSGON, except:

- Signon CLIST processing
- Issuing the message DC258003, USER IS SIGNED ON

Copying the Security Request Block

To issue the #SECSGON macro, you must copy the Security Request Block. This block is mapped by the #SECRB DSECT.

Note: For more information, see the documentation of #SECRB (see page 428).

#SECSGON Return Codes

The return code for a signon request is stored in register 15 and the SRBXR15 field of the SRB. The following table lists the possible return codes provided by the security system in response to a #SECSGON macro:

00	Request was successful; access allowed
04	User identifier not found
08	User not authorized; access denied
12	Interface/parameter list error
16	Password missing or invalid
20	Password has expired (this condition can result only if signon is externally secured)

Note: If MULTIPLE signon is enabled, and a user attempts to change the password while that user is signed on to another terminal, SRBXR15 will return a value of 8 (ACCESS DENIED) but signon will complete successfully. SRB field SRBXR0 contains reason code SRBXNNPW (password cannot be changed).

Chapter 12: Notes on Security Statement Syntax

This section contains the following topics:

About Authorization

What is Authorization

You can execute a security statement if you have the required authorization. The required authorization is expressed in terms of the privilege or privileges a user must hold. This appears immediately preceding the syntax diagram for each security statement documented in the following chapters.

Holders of SYSADMIN

A user who holds SYSADMIN privilege is authorized to execute all security statements. SYSADMIN is identified as the required authorization in documentation of security statement syntax when it is the *only* privilege that qualifies the user to execute the statement.

Resource Identifiers

What is an Identifier

Identifiers are the smallest lexical units used to name resources in the CA IDMS environment.

The following are examples of identifiers:

- *Area-name*
- *User-identifier*
- *Task-identifier*

Qualifying Identifiers

Identifiers for some resources may need to be qualified by other identifiers. For example, an area name is always qualified by the name of the segment with which it is associated:

```
segment-name.area-name
```

Syntax diagrams in the chapters that follow indicate when an identifier may be qualified and whether the qualifier is required or optional.

Forming Identifiers

Valid Characters

An identifier consists of a combination of the following:

- Letters (A through Z and a through z)—See Delimited Identifiers (see page 227) for information about the significance of lowercase and uppercase letters
- Digits (0 through 9)
- At sign (@)
- Dollar sign ($)
- Pound sign (#)
- Underscore (_)

The first character of an identifier must be a letter, @, $, or #.

Maximum Length

The maximum length of a given identifier is presented in Syntactic Limits (see page 230).

Delimited Identifiers

Why Delimit Identifiers

You delimit an identifier in double quotation marks to do the following:

- **Allow the use of special characters and blanks**. An identifier enclosed in quotation marks can consist of any combination of characters. For example, this is a valid identifier:

  ```
  "&ATM*F(0517). MA"
  ```

  To include a double quotation mark as part of the identifier itself, use two consecutive double quotation marks. For example:

  ```
  "M1K""L9&ZZ".
  ```

- **Make case significant**. When you enclose an identifier in quotation marks, CA IDMS does not convert lowercase letters to uppercase.

  Lowercase letters in quotation marks are not equal to uppercase letters or to lowercase letters that are not in quotation marks. In the following example, the identifiers on the left all identify the same table; the identifier on the right identifies a different table:

  ```
  employee        "employee"
  EMPLOYEE
  "EMPLOYEE"
  ```

Placement of Quotation Marks

If one or more parts of a qualified identifier require quotation marks, place the quotation marks only around the individual parts. Do not include two identifiers in one set of quotation marks. For example, both parts of the following qualified identifier require quotation marks:

```
"temp-tab-1"."Commission to Date"
```

When you calculate the length of an identifier, do not include delimiting quotation marks.

Note: The delimiter rules also apply to passwords using special characters or blanks even though passwords are not delimiters.

Expansion of Table-Name

Represents a qualified or unqualified table, view, function, procedure, or table procedure identifier in an SQL statement.

Expansion of Table-Name Syntax

Expansion of table-name

```
►►─┬────────────────┬─┬─ table-identifier ─────────────┬──────────────────►◄
   └─ schema-name. ─┘ ├─ view-identifier ──────────────┤
                      ├─ procedure-identifier ─────────┤
                      └─ table-procedure-identifier ───┘
```

Expansion of Table-Name Parameters

schema-name

Specifies the schema with which the table, view, procedure or table procedure identified by table-identifier, view-identifier, procedure-identifier, or table-procedure-identifier is associated.

table-identifier

Identifies either a base table defined in the dictionary or a temporary table defined during the current transaction.

view-identifier

Identifies a view defined in the dictionary.

procedure-identifier

Identifies a procedure defined in the dictionary.

function-identifier

Identifies a function defined in the dictionary.

table-procedure-identifier

Identifies a table procedure defined in the dictionary.

Expansion of Authorization Identifier

Represents a user identifier or a group identifier in an authorization statement.

Expansion of Authorization Identifier Syntax

```
►►─┬─ user-identifier ──┬──────────────────────────────────────────────────►◄
   └─ group-identifier ─┘
```

Expansion of Authorization Identifier Parameters

user-identifier

Identifies a user.

group-identifier

Identifies a group.

Expansion of Authorization Identifier Usage

User Catalog Definition

The CREATE/ALTER/DROP statements for USER manipulate the definitions of *user-identifier* in the user catalog. Similarly, the CREATE/ALTER/DROP statements for GROUP manipulate the definitions of *group-identifier* in the user catalog.

When you specify **authorization-identifier** in any other security statement (or in the ADD USER parameter of CREATE/ALTER GROUP), **authorization-identifier** must be defined in the user catalog *if* either of these is true:

- The identifier is a *user-identifier* and security on the specified resource is enforced by CA IDMS internal security.
- The identifier is a *group-identifier*.

If authentication of users is handled by an external security system, *user-identifier* need not be defined in the user catalog.

Authorizing a User to Update a Table

In the following GRANT statement, the authorization identifier is the user identifier RES:

```
grant update
  on table employee
  to res;
```

Revoking Execution Privileges from a Group

In the following GRANT statement, the authorization identifier is the group identifier ACCT_GRP_1:

```
revoke execute
  on category emp_update
  from acct_grp_1;
```

Syntactic Limits

This table lists the maximum size or value, as applicable, of parameters you use in security syntax statements:

Item	Maximum
user-identifier	18 characters
group-identifier	18 characters
Description of a user or group	40 characters
User *password*	8 characters
User-name	32 characters
Profile-name	18 characters
Attribute-keyword	8 characters
Attribute-value	32 characters
Application-name	8 characters
Category-name	32 characters
Activity-name	18 characters
System-name	8 characters
Activity-number	256 (minimum = 1)

Chapter 13: Syntax for Securing Global Resources

This section contains the following topics:

ALTER GROUP

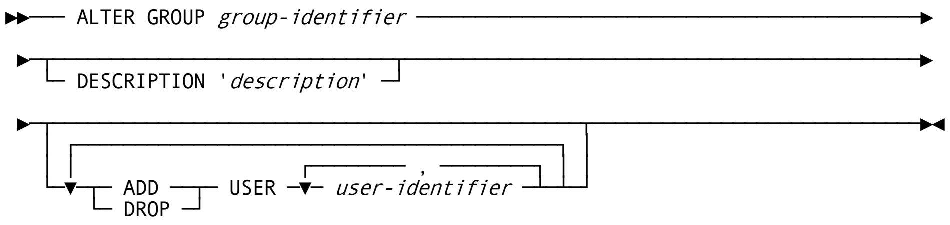
Modifies the definition of a group by adding users, dropping users, or changing the description of the group.

ALTER GROUP Authorization

To issue an ALTER GROUP statement, you must hold one of the following privileges:

- SYSADMIN
- ALTER privilege on the group

ALTER GROUP Syntax



ALTER GROUP Parameters

group-identifier

Identifies the group to be modified.

Group-identifier must be a group that has been defined in the user catalog with the CREATE GROUP statement.

DESCRIPTION '*description*'

Supplies a description of *group-identifier*.

Description can be at most 40 characters in length.

ADD

Adds the specified users to *group-identifier*.

DROP

Drops the specified users from *group-identifier*.

USER *user-identifier*

Identifies the user to be added to or dropped from *group-identifier*.

User-identifier must be a user that has been defined in the user catalog with the CREATE USER statement.

ALTER GROUP Usage

No Nesting of Groups

You cannot add a group to another group. You can add only users to a group.

Adding Users to a Group

The following ALTER GROUP statement adds three users to the hr_corp group:

```
alter group hr_corp
  add user sam, flo, guy;
```

Dropping a User from a Group

In the following ALTER GROUP statement, user sue is dropped from the hr_corp group:

```
alter group hr_corp
  drop user sue;
```

ALTER GROUP More Information

- For more information about groups, see CREATE GROUP (see page 238) and DROP GROUP (see page 245).
- For more information about creating a user, see CREATE USER (see page 239).
- For more information about the ALTER privilege, see GRANT Definition Privileges (see page 252).

ALTER USER

Modifies the definition of a user in the user catalog.

ALTER USER Authorization

To issue an ALTER USER statement, you must hold one of the following privileges:

- SYSADMIN
- ALTER privilege on the user

ALTER USER Syntax

```
►►─── ALTER USER user-identifier ─────────────────────────────────────────────►

 ►─┬────────────────────────────┬────────────────────────────────────────────►
   └─ DESCRIPTION 'description' ┘

 ►─┬──────────────────────────────────┬──────────────────────────────────────►
   └─ GROUP ─┬─ PUBLIC ───────────────┤
             └─ group-identifier ─────┘

 ►─┬──────────────────────┬──────────────────────────────────────────────────►
   └─ NAME 'user-name' ───┘

 ►─┬──────────────────────────────┬──────────────────────────────────────────►
   └─ PASSWORD ─┬─ password ─┬────┘
                └─ NULL ─────┘

 ►─┬─────────────────────────────────┬───────────────────────────────────────►◄
   └─ PROFILE ─┬─ profile-name ─┬────┘
               └─ NULL ─────────┘
```

ALTER USER Parameters

user-identifier

Identifies the user to be modified.

User-identifier must be a user that has been defined in the user catalog with the CREATE USER statement.

DESCRIPTION '*description*'

Supplies a description of the user.

Description can be at most 40 characters in length.

GROUP

Specifies the default group for *user-identifier*.

PUBLIC

Identifies the group PUBLIC.

All users automatically belong to group PUBLIC.

group-identifier

Identifies a group.

Group-identifier must be a group that has been defined in the user catalog with the CREATE GROUP statement.

NAME '*user-name*'

Supplies the name associated with *user-identifier*.

User-name can be at most 32 characters in length. This parameter is available for documentation; its value is not used in CA IDMS security processing.

PASSWORD *password*

Supplies the password associated with *user-identifier*.

Password can be at most eight characters in length.

NULL

Removes the password defined for *user-identifier* in the user catalog.

PROFILE

Specifies the user profile associated with the *user-identifier*.

profile-name

Identifies a user profile.

NULL

Removes the profile associated with *user-identifier*.

If NULL is specified, no user profile attributes will be established when the user executes CA IDMS software.

ALTER USER Usage

Null Password

If *password* is NULL and CA IDMS internal security is used to validate users, the user will be able to sign on to a system without supplying a password.

Changing the Default Group and User Profile

The following ALTER USER statement specifies a new default group and user profile for user sue:

```
alter user sue
  group hr_corp
  profile hr_prof;
```

ALTER USER More Information

- For more information **about users**, see Create User (see page 239) and DROP USER (see page 247).
- For more information **about user profiles**, see CREATE USER PROFILE (see page 242).
- For more information **about the ALTER privilege**, see GRANT Definition Privileges (see page 252).

ALTER USER PROFILE

Modifies the definition of a user profile in the user catalog.

ALTER USER PROFILE Authorization

To issue an ALTER USER PROFILE statement, you must hold one of the following privileges:

- SYSADMIN
- ALTER privilege on the user profile

ALTER USER PROFILE Syntax

```
►►─── ALTER USER PROFILE profile-name ─────────────────────────────────────►

                   ┌────────── , ──────────┐
 ►─── ATTRIBUTEs ─▼─ attribute-specification ─┴───────────────────────────►◄
```

Expansion of Attribute-Specification

```
►►─ attribute-keyword = ' ─┬─┬──────────┬─┬─ &USER. ───┬─┬──────────┬─┬─ ' ─►
                           │ └─ prefix ─┘ ├─ &GROUP. ──┤ └─ suffix ─┘ │
                           │              └─ &SYSTEM. ─┘              │
                           ├─ attribute-value ────────────────────────┤
                           └─ NULL ───────────────────────────────────┘

►─┬────────────────────────────┬──────────────────────────────────────────►◄
  └─ OVERRIDE ─┬─ YES ─┬───────┘
               └─ NO ──┘
```

ALTER USER PROFILE Parameters

profile-name

Identifies the profile to be modified.

Profile-name must be a profile that has been defined in the user catalog with the CREATE USER PROFILE statement.

ATTRIBUTEs attribute-specification

Supplies one or more attributes of the profile.

An attribute is a keyword and an associated value for the keyword.

***attribute-keyword* =**

Specifies the attribute keyword whose value is to be added, replaced, or removed.

An identifier of not more than eight characters can be used as an attribute keyword.

Note: Certain keywords have special significance to the CA IDMS runtime environment. For a list of these keywords and discussion of their meaning, see chapter "System Profiles" in the *CA IDMS System Tasks and Operator Commands Guide*.

&USER

Supplies the attribute value. It is a substitution parameter.

The value of &USER is equal to the user ID of the user on whose behalf the user profile is invoked.

&GROUP

Represents the current group. It is a substitution parameter. The value of &GROUP is equal to the name of the default group for the current user.

&SYSTEM

Supplies the attribute value. It is a substitution parameter.

The value of &SYSTEM is equal to the SYSTEM ID value for the current system in the SYSTEM statement of system generation, or 'BATCH' in the case of local mode execution.

prefix

Supplies a prefix for the value in the substitution parameter.

suffix

Supplies a suffix for the value in the substitution parameter.

attribute-value

Provides the value portion of the attribute specification.

Attribute-value may be at most 32 characters in length and must be enclosed in single quotation marks if it contains embedded blanks or special characters other than @, $, and #.

NULL

Removes the attribute from *profile-name*.

OVERRIDE

Specifies whether the user can modify the attribute specification with a DCUF SET PROFILE command.

YES allows the user to override the attribute specification. NO prevents the user from overriding the attribute specification.

If OVERRIDE is not specified and the attribute keyword already exists in the profile, the OVERRIDE value remains the same. If OVERRIDE is not specified and the attribute keyword does not exist in the profile, the OVERRIDE value defaults to YES.

ALTER USER PROFILE Usage

Nesting Profiles

The attribute keyword INCLUDE specifies that *attribute-value* identifies another profile. The attributes of the included, or nested, profile are added to those of the current profile.

Up to 10 levels of nesting are supported.

Substitution Parameters

The value of a substitution parameter in *attribute-specification*:

- Must not exceed 32 characters, including a prefix and suffix
- Must not contain special characters other than @, $, and #

The following ALTER USER PROFILE statement defines two attributes in an existing profile:

```
alter user profile corp_dev
  attributes
    dept='5621',
    prtdest='wwuav';
```

ALTER USER PROFILE More Information

- For more information **about creating a user profile**, see CREATE USER PROFILE (see page 242).
- For more information **about dropping a user profile**, see DROP USER PROFILE (see page 249).
- For more information **about attributes**, see chapter "System Profiles" in the *CA IDMS System Tasks and Operator Commands Guide*.

CREATE GROUP

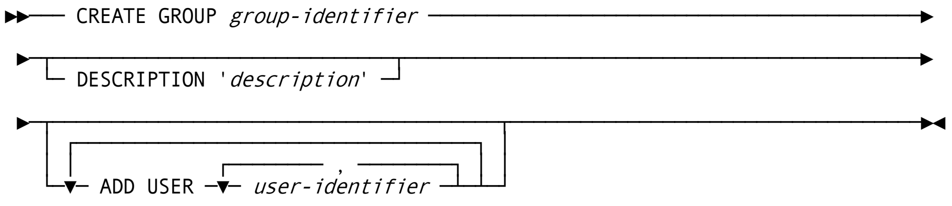
Creates the definition of a group of users.

CREATE GROUP Authorization

To issue a CREATE GROUP statement, you must hold one of the following privileges:

- SYSADMIN
- CREATE privilege on the group

CREATE GROUP Syntax



CREATE GROUP Parameters

group-identifier

Identifies the group to be created.

Group-identifier can be at most 18 characters in length. It must be unique among the set of authorization identifiers (users and groups) in the user catalog.

Note: For more information about identifiers, see chapter Notes on Security Statement Syntax (see page 225).

DESCRIPTION '*description*'

Supplies a description of the group.

Description can be at most 40 characters in length. This parameter is available for documentation; its value is not used in CA IDMS security processing.

ADD USER *user-identifier*

Adds the specified user to *group-identifier*.

Note: For more information about identifiers, see chapter Notes on Security Statement Syntax (see page 225).

CREATE GROUP Usage

No Nesting of Groups

You cannot add a group to another group. You can only add users to a group.

This statement creates a group of users for data communications administrators:

```
create group dca_group
  description 'Authorization group for DCAs'
  add user dca_s15, dca_sys16, dca_sys17, dca_sys18, dca_sys19;
```

CREATE GROUP More Information

- For more information about groups, see ALTER GROUP (see page 231) and DROP GROUP (see page 245).
- For more information about creating a user, see CREATE USER (see page 239).
- For more information about the CREATE privilege, see GRANT Definition Privileges (see page 252).

CREATE USER

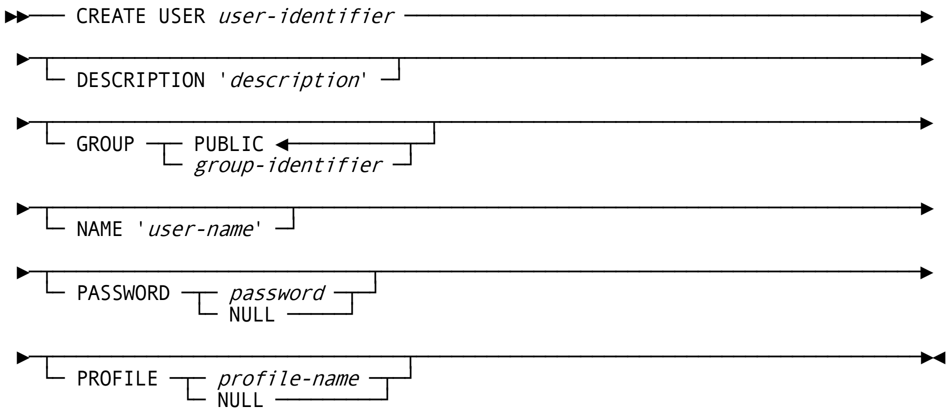
Creates the definition of a user in the user catalog.

CREATE USER Authorization

To issue a CREATE USER statement, you must hold one of the following privileges:

- SYSADMIN
- CREATE privilege on the user

CREATE USER Syntax



CREATE USER Parameters

user-identifier

Identifies the user to be created.

User-identifier can be at most 18 characters in length. It must be unique among the set of authorization identifiers (users and groups) in the user catalog.

Note: For more information about identifiers, see Notes on Security Statement Syntax (see page 225).

DESCRIPTION '*description*'

Supplies a description of the user.

Description can be at most 40 characters in length. This parameter is available for documentation; its value is not used in CA IDMS security processing.

GROUP

Specifies the default group of *user-identifier*.

PUBLIC

Identifies the group PUBLIC.

group-identifier

Identifies a group.

If *group-identifier* has been defined in the user catalog, the user is added to the group. If *group-identifier* has not been defined in the user catalog, an error message is issued and the user is not added to the group.

Note: For more information about identifiers, see Notes on Security Statement Syntax (see page 225).

NAME '*user-name*'

Supplies a name to be associated with *user-identifier* in the user catalog.

User-name can be at most 32 characters in length. This parameter is available for documentation; its value is not used in CA IDMS security processing.

PASSWORD *password*

Supplies a password for *user-identifier*.

Password may be at most eight characters in length.

NULL

Specifies that no password is defined for *user-identifier* in the user catalog.

PROFILE

Specifies a user profile to be associated with *user-identifier*.

profile-name

Identifies a user profile.

Profile-name must be a user profile defined in the user catalog with the CREATE USER PROFILE statement.

NULL

Specifies that no profile is associated with *user-identifier*.

If NULL is specified, no user profile attributes will be established when the user executes CA IDMS software.

CREATE USER Usage

Null Password

If *password* is NULL and CA IDMS internal security is used to validate users, the user will be able to sign on to a system without supplying a password.

Creating a Named User

This statement creates a user ID for an individual:

```
create user tim
  name 'Thomas McNall'
  password qwerty
  profile dev_prof;
```

Creating a Generic User

This statement creates a user ID for a given role:

```
create user dca_s18
  description 'Administrator for System 18'
  group dca_group
  password s18pass;
```

CREATE USER More Information

- For more information about users, see ALTER USER (see page 233) and DROP USER (see page 247).
- For more information about user profiles, see CREATE USER PROFILE (see page 242).
- For more information about the CREATE privilege, see GRANT Definition Privileges (see page 252).

CREATE USER PROFILE

Creates the definition of a user profile in the user catalog.

CREATE USER PROFILE Authorization

To issue a CREATE USER PROFILE statement, you must hold one of the following privileges:

- SYSADMIN
- CREATE privilege on the user profile

CREATE USER PROFILE Syntax

```
►►── CREATE USER PROFILE profile-name ──────────────────────────────────►

►─┬─────────────────────────────────────────────────────────────────┬──►◄
  │              ┌────────────── , ──────────────┐                  │
  └─ ATTRIBUTEs ─▼─ attribute-specification ─────┴──────────────────┘
```

Expansion of Attribute-Specification

```
►►── attribute-keyword = ' ─┬─┬────────────┬─┬─ &USER. ───┬─┬────────────┬─┬─ ' ──►
                            │ └─ prefix ───┘ ├─ &GROUP. ──┤ └─ suffix ───┘ │
                            │                └─ &SYSTEM. ─┘                │
                            └─ attribute-value ────────────────────────────┘

►─┬──────────────────────────────┬─────────────────────────────────────►◄
  └─ OVERRIDE ─┬─ YES ◄─┬────────┘
               └─ NO ───┘
```

CREATE USER PROFILE Parameters

profile-name

Identifies the profile to be created.

Profile-name can be at most 18 characters in length.

ATTRIBUTEs attribute-specification

Specifies one or more attributes of the profile.

An attribute is a keyword and an associated value for the keyword.

***attribute-keyword* =**

Specifies the attribute keyword.

An identifier of not more than eight characters can be used as an attribute keyword.

Note: Certain keywords have special significance to the CA IDMS runtime environment. For a list of these keywords and discussion of their meaning, see the chapter "System Profiles" in the *CA IDMS System Tasks and Operator Commands Guide*.

&USER

Is a substitution parameter that supplies the attribute value.

The value of &USER is equal to the user ID of the user on whose behalf the user profile is invoked.

&GROUP

Is a substitution parameter representing the current group. The value of &GROUP is equal to the name of the default group for the current user.

&SYSTEM

Is a substitution parameter that supplies the attribute value.

The value of &SYSTEM is equal to the SYSTEM ID value for the current system in the SYSTEM statement of system generation, or 'BATCH' in the case of local mode execution.

prefix

Supplies a prefix for the value in the substitution parameter.

suffix

Supplies a suffix for the value in the substitution parameter.

attribute-value

Supplies the value portion of the attribute specification.

Attribute-value may be at most 32 characters in length and must be enclosed in single quotation marks if it contains embedded blanks or special characters other than @,$, and #.

OVERRIDE

Specifies whether the user can modify the attribute specification with a DCUF SET PROFILE command.

YES allows the user to override the attribute specification. NO prevents the user from overriding the attribute specification.

CREATE USER PROFILE Usage

Nesting Profiles

The attribute keyword INCLUDE specifies that *attribute-value* identifies another profile. The attributes of the included, or nested, profile are added to those of the current profile.

Up to 10 levels of nesting are supported.

Substitution Parameters

The value of a substitution parameter in *attribute-specification*:

- Must not exceed 32 characters, including a prefix and suffix
- Must not contain special characters other than @, $, and #

The following CREATE USER PROFILE statement creates a profile with two attributes:

```
create user profile corp_dev
 attributes
   dept='1237',
   prtdest='gdnc03';
```

CREATE USER PROFILE More Information

- For more information about altering a user profile, see ALTER USER PROFILE (see page 235).
- For more information about dropping a user profile, see DROP USER PROFILE (see page 249).
- For more information about attributes, see chapter "System Profiles" in the *CA IDMS System Tasks and Operator Commands Guide*.

DROP GROUP

Deletes the definition of a group from the user catalog.

DROP GROUP Authorization

To issue a DROP GROUP statement, you must hold one of the following privileges:

- SYSADMIN
- DROP privilege on the group

DROP GROUP Syntax

```
►►─── DROP GROUP group-identifier ─────────────────────────────────────────►◄
```

DROP GROUP Parameters

group-identifier

Identifies the group to be dropped.

Group-identifier must be a group that has been defined in the user catalog with the CREATE GROUP statement.

DROP GROUP Usage

Group PUBLIC

You cannot drop the group PUBLIC.

Implicitly Revoking Privileges

When you drop a group, you automatically revoke all privileges that have been granted to the group. Thus, a user who was in the dropped group no longer holds privileges received as a result of membership in the group.

Issuing DROP GROUP Before and After SDEL Execution

The first time you issue a DROP GROUP statement for a group identifier, the identifier is flagged for logical deletion. To delete all privileges associated with each logically deleted group, you execute the SDEL task in each system of the domain and against each dictionary in the system that contains security definitions. (The system dictionary and each application dictionary with an SQL catalog component contain security definitions.) Then you reissue the DROP GROUP statement to physically delete the group from the user catalog.

Note: If the SDEL task has been defined as an autotask and is invoked at startup, all dictionaries are processed if a logically deleted group is found. If SDEL is invoked manually, each dictionary must be processed separately.

For more information about the SDEL task, see the *CA IDMS System Tasks and Operator Commands Guide*.

If you physically delete a group before running SDEL on all systems, follow the following steps:

1. Create the group again.
2. Drop the group once.
3. Run SDEL on all systems and dictionaries.
4. Drop the group a second time.

The following DROP GROUP statement drops the definition of the corp_admin group and implicitly revokes all privileges that have been granted to the group:

```
drop group corp_admin;
```

DROP GROUP More Information

- For more information about creating and altering a group, see CREATE GROUP (see page 238) and Alter Group.
- For more information about granting and revoking privileges, see descriptions of GRANT and REVOKE statements in the following sections:
  - GRANT Definition Privileges (see page 252)
  - REVOKE Definition Privileges (see page 256)
  - Syntax for Securing System Resources (see page 261)
  - Syntax for Securing Database Resources (see page 295)

DROP USER

Deletes the definition of a user from the user catalog and drops the user from all groups of which the user is a member.

To complete a physical deletion from the user catalog, you must issue the DROP USER statement two times, as described in the "Usage" section following the parameter descriptions.

DROP USER Authorization

To issue a DROP USER statement, you must hold one of the following privileges:

- SYSADMIN
- DROP privilege on the user

DROP USER Syntax

```
►►── DROP USER user-identifier ──────────────────────────────────────────►◄
```

DROP USER Parameters

user-identifier

Identifies the user to be dropped.

User-identifier must be a user that has been defined in the user catalog with the CREATE USER statement.

DROP USER Usage

Issuing DROP USER Before and After SDEL Execution

The first time you issue a DROP USER statement for a user identifier, the identifier is flagged for logical deletion. To delete all privileges associated with each logically deleted user, you execute the SDEL task in each system of the domain and against each dictionary in the system that contains security definitions. (The system dictionary and each application dictionary with an SQL catalog component contain security definitions.) Then you reissue the DROP USER statement to physically delete the user from the user catalog.

Note: If the SDEL task has been defined as an autotask and is invoked at startup, all dictionaries are processed if a logically deleted user is found. If SDEL is invoked manually, each dictionary must be processed separately.

For more information about the SDEL task, see the *CA IDMS System Tasks and Operator Commands Guide*.

If you physically delete a user before running SDEL on all systems, follow these steps:

1. Create the user again.
2. Drop the user once.
3. Run SDEL on all systems and dictionaries.
4. Drop the user a second time.

Implicitly Revoking Privileges

When you drop a user, you implicitly revoke all privileges that have been granted to the user.

The following DROP USER statement removes the definition of user sue from the user catalog and removes user sue from any group to which user sue was assigned:

```
drop user sue;
```

DROP USER More Information

- For more information about creating and altering a user, see CREATE USER (see page 239) and ALTER USER.
- For more information about granting and revoking privileges, see descriptions of GRANT and REVOKE statements in the following sections:
  - GRANT Definition Privileges (see page 252)
  - REVOKE Definition Privileges (see page 256)
  - Syntax for Securing System Resources (see page 261)
  - Syntax for Securing Database Resources (see page 295)

DROP USER PROFILE

Deletes the definition of a user profile from the user catalog.

DROP USER PROFILE Authorization

To issue a DROP USER PROFILE statement, you must hold one of the following privileges:

- SYSADMIN
- DROP privilege on the user profile

DROP USER PROFILE Syntax

```
►►── DROP USER PROFILE profile-name ──────────────────────────────────────►◄
```

DROP USER PROFILE Parameters

profile-name

Identifies the profile to be dropped.

Profile-name must be a profile that has been defined in the user catalog with the CREATE USER PROFILE statement.

DROP USER PROFILE Usage

Users Associated With Profiles That are Dropped

If you drop a profile specified in a user definition, no user profile attributes will be associated with the user session.

The following DROP USER PROFILE statement removes the definition of the hr_prof profile from the user catalog:

```
drop user profile hr_prof;
```

DROP USER PROFILE More Information

- For more information **about creating and altering a user profile**, see CREATE USER PROFILE (see page 242) and ALTER USER PROFILE.
- For more information **about defining users**, see CREATE USER (see page 239) and ALTER USER.

GRANT Administration Privilege

Gives one or more users SYSADMIN privilege.

GRANT Administration Privilege Authorization

To grant SYSADMIN privilege, you must hold SYSADMIN privilege.

GRANT Administration Privilege Syntax

```
►►── GRANT SYSADMIN ──────────────────────────────────────────────►

 ►── TO ─▼┬─ PUBLIC ──────────────── , ────────┬──────────────────►◄
          │┌───────────────────────────────────┘
          └┴─ authorization-identifier ─┘
```

GRANT Administration Privilege Parameters

SYSADMIN

Specifies that you are giving SYSADMIN privilege to the users or groups identified in the TO parameter.

TO

Specifies the users or groups to whom you are giving SYSADMIN privilege.

PUBLIC

Specifies all users.

Important: If you grant SYSADMIN to PUBLIC, any user can administer the security system.

authorization-identifier

Identifies a user or group.

Note: Expanded syntax for **authorization-identifier** is presented in Notes on Security Statement Syntax (see page 225).

GRANT Administration Privilege Usage

Ultimate Authority to Grant and Revoke

SYSADMIN controls access to all resources. A user with SYSADMIN privilege can grant and revoke any privilege on any resource in the domain.

Caution in Granting SYSADMIN

A user to whom you grant SYSADMIN privilege can administer the security system. If you grant SYSADMIN to PUBLIC, any user can administer the security system.

Note: If resource type SYSA is unsecured in the SRTT, any user can administer the security system.

You can decentralize security administration by granting DCADMIN and DBADMIN privileges to users.

Note: For more information about the DCADMIN and DBADMIN administration privileges, see the following sections:

- Securing Database Resources (see page 117)
- Syntax for Securing Database Resources (see page 295)

Granting SYSADMIN to the Security Administrator

The following GRANT statement gives SYSADMIN privilege to group secadmin:

```
grant sysadmin
  to secadmin;
```

GRANT Administration Privilege More Information

For more information **about revoking SYSADMIN privilege**, see REVOKE Administration Privilege (see page 255).

GRANT Definition Privileges

Gives one or more users or groups the privilege of performing definition functions on users, groups, or user profiles in the user catalog.

GRANT Definition Privileges Authorization

To grant a definition privilege on a user catalog resource, you must hold one of the following privileges:

- The corresponding grantable privilege (you can grant the privilege, but you cannot specify WITH GRANT OPTION)
- SYSADMIN privilege

GRANT Definition Privileges Syntax

```
►►── GRANT ─┬─ DEFINE ─────────────┬──────────────────────────────────────►
            │  ┌──────── , ───────┐ │
            └─▼┬─ ALTER ─────┬────┴─┘
               ├─ CREATE ────┤
               ├─ DISPLAY ───┤
               └─ DROP ──────┘

 ►── ON ──┬─ GROUP group-identifier ────┬──────────────────────────────────►
          ├─ USER user-identifier ──────┤
          └─ USER PROFILE profile-name ─┘

          ┌─────────────── , ───────────────┐
 ►── TO -▼┬─ PUBLIC ───────────────────────┬┴─────────────────────────────►
          └─ authorization-identifier ─────┘

 ►─┬──────────────────────┬──────────────────────────────────────────────►◄
   └─ WITH GRANT OPTION ──┘
```

GRANT Definition Privileges Parameters

DEFINE

Gives the ALTER, CREATE, and DROP privileges on the resource identified in the ON parameter to the users or groups identified in the TO parameter.

ALTER

Gives the ALTER privilege on the resource identified in the ON parameter to the users or groups identified in the TO parameter.

The ALTER privilege on a resource allows the user to modify the resource definition.

CREATE

Gives the CREATE privilege on the resource identified in the ON parameter to the users or groups identified in the TO parameter.

The CREATE privilege on a resource allows the user to define the resource.

DISPLAY

Gives the DISPLAY privilege on the resource identified in the ON parameter to the users or groups identified in the TO parameter.

The DISPLAY privilege allows the user to issue a DISPLAY RESOURCE statement on the named resource. The grantable DISPLAY privilege allows a user to issue a DISPLAY PRIVILEGES statement on the named resource.

DROP

Gives the DROP privilege on the resource identified in the ON parameter to the users or groups identified in the TO parameter.

The DROP privilege on a resource allows the user to delete the definition of the resource.

ON

Specifies the resource to which the specified definition privileges apply.

GROUP *group-identifier*

Identifies a group.

You can wildcard *group-identifier*.

Note: For more information, see Using a Wildcard (see page 76).

USER *user-identifier*

Identifies a user.

You can wildcard *user-identifier*.

Note: For more information, see Using a Wildcard (see page 76).

USER PROFILE *profile-name*

Identifies a user profile.

You can wildcard *profile-name*.

Note: For more information, see Using a Wildcard (see page 76).

TO

Specifies the users to whom you are giving definition privileges.

PUBLIC

Specifies all users.

authorization-identifier

Identifies a user or group.

Note: Expanded syntax for **authorization-identifier** is presented in the chapter Notes on Security Statement Syntax (see page 225).

WITH GRANT OPTION

Gives the privilege of granting the specified definition privileges to the users or groups identified in the TO parameter.

A privilege granted with the WITH GRANT OPTION is called a grantable privilege.

GRANT Definition Privileges Usage

Granting Definition Privilege with a Wildcard

By wildcarding the resource name when you grant a definition privilege, you allow a user to define multiple resources that are named with the same beginning characters.

For example, if you grant DEFINE privilege on user profile HR_* to group HR_ADMIN (see the following example), you allow the human resources administrative group to create, alter, and drop profiles that begin with the characters 'HR_'.

The DEFINE Keyword

When you use the DEFINE keyword with a GRANT statement, you grant a set of definition privileges on a resource to one or more users or groups.

When you use the DEFINE keyword with a REVOKE statement, you revoke all definition privileges that have been previously granted on the resource from the specified users or groups.

This means that if you GRANT CREATE privilege on a resource, you can revoke the privilege with either a REVOKE CREATE statement or a REVOKE DEFINE statement. Using REVOKE DEFINE is an efficient technique when you intend to revoke all definition privileges on the resource from a user or group, whether the privileges were granted singly or as a set.

Similarly, you can GRANT DEFINE on a resource to a user and then REVOKE DROP on the resource from the same user. This is an efficient technique for granting all but one definition privilege.

The following GRANT statement gives the HR_ADMIN group the privilege to define user profiles that begin 'HR_':

```
grant define
  on user profile hr_*
  to hr_admin;
```

GRANT Definition Privileges More Information

For more information *about revoking definition privileges*, see REVOKE Administration Privilege (see page 255).

REVOKE Administration Privilege

Revokes SYSADMIN privilege from a user or group.

REVOKE Administration Privilege Authorization

To revoke SYSADMIN privilege, you must hold SYSADMIN privilege.

REVOKE Administration Privilege Syntax

```
►►── REVOKE SYSADMIN ─────────────────────────────────────────────────────────►

              ┌──────────── , ─────────────┐
 ►── FROM ─▼┬─ PUBLIC ───────────────────┬┴──────────────────────────────────►◄
            └─ authorization-identifier ─┘
```

REVOKE Administration Privilege Parameters

SYSADMIN

Specifies that you are revoking SYSADMIN privilege from the users or groups named in the FROM parameter.

FROM

Specifies the users or groups from whom you are revoking SYSADMIN privilege.

PUBLIC

Specifies all users.

authorization-identifier

Identifies a user or group.

Note: Expanded syntax for **authorization-identifier** is presented in the chapter Notes on Security Statement Syntax (see page 225).

REVOKE Administration Privilege Usage

Revoking SYSADMIN from All Users

If the SYSA resource type is secured in the SRTT and SYSADMIN is revoked from all users and groups that hold SYSADMIN privilege, no user has the privilege to grant SYSADMIN and, therefore, no user has overall authority to administer the security system.

In this situation, you must turn off security for the SYSA resource type in the SRTT. With SYSA security off, any user can grant SYSADMIN privilege. Once you have granted the privilege to the appropriate users, you reactivate security for SYSA.

Revoking SYSADMIN from PUBLIC

This statement revokes SYSADMIN privilege from the group PUBLIC:

```
revoke sysadmin
  from public;
```

REVOKE Administration Privilege More Information

- For more information **about grant SYSADMIN privilege**, see GRANT Administration Privilege (see page 250).
- For more information **about the role of the SYSADMIN user**, see the chapter Securing Global Resources (see page 83).

REVOKE Definition Privileges

Revokes from one or more users or groups the privilege of performing definition functions on a user, group, or user profile in the user catalog.

REVOKE Definition Privileges Authorization

To revoke a definition privilege on a global resource, you must hold one of the following privileges:

- SYSADMIN privilege
- The corresponding grantable privilege

REVOKE Definition Privileges Syntax

```
►►─── REVOKE ─┬─ DEFINE ───────────────┬──────────────────────────────────────►
              │  ┌─────── , ───────┐   │
              └─▼─┬─ ALTER ───────┬─┘───┘
                  ├─ CREATE ──────┤
                  ├─ DISPLAY ─────┤
                  └─ DROP ────────┘

►─── ON ─┬─ GROUP group-identifier ───────────┬──────────────────────────────►
         ├─ USER user-identifier ─────────────┤
         └─ USER PROFILE profile-name ────────┘

►─── FROM ─▼─┬─ PUBLIC ───────────────────┬─┘──────────────────────────────►◄
           ┌─┴──────────── , ─────────────┴─┐
             └─ authorization-identifier ─┘
```

REVOKE Definition Privileges Parameters

DEFINE

Revokes the ALTER, CREATE, and DROP privileges on the resource identified in the ON parameter from the users or groups identified in the FROM parameter.

ALTER

Revokes the ALTER privilege on the resource identified in the ON parameter from the users or groups identified in the FROM parameter.

CREATE

Revokes the CREATE privilege on the resource identified in the ON parameter from the users or groups identified in the FROM parameter.

DISPLAY

Revokes the DISPLAY privilege on the resource identified in the ON parameter from the users or groups identified in the FROM parameter.

DROP

Revokes the DROP privilege on the resource identified in the ON parameter from the users or groups identified in the FROM parameter.

ON

Specifies the resource to which the definition privileges apply.

GROUP *group-identifier*

Identifies a group.

USER *user-identifier*

Identifies a user.

USER PROFILE *profile-name*

Identifies a user profile.

FROM

Specifies the users or groups from whom you are revoking definition privileges.

PUBLIC

Specifies all users.

authorization-identifier

Identifies a user or group.

Note: Expanded syntax for **authorization-identifier** is presented in Notes on Security Statement Syntax (see page 225).

REVOKE Definition Privileges Usage

The DEFINE keyword

When you use the DEFINE keyword with a GRANT statement, you grant a set of definition privileges on a resource to one or more users or groups.

When you use the DEFINE keyword with a REVOKE statement, you revoke all definition privileges that have been previously granted on the resource from the specified users or groups.

This means that if you GRANT CREATE privilege on a resource, you can revoke the privilege with either a REVOKE CREATE statement or a REVOKE DEFINE statement. Using REVOKE DEFINE is an efficient technique when you intend to revoke all definition privileges on the resource from a user or group, whether the privileges were granted singly or as a set.

Similarly, you can GRANT DEFINE on a resource to a user and then REVOKE DROP on the resource from the same user. This is an efficient technique for granting all but one definition privilege.

Using a Wildcard When Revoking Definition Privilege

If you use a wildcard in the resource name in a REVOKE statement, the wildcarded name must match the wildcarded name used in a previous GRANT statement. Similarly, if definition privilege was granted on a wildcarded resource name, it can be revoked only by specifying the same name in the REVOKE statement.

For example, assume that this grant has been made:

```
grant define
  on user profile hr*
  to user1, user2, user3;
```

- To revoke the privilege, you must specify HR* in the REVOKE statement.
- If you revoke DEFINE on user profile HRA from USER1, it has no effect on the privilege granted in the preceding example.

The following statement revokes the privilege to define certain user profiles from the hr_corp group:

```
revoke define
  on user profile hr_*
  from hr_corp;
```

REVOKE Definition Privileges More Information

For more information **about granting definition privilege**, see GRANT Definition Privileges (see page 252).

Chapter 14: Syntax for Securing System Resources

This chapter contains the following topics:

This section contains the following topics:

ALTER RESOURCE

Modifies the definition of an activity or a category in the system dictionary.

Authorization

To modify a system, activity, or category definition, you must hold one of these privileges:

- DCADMIN
- SYSADMIN

Syntax

```
►►── ALTER RESOURCE ──────────────────────────────────────────────────────────►

►─┬─ ACTIVITY application-name.activity-name NUMBER activity-number ─┬─────►◄
  └─ CATEGORY category-name ─▼─┬─ ADD ──┬─ category-component ─┘──┘
                               └─ DROP ─┘
```

Expansion of Category-Component

```
                    ┌──────────────────────── , ────────────────────────┐
►►─┬─ ACCESS MODULE ─▼─ dictionary-name.schema-name.access-module-name ─┴─┬─►◄
   │               ┌──────────────────── , ──────────────────┐            │
   ├─ LOAD MODULE ─▼─ dictionary-name.Vnnnn.load-module-name ─┴────────────┤
   │           ┌─────────────── , ────────┐                                │
   ├─ PROGRAM ─▼─ file-name.program-name ─┴────────────────────────────────┤
   │         ┌──────── , ───┐                                              │
   ├─ QUEUE ─▼─ queue-name ─┴──────────────────────────────────────────────┤
   │           ┌──────────────────── , ──────────────────────┐             │
   ├─ RUNUNIT ──▼─ database-name.subschema-name.program-name ─┴─────────────┤
   │        ┌──────── , ───┐                                               │
   └─ TASK ─▼─ task-code ─┴────────────────────────────────────────────────┘
```

Parameters

ACTIVITY

Specifies that you are modifying the security definition of an activity.

application-name

Identifies the application that includes the activity to be altered.

activity-name

Identifies the application activity to be altered.

The activity must have been previously defined with a CREATE RESOURCE statement.

NUMBER activity-number

Specifies the security classification number to be assigned to the activity.

Activity-number must be in the range 1 to 256 and must be unique for the application.

CATEGORY category-name

Specifies that you are modifying the definition of a category.

Category-name must have been previously defined with a CREATE RESOURCE statement.

ADD

Specifies that you are adding a component to the category.

DROP

Specifies that you are dropping a component from the category.

category-component

Identifies the component to be added to or dropped from the category.

Expanded syntax for *category-component* directly follows syntax for ALTER RESOURCE.

A component that has been assigned to a category cannot be assigned to a second category.

You can wildcard the name of the category component.

Note: For more information about wildcarding, see Using a Wildcard (see page 76).

ACCESS MODULE

Specifies that the category component is an access module.

If you drop an access module from a category, users with execution privilege on the category may no longer be able to execute the access module.

dictionary-name

Identifies the dictionary where the access module is stored (DDLCATLOD area).

schema-name

Identifies the schema with which the access module is associated.

access-module-name

Identifies the access module.

LOAD MODULE

Specifies that the category component is a load module.

If you drop a load module from a category, users with execution privilege on the category may no longer be able to execute the load module.

dictionary-name

Identifies the dictionary where the load module is stored (DDLDCLOD area).

Vnnnn

Specifies the version of the load module.

Nnnn identifies the version number of the load module. Leading zeros must be included.

load-module-name

Identifies the load module.

Load-module-name must match the name of an entity defined in the dictionary by means of a CA IDMS or CA ADS compiler.

PROGRAM

Specifies that the category component is a program.

If you drop a program from a category, users with execution privilege on the category may no longer be able to execute the program.

file-name

Supplies the external file name of the load library where the program is stored.

File-name is either 'CDMSLIB' or 'Vnnnn' where nnnn identifies the version number (2 - 9999) of the program. Nnnn must include leading zeroes.

program-name

Identifies the program.

QUEUE

Specifies that the category component is a queue.

If you drop a queue from a category, users with execution privilege on the category may no longer be able to create or access the queue.

queue-name

Identifies the queue.

RUNUNIT

Specifies that the category component is a run unit.

If you drop a run unit from a category, users with execution privilege on the category may no longer be able to execute the run unit.

database-name

Specifies the database to be accessed by the run unit.

subschema-name

Specifies the subschema to be used by the run unit.

program-name

Specifies the name of the program binding the run unit.

TASK

Specifies that the category component is a task.

If you drop a task from a category, users with execution privilege on the category may no longer be able to execute the task.

task-code

Identifies the task.

Usage

Execution Privilege on Individual Access Modules

You can give access module execution privilege by issuing a GRANT EXECUTE ON ACCESS MODULE statement. However, if you subsequently add the same access module to a category, the separate grant of access module execution is ignored at runtime.

If you then drop the access module from the category, and you have not revoked the separate grant of access module execution privilege, the privilege will again be respected at runtime.

Matching Wildcarded Category Component Names

If you use a wildcard when you specify a category component name, the category to which the resource is assigned is the one that most closely matches the resource name.

A match is determined by these rules:

- The resource name must match the category component name character for character up to the wildcard in order to be assigned to the category.
- If the resource name matches more than one category component name up to the wildcard, the resource is assigned to the category in which the most number of characters match.

For example, assume you add these two category component definitions:

```
alter resource category hr_prod
  add load module hrdict.v0001*;

alter resource category hr_test
  add load module hrdict.*;
```

A load module named HRDICT.V0001.HRMAP1 will be assigned to category HR_PROD, whereas load module HRDICT.V0002.HRMAP1 will be assigned to category HR_TEST.

Altering Categories After a Grant

If a component is added to a category with ALTER RESOURCE after a grant of privilege on the category, the privilege is implicitly granted on the added component. Similarly, if a component is dropped from a category with ALTER RESOURCE after a grant of privilege on the category, the privilege on the dropped component is implicitly revoked.

Examples

Altering an Activity: The following statement assigns a new number to an

existing activity created for a General Ledger application:

```
alter resource
activity cgl.post
number 5;
```

Altering a Category: The following statement replaces a resource in an

existing category created for a General Ledger application:

```
alter resource
category glappl
drop queue fl*
add queue gl*;
```

More information

For more information about creating and dropping resources, see the topics CREATE RESOURCE (see page 267) and DROP RESOURCE (see page 273).

CREATE RESOURCE

Creates the definition of a system, activity, or category in the system dictionary.

Authorization

To create a system, activity, or category definition, you must hold one of these privileges:

- DCADMIN
- SYSADMIN

Syntax

```
►►── CREATE RESOURCE ─────────────────────────────────────────────────────────►

 ►─┬─ ACTIVITY application-name.activity-name NUMBER activity-number ─┬───────►◄
   │                           ┌──────────────────────────┐            │
   ├─ CATEGORY category-name ─▼─ ADD category-component ─┴────────────┤
   │                                                                   │
   └─ SYSTEM system-identifier ────────────────────────────────────────┘
```

Expansion of Category-Component

```
                    ┌──────────────────────── , ────────────────────────┐
►►─┬─ ACCESS MODULE ─▼─ dictionary-name.schema-name.access-module-name ─┴──┬──►◄
   │              ┌──────────────────── , ──────────────────┐              │
   ├─ LOAD MODULE ─▼─ dictionary-name.Vnnnn.load-module-name ─┴──────────────┤
   │          ┌───────────── , ─────────┐                                    │
   ├─ PROGRAM ─▼─ file-name.program-name ─┴────────────────────────────────────┤
   │        ┌────── , ─────┐                                                 │
   ├─ QUEUE ─▼─ queue-name ─┴──────────────────────────────────────────────────┤
   │           ┌─────────────────────── , ───────────────────┐               │
   ├─ RUNUNIT ──▼─ database-name.subschema-name.program-name ─┴──────────────┤
   │       ┌────── , ─────┐                                                  │
   └─ TASK ─▼─ task-code ─┴────────────────────────────────────────────────────┘
```

Parameters

ACTIVITY

Specifies that you are defining an activity as a secured resource.

An activity is a discrete application function. After you have defined an activity, you grant execution privilege on the activity to users.

Note: For more information about activities, see the chapter Securing System Resources (see page 95).

application-name

Identifies the application that includes the activity to be secured.

Application-name must match the name of the application (passed in #SECHECK) whose function is being secured. It can be at most eight characters in length.

activity-name

Names the application function to be secured.

Activity-name must be unique within the application. It can be at most 18 characters in length.

NUMBER activity-number

Specifies the activity number assigned to the application function.

Activity-number must be unique within the application. It must be in the range 1 to 256, and it must match the security class assigned within the application.

Note: For more information about assigning activity numbers within applications, see the chapter Securing System Resources (see page 95).

CATEGORY category-name

Specifies that you are defining category category-name as a secured resource.

A category can contain tasks, load modules, programs, access modules, run units, and queues. After you have defined a category, you grant execution privilege on the category to users.

Category-name can be at most 32 characters in length.

You can define a maximum of 32,767 categories for a DC system.

Note: For more information about categories, see the chapter Securing System Resources (see page 95).

ADD category-component

Identifies a component to be added to the category.

Expanded syntax for category-component directly follows syntax for CREATE RESOURCE.

A component that has been assigned to a category cannot be assigned to a second category.

You can wildcard the name of the category component.

Note: For more information about wildcarding, see Using a Wildcard (see page 76).

SYSTEM system-identifier

Specifies that system system-identifier is a secured resource. When the system is secured, users must hold definition privileges to define the system and signon privilege to sign on to the system.

When the system is defined, SYSTEM ID in the SYSTEM statement of system generation must match the value of system-identifier.

ACCESS MODULE

Specifies that the category component is an access module.

After you add an access module to a category, a user with execution privilege on the category can load the access module but cannot execute it.

dictionary-name

Identifies the dictionary where the access module is stored (DDLCATLOD area).

schema-name

Identifies the schema with which the access module is associated.

access-module-name

Identifies an access module.

LOAD MODULE

Specifies that the category component is a load module.

After you add a load module to a category, a user with execution privilege on the category can execute the load module.

dictionary-name

Identifies the dictionary where the load module is stored (DDLDCLOD area).

Vnnnn

Specifies the version of the load module.

Nnnn identifies the version number of the load module. Leading zeros must be included.

load-module-name

Identifies a load module.

PROGRAM

Specifies that the category component is a program.

After you add a program to a category, a user with execution privilege on the category can execute the program.

file-name

Supplies the name of the external file of the load library where the program is stored.

File-name is either 'CDMSLIB' or 'Vnnnn' where nnnn identifies the version number (2 - 9999) of the program. Nnnn must include leading zeroes.

program-name

Identifies a program.

QUEUE

Specifies that the category component is a queue.

After you add a queue to a category, a user with execution privilege on the category can create or access the queue.

queue-name

Identifies a queue.

RUNUNIT

Specifies that the category component is a run unit.

After you add a run unit to a category, a user with execution privilege on the category can use program-name to access data described in subschema-name of database-name.

database-name

Identifies the database to be accessed by the run unit.

subschema-name

Identifies the subschema to be used by the run unit.

program-name

Identifies the program binding the run unit.

TASK

Specifies that the category component is a task.

After you add a task to a category, a user with execution privilege on the category can execute the task.

task-code

Identifies the task.

Usage

Adding Categories or Applications After a Grant

A category created after you grant privileges on the category with a wildcard is not included in the scope of the grant even if the resource name matches the wildcarded name. The wildcard is processed at the time the grant is made, not at runtime.

For example, if you create categories X1, X2, and X3, and then grant privileges on category X*, the three categories are within the scope of the grant. If you then create category X4, this category is not within the scope of the grant.

You can include X4 within the scope of the grant by reissuing the original grant on category X*.

The same considerations apply if you wildcard activity-name when granting execution privilege on an activity. The wildcard is processed at the time the grant is made, not at runtime.

Altering Categories After a Grant

If a component is added to a category (ALTER RESOURCE) after a grant of privilege on the category, the privilege is implicitly granted on the added component. Similarly, if a component is dropped from a category (ALTER RESOURCE) after a grant of privilege on the category, the privilege on the dropped component is implicitly revoked.

Matching Wildcarded Category Component Names

If you use a wildcard when you specify a category component name, the category to which the resource is assigned is the one that most closely matches the resource name.

A match is determined by these rules:

- The resource name must match the category component name character for character up to the wildcard in order to be assigned to the category.
- If the resource name matches more than one category component name up to the wildcard, the resource is assigned to the category in which the most number of characters match.

For example, assume you add these two category component definitions:

```
create resource category hr_prod
  add load module hrdict.v0001*;

create resource category hr_test
  add load module hrdict.*;
```

A load module named HRDICT.V0001.HRMAP1 will be assigned to category HR_PROD, whereas load module HRDICT.V0002.HRMAP1 will be assigned to category HR_TEST.

Examples

The following statement creates a category called PROD_ACCESS that includes resources required for a user to perform production processing:

```
create resource
  category prod_access
    add access module proddict.prod*
    add load module proddict.v0001.*
    add program cdmslib.*
    add rununit p*;
```

Creating an Activity

The following statement assigns an activity name and number to a General Ledger application:

```
create resource
  activity cgl.post
  number 4;
```

Creating a Category

The following statement creates a category of resources used by a General Ledger application:

```
create resource
  category glappl
    add load module appldict.v0001.gl*
    add task gl*
    add queue gl*
    add program cdmslib.gl*;
```

More information

For more information about altering and dropping resources, see “ALTER RESOURCE” and “DROP RESOURCE.”

DROP RESOURCE

Deletes the definition of a secured resource.

Authorization

To delete the definition of a system, activity, or category as a secured resource, you must hold one of these privileges:

- DCADMIN
- SYSADMIN

Syntax

```
►►─── DROP RESOURCE ─┬─ ACTIVITY application-name.activity-name ─┬──────────►◄
                     ├─ CATEGORY category-name ──────────────────┤
                     └─ SYSTEM system-identifier ────────────────┘
```

Parameters

ACTIVITY

Specifies that the resource to be dropped is an activity.

application-name.activity-name

Identifies the activity.

CATEGORY

Specifies that the resource to be dropped is a category.

category-name

Identifies the category.

SYSTEM

Specifies that the resource to be dropped is a system.

system-identifier

Identifies the system.

Usage

Automatic Revoking of Privileges

When you drop a resource, you implicitly revoke all privileges granted on the resource.

Examples

Dropping an Activity

The following statement drops an existing activity created for a General Ledger application:

```
drop resource activity cgl.post;
```

Dropping a Category

The following statement drops an existing category created for a General Ledger application:

```
drop resource category glappl;
```

More information

For more information about creating and altering resources, see CREATE RESOURCE (see page 267) and ALTER RESOURCE (see page 261).

GRANT Administration Privilege

Gives one or more users or groups DCADMIN privilege.

Authorization

To grant DCADMIN privilege, you must hold one of these privileges:

- DCADMIN
- SYSADMIN

Syntax

```
►►─── GRANT DCADMIN ──────────────────────────────────────────────────►

             ┌────────────── , ──────────────┐
 ►─── TO ─▼─┬─ PUBLIC ──────────────────────┬┴────────────────────────►◄
            └─ authorization-identifier ─┘
```

Parameters

DCADMIN

Specifies that you are giving DCADMIN privilege to the users or groups identified in the TO parameter.

DCADMIN controls access to DC system resources. A user with DCADMIN privilege can define system resources and can grant and revoke privileges on system resources.

Note: For more information about the DCADMIN privilege, see Securing System Resources (see page 95).

TO

Specifies the users or groups to whom you are giving DCADMIN privilege.

PUBLIC

Specifies all users.

Important. If you grant DCADMIN to group PUBLIC, any user can administer security for the system.

authorization-identifier

Identifies a user or group.

Note: Expanded syntax for authorization-identifier is presented in Notes on Security Statement Syntax.

Usage

Decentralizing Administration

You can decentralize security administration for the system by granting DCADMIN and system definition privileges to other users.

A holder of SYSADMIN or DCADMIN can also specify WITH GRANT OPTION when granting system definition privileges to allow the recipient to grant the same privileges to others.

Examples

Granting DCADMIN to Administrators:

The following statement grants DCADMIN privilege to the security administrator ID and the DCA group ID:

```
grant dcadmin
  to secadmin, dca_group;
```

More information

For more information about revoking DCADMIN privilege, see REVOKE Administration Privilege (see page 255).

GRANT Execution Privilege

Gives one or more users or groups access to activities or categories.

Authorization

To grant an execution privilege, you must hold one of these privileges:

- DCADMIN
- SYSADMIN

Syntax

```
►►─── GRANT EXECUTE ──────────────────────────────────────────────────────►

                      ┌──────────────── , ─────────────────┐
►─── ON ─┬─ ACTIVITY ─▼─ application-name.activity-name ───┴──┬──────────►
         │            ┌───────── , ───────┐                   │
         └─ CATEGORY ─▼─ category-name ───┴───────────────────┘

           ┌────────────── , ─────────────┐
►─── TO ─▼─┬─ PUBLIC ─────────────────────┬┴──────────────────────────────►◄
           └─ authorization-identifier ───┘
```

Parameters

EXECUTE

Specifies that you are giving execution privilege to the users or groups identified in the TO parameter.

ON

Specifies the resources to which execution privilege applies.

ACTIVITY

Specifies that you are giving execution privilege on one or more activities.

application-name.activity-name

Identifies an activity.

Application-name.activity-name must have been previously defined as an activity with a CREATE RESOURCE statement.

You can wildcard activity-name. You cannot wildcard application-name. The wildcard character can appear in activity-name at any point after the period following application-name.

Note: For more information about wildcarding, see Using a Wildcard (see page 76).

CATEGORY

Specifies that you are giving execution privilege on one or more categories.

category-name

Identifies a category.

Category-name must have been previously defined as a category with a CREATE RESOURCE statement.

You can wildcard category-name.

Note: For more information about wildcarding, see Using a Wildcard (see page 76).

TO

Specifies the users or groups to whom you are giving execution privilege.

PUBLIC

Specifies all users.

authorization-identifier

Identifies a user or group.

Note: Expanded syntax for authorization-identifier is presented in the chapter “Notes on Security Statement Syntax.”

Usage

Wildcarding Activity-Name

If you wildcard activity-name in a GRANT EXECUTE statement, you grant execution privilege on all activities whose names match the wildcarded name.

For example, this statement gives execution privilege on all DCMT activities whose names begin with 'VARY' to DCA_GROUP:

```
grant execute
 on activity dcmt.vary*
 to dca_group;
```

Adding Categories or Activities After a Grant

A category created after you grant privileges on the category with a wildcard is not included in the scope of the grant even if the resource name matches the wildcarded name. The wildcard is processed at the time the grant is made, not at runtime.

For example, if you create categories X1, X2, and X3, and then grant privileges on category X*, the three categories are within the scope of the grant. If you then create category X4, this category is not within the scope of the grant.

You can include X4 within the scope of the grant by revoking the original grant on category X* and then reissuing it.

The same considerations apply if you wildcard activity-name when granting execution privilege on an activity. The wildcard is processed at the time the grant is made, not at runtime.

Examples

Granting Execution Privilege on an Activity

The following statement grants execution privilege on a General Ledger activity to a development group:

```
grant execute
  on activity cgl.post
  to appldev1;
```

Granting Execution Privilege on a Category

The following statement grants execution privilege on the General Ledger category to all users:

```
grant execute
  on category glappl
  to public;
```

Granting Execution Privilege to PUBLIC

If execution privilege on a category has been granted to group PUBLIC, a user who has not signed on has the ability to invoke tasks and access other secured resources in the category.

More information

For more information about revoking execution privilege, see REVOKE Execution Privilege (see page 288).

GRANT Signon Privilege

Gives one or more users access to a CA IDMS system.

Authorization

To issue the GRANT SIGNON statement, you must hold one of these privileges:

- DCADMIN
- SYSADMIN

Syntax

```
►►── GRANT SIGNON ON SYSTEM system-identifier ──────────────────────────►

►─┬──────────────────────────────┬──────────────────────────────────────►
  └─ PROFILE ─┬─ profile-name ─┬─┘
              └─ NULL ─────────┘

            ┌──────────── , ────────┐
►── TO ─▼── user-identifier ────────┴───────────────────────────────────►◄
```

Parameters

SIGNON

Specifies that you are giving signon privilege to the users identified in the TO parameter.

ON SYSTEM system-identifier

Identifies the system to which signon privilege applies.

System-identifier must have been defined as a resource with the CREATE RESOURCE SYSTEM statement.

Note: You cannot wildcard system-identifier.

PROFILE profile-name

Identifies the system profile to be used in signon processing.

NULL

Specifies that no system profile should be used in signon processing.

TO user-identifier

Identifies a user to whom you are giving signon privilege.

Usage

Changing the User's System Profile

To change the system profile associated with a user identifier:

- Revoke signon privilege from the user identifier.
- Grant signon privilege to the user identifier specifying the new profile.

Examples

Granting Signon with a Profile Specification

The following statement grants signon privilege and specifies the system profile to be invoked in signon processing:

```
grant signon
  on system syst0099
  profile pub99
  to sam;
```

Changing the User's System Profile

The following statements change user SAM's system profile to CORP99:

```
revoke signon on system syst0099
  from sam;

grant signon on system syst0099
  profile corp99
  to sam;
```

More information

- For more information about signon processing, see the chapter Signon Processing (see page 57).
- For more information about revoking signon privilege, see REVOKE Signon Privilege (see page 290).
- For more information about system profiles, see the *CA IDMS System Tasks and Operator Commands Guide*.

GRANT System Definition Privileges

Gives to one or more users or groups the privilege to define a system or a system profile in the system dictionary.

Authorization

To grant system definition privileges, you must hold one of these privileges:

- DCADMIN privilege
- The corresponding grantable privilege on the system or system profile
- SYSADMIN

Syntax

```
►►──── GRANT ─┬── DEFINE ────────────────────────────────────────────────────►
              │ ┌─────── , ──────┐ │
              └─▼─┬─ ALTER ───┬──┴─┘
                  ├─ CREATE ──┤
                  ├─ DISPLAY ─┤
                  └─ DROP ────┘

►──── ON ─┬─ SYSTEM system-identifier ───┬───────────────────────────────────►
          └─ SYSTEM PROFILE profile-name ┘

          ┌─────────────── , ───────────────┐
►──── TO ─▼─┬─ PUBLIC ──────────────────┬───┴────────────────────────────────►
            └─ authorization-identifier ┘

►─┬───────────────────────┬──────────────────────────────────────────────────►◄
  └─ WITH GRANT OPTION ───┘
```

Parameters

DEFINE

Gives the ALTER, CREATE, DISPLAY, and DROP privileges on the resource identified in the ON parameter to the users or groups identified in the TO parameter.

The DEFINE privilege on a system allows the user to use the CA IDMS system generation compiler to add, modify, or delete the system. The DEFINE privilege also allows the user to display or punch the definition of the system.

The DEFINE privilege on a system profile allows the user to create, alter, or drop a system profile, or to display security definitions for the system profile.

ALTER

Gives the ALTER privilege on the resource identified in the ON parameter to the users or groups identified in the TO parameter.

The ALTER privilege on a system allows a user to use the system generation compiler to modify the definition of the system.

The ALTER privilege on a system profile allows a user to alter the system profile.

CREATE

Gives the CREATE privilege on the resource identified in the ON parameter to the users or groups identified in the TO parameter.

The CREATE privilege on a system allows a user to use the system generation compiler to add the system.

The CREATE privilege on a system profile allows a user to create the system profile.

DISPLAY

Gives the DISPLAY privilege on the resource identified in the ON parameter to the users or groups identified in the TO parameter.

The DISPLAY privilege allows the user to issue a DISPLAY RESOURCE statement on the named system or system profile. The grantable DISPLAY privilege allows a user to issue a DISPLAY PRIVILEGES statement on the named system or system profile.

The DISPLAY privilege on a system also allows a user to use the system generation compiler to display or punch the system definition.

DROP

Gives the DROP privilege on the resource identified in the ON parameter to the users or groups identified in the TO parameter.

The DROP privilege on a system allows a user to use the system generation compiler to delete the definition of the system.

The DROP privilege on a system profile allows a user to drop the system profile.

ON

Specifies the resource to which the definition privileges apply.

SYSTEM system-identifier

Identifies a system.

You can wildcard system-identifier.

Note: For more information about wildcarding, see Using a Wildcard (see page 76).

If you do not wildcard system-identifier, the value you specify must match the identifier of a system secured with the CREATE RESOURCE SYSTEM statement.

SYSTEM PROFILE profile-name

Identifies a system profile.

Note: For more information about creating system profiles, see the *CA IDMS System Tasks and Operator Commands Guide*.

You can wildcard system-profile.

Note: For more information about wildcarding, see Using a Wildcard (see page 76).

TO

Specifies the users or groups to whom you are giving the definition privileges.

PUBLIC

Specifies all users.

authorization-identifier

Identifies a user or group.

Note: Expanded syntax for authorization-identifier is presented in the chapter Notes on Security Statement Syntax (see page 225).

WITH GRANT OPTION

Gives the privilege of granting the specified definition privileges to the users or groups identified in the TO parameter.

A privilege granted with the WITH GRANT OPTION is called a grantable privilege.

Usage

The DEFINE Keyword

When you use the DEFINE keyword with a GRANT statement, you grant a set of definition privileges on a resource to one or more users or groups.

When you use the DEFINE keyword with a REVOKE statement, you revoke all definition privileges that have been previously granted on the resource from the specified users or groups.

This means that if you GRANT CREATE privilege on a resource, you can revoke the privilege with either a REVOKE CREATE statement or a REVOKE DEFINE statement. Using REVOKE DEFINE is an efficient technique when you intend to revoke all definition privileges on the resource from a user or group, whether the privileges were granted singly or as a set.

Similarly, you can GRANT DEFINE on a resource to a user and then REVOKE DROP on the resource from the same user as a way to grant all but one definition privilege.

Examples

Granting Privilege to Define the System

The following statement grants the privilege to define a system to the system DCA:

```
grant define
  on system syst0099
  to dca0099;
```

More information

- For more information about system profiles, see the *CA IDMS System Tasks and Operator Commands Guide*.
- For more information about revoking system definition privileges, see REVOKE System Definition Privileges (see page 291).

REVOKE Administration Privilege

Revokes DCADMIN privilege from one or more users or groups.

Authorization

To revoke DCADMIN privilege, you must hold one of these privileges:

- DCADMIN
- SYSADMIN

Syntax

```
►►── REVOKE DCADMIN ────────────────────────────────────────────────────►

                 ┌────────────── , ──────────────┐
►── FROM ─▼┬─ PUBLIC ────────────────────┬┴────────────────────────────►◄
           └─ authorization-identifier ──┘
```

Parameters

DCADMIN

Specifies that you are revoking DCADMIN privilege from the users or groups named in the FROM parameter.

FROM

Specifies the users or groups from whom you are revoking DCADMIN privilege.

PUBLIC

Specifies all users.

authorization-identifier

Identifies a user or group.

The privilege must have been previously given to authorization-identifier by means of the GRANT statement.

Note: Expanded syntax for authorization-identifier is presented in chapter “Notes on Security Statement Syntax."

Example

Revoking DCADMIN from Administrators

The following statement revokes DCADMIN privilege from the DCA group ID:

```
revoke dcadmin
 from dca_group;
```

More information

For more information about granting DCADMIN privilege, see GRANT Administration Privilege (see page 275).

REVOKE Execution Privilege

Revokes from one or more users or groups the privilege to access an activity or category.

Authorization

To revoke execution privilege, you must hold one of these privileges:

- DCADMIN
- SYSADMIN

Syntax

```
►►─── REVOKE EXECUTE ───────────────────────────────────────────────────►

                      ┌──────────────── , ──────────────┐
►─── ON ─┬─ ACTIVITY ─▼─ application-name.activity-name ─┴──────────┬───►
         │            ┌─────── , ──────┐                           │
         └─ CATEGORY ─▼─ category-name ─┴──────────────────────────┘

            ┌──────────────── , ──────────────┐
►─── FROM ─▼─┬─ PUBLIC ───────────────────────┴───────────────────────►◄
             └─ authorization-identifier ─┘
```

Parameters

EXECUTE

Specifies that you are revoking execution privilege from the users or groups identified in the FROM parameter.

ON

Specifies the resource to which execution privilege applies.

ACTIVITY

Specifies that you are revoking execution privilege on one or more activities.

application-name.activity-name

Identifies an activity.

You can wildcard activity-name. You cannot wildcard application-name. The wildcard character can appear in activity-name at any point following the period after application-name.

Note: For more information about wildcarding, see Using a Wildcard (see page 76).

CATEGORY

Specifies that you are revoking execution privilege on one or more categories.

category-name

Identifies a category.

FROM

Specifies the users or groups from whom you are revoking execution privilege.

PUBLIC

Specifies all users.

authorization-identifier

Identifies a user or group.

The privilege must have been previously given to authorization-identifier by means of the GRANT statement.

Note: Expanded syntax for authorization-identifier is presented in the chapter Notes on Security Statement Syntax (see page 225).

Examples

Revoking Execution Privilege on an Activity

The following statement revokes execution privilege on a General Ledger activity from a development group:

```
revoke execute
  on activity cgl.post
  from appldev1;
```

Revoking Execution Privilege on a Category

The following statement revokes execution privilege on the General Ledger category from PUBLIC:

```
revoke execute
  on category glappl
  from public;
```

More information

For more information about granting execution privilege, see GRANT Execution Privilege (see page 277).

REVOKE Signon Privilege

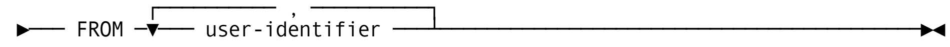
Revokes the privilege to access a system from one or more users.

Authorization

To revoke signon privilege, you must hold one of these privileges:

- DCADMIN
- SYSADMIN

Syntax



```
►►── REVOKE SIGNON ON SYSTEM system-identifier ────────────────►

            ┌─────────── , ───────────┐
►── FROM ─▼─── user-identifier ───┴──────────────────────────►◄
```

Parameters

SIGNON

Specifies that you are revoking signon privilege to the system identified in the ON parameter from the users identified in the FROM parameter.

ON SYSTEM system-identifier

Identifies the system to which the signon privilege applies.

FROM user-identifier

Identifies a user from whom you are revoking signon privilege.

The privilege must have been previously given to user-identifier by means of the GRANT statement.

Example

Revoking Signon

The following statement revokes signon privilege on a specified system from a user:

```
revoke signon
  on system syst0099
  from sam;
```

More information

For more information about granting signon privilege, see GRANT Signon Privilege (see page 280).

REVOKE System Definition Privileges

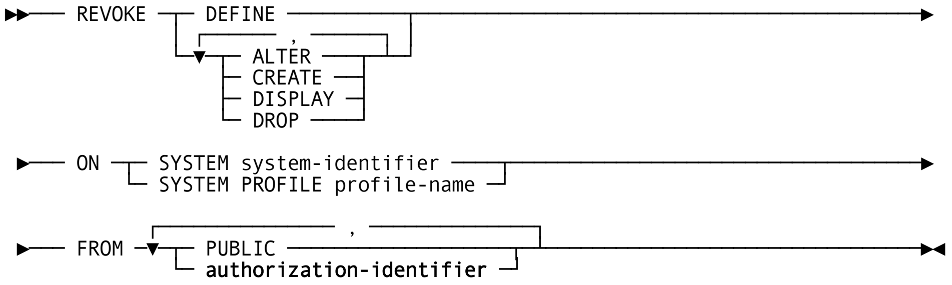
Revokes from one or more users or groups the privilege to define a system or a system profile in the system dictionary.

Authorization

To revoke system privileges, you must hold one of these privileges:

- DCADMIN privilege
- The corresponding grantable privilege on the system or system profile
- SYSADMIN

Syntax



Parameters

DEFINE

Revokes the ALTER, CREATE, DISPLAY, and DROP privileges on the resource identified in the ON parameter from the users or groups identified in the TO parameter.

ALTER

Revokes the ALTER privilege on the resource identified in the ON parameter from the users or groups identified in the TO parameter.

CREATE

Revokes the CREATE privilege on the resource identified in the ON parameter from the users or groups identified in the TO parameter.

DISPLAY

Revokes the DISPLAY privilege on the resource identified in the ON parameter from the users or groups identified in the TO parameter.

DROP

Revokes the DROP privilege on the resource identified in the ON parameter from the users or groups identified in the TO parameter.

ON

Specifies the resource to which the definition privileges apply.

SYSTEM system-identifier

Identifies a system.

SYSTEM PROFILE profile-name

Identifies a system profile.

FROM

Specifies the users or groups from whom you are revoking the specified definition privileges.

PUBLIC

Specifies all users.

authorization-identifier

Identifies a user or group.

The privileges must have been previously given to authorization-identifier by means of the GRANT statement.

Note: Expanded syntax for authorization-identifier is presented in the chapter “Notes on Security Statement Syntax."

Usage

The DEFINE Keyword

When you use the DEFINE keyword with a GRANT statement, you grant a set of definition privileges on a resource to one or more users or groups.

When you use the DEFINE keyword with a REVOKE statement, you revoke all definition privileges that have been previously granted on the resource from the specified users or groups.

This means that if you GRANT CREATE privilege on a resource, you can revoke the privilege with either a REVOKE CREATE statement or a REVOKE DEFINE statement. Using REVOKE DEFINE is an efficient technique when you intend to revoke all definition privileges on the resource from a user or group, whether the privileges were granted singly or as a set.

Similarly, you can GRANT DEFINE on a resource to a user and then REVOKE DROP on the resource from the same user as a way to grant all but one definition privilege.

Examples

Revoking Privilege to Define the System

The following statement revokes the privilege to define a system from the system DCA:

```
revoke define
  on system syst0099
  from dca0099;
```

More information

- For more information about system profiles, see the *CA IDMS System Tasks and Operator Commands Guide*.
- For more information about granting system definition privileges, see GRANT System Definition Privileges (see page 282).

Chapter 15: Syntax for Securing Database Resources

This section contains the following topics:

GRANT Access Module Execution Privilege

Gives one or more users or groups the privilege of executing a specified access module.

GRANT Access Module Execution Privilege Authorization

To grant access module execution privilege, one of the following must be true:

- You hold grantable execution privilege on the access module (you can grant execution privilege, but you cannot specify WITH GRANT OPTION).
- You own the schema with which the access module is associated.
- You hold DBADMIN privilege on the dictionary that contains the access module.
- You hold SYSADMIN privilege.

You must be connected to the application dictionary that contains the access module when you issue the statement.

GRANT Access Module Execution Privilege Syntax

```
►►─── GRANT EXECUTE ──────────────────────────────────────────────────────────►

►─── ON ACCESS MODULE ─┬──────────────┬─ access-module-name ─────────────────►
                       └ schema-name. ┘

              ┌─────────────── , ─────────────┐
►─── TO ─▼┬─ PUBLIC ─────────────────────┬────┴──────────────────────────────►
          └─ authorization-identifier ───┘

►─┬─────────────────────┬────────────────────────────────────────────────────►◄
  └─ WITH GRANT OPTION ─┘
```

GRANT Access Module Execution Privilege Parameters

ON ACCESS MODULE *access-module-name*

Identifies the access module to which the EXECUTE privilege applies.

You can wildcard *access-module-name*. If you specify *schema-name*, the wildcard character is valid after the period following *schema-name*.

Note: For more information, see Using a Wildcard (see page 76).

schema-name

Identifies the schema associated with *access-module-name*.

If you do not specify *schema-name*, it defaults to the current schema in effect for your session.

Note: For more information about using a schema name to qualify an access module name, see the *CA IDMS SQL Reference Guide*.

TO

Identifies the users or groups to whom you are giving EXECUTE privilege.

PUBLIC

Specifies all users.

authorization-identifier

Identifies a user or group.

WITH GRANT OPTION

Gives the privilege of granting EXECUTE privilege on the named access module to the users identified in the TO parameter.

A privilege granted with the WITH GRANT OPTION is called a grantable privilege.

GRANT Access Module Execution Privilege Usage

CA IDMS Internal Security Enforcement

When executing an access module in a database for which CA IDMS internal security is in effect, the *owner* of the access module must either hold the applicable privileges on the table-like objects named in the SQL statements in the module, or own the table-like objects. You own the access module if you own the schema associated with the access module.

For a user who is *not* the owner to execute an access module, these conditions must be satisfied:

- The user must hold execution privilege on the access module.
- The owner must either hold the applicable *grantable* privileges on the table-like objects named in the SQL statements in the module, or own the table-like objects.

These rules allow you to restrict a user's means of accessing data to application programs. If you grant table access privileges, the user can also access data through the Command Facility.

External Security

When executing an access module in a database for which external security is in effect, the user, regardless of ownership, must hold the applicable privileges on *all* tables accessed by SQL statements in the module, whether accessed directly or indirectly through a view.

Granting Execution Privilege

The following GRANT statement gives execution privilege on all access modules associated with schema HR that begin with 'EMP' to the groups PER_GRP_1 and PER_GRP_2:

```
grant execute
  on access module hr.emp*
  to per_grp_1, per_grp_2;
```

GRANT Access Module Execution Privilege More Information

For more information **about revoking execution privilege**, see REVOKE Access Module Execution Privilege (see page 318).

GRANT Administration Privilege

Gives one or more users or groups DBADMIN privilege on a specified database.

GRANT Administration Privilege Authorization

To grant DBADMIN privilege, you must hold one of these privileges:

- DBADMIN on the database
- SYSADMIN

You must be connected to the system dictionary when you issue the statement.

GRANT Administration Privilege Syntax

```
►►─── GRANT DBADMIN ON DB database-name ─────────────────────────────────►

               ┌──────────── , ─────────────┐
 ►─── TO ─▼─┬─ PUBLIC ────────────────────┬─┴───────────────────────────►◄
            └─ authorization-identifier ──┘
```

GRANT Administration Privilege Parameters

DBADMIN

Specifies that you are giving DBADMIN privilege on the database identified in the ON parameter to the users or groups identified in the TO parameter.

ON DB *database-name*

Specifies the database to which DBADMIN privilege applies.

Database-name refers to either the name of a segment or a database name entry in the database name table.

DBADMIN controls access to database resources. A user with DBADMIN privilege can grant and revoke privileges for the specified database. DBADMIN privilege also allows users to maintain physical database definitions in the dictionary identified by *database-name*.

Note: For more information about the DBADMIN privilege, see Securing Database Resources (see page 117).

You can wildcard *database-name*.

Note: For more information about wildcarding, see Using a Wildcard (see page 76).

TO

Specifies the users or groups to whom you are giving DBADMIN privilege.

PUBLIC

Specifies all users.

Important: If you grant DBAMIN to group PUBLIC, any user can administer security on the database.

authorization-identifier

Identifies a user or group.

Note: Expanded syntax for *authorization-identifier* is presented in Notes on Security Statement Syntax (see page 225).

Granting DBADMIN to the DBA

The following statement grants DBADMIN privilege on database GLDB to the DBA group ID:

```
grant dbadmin
  on db gldb
  to dba_gldb;
```

GRANT Administration Privilege More Information

For more information **about revoking DBADMIN privilege**, see REVOKE Administration Privilege (see page 320).

GRANT All Table Privileges

Gives one or more users all definition and access privileges on a specified table-like object.

GRANT All Table Privileges Authorization

To grant all table privileges, one of the following must be true:

- You hold all privileges on the table-like object as grantable privileges. (you can grant the privileges, but you cannot specify WITH GRANT OPTION)
- You own the table-like object.
- You hold DBADMIN privilege the application dictionary where the table-like object is defined and on the database that contains the table-like object data.
- You hold SYSADMIN privilege.

GRANT All Table Privileges Syntax

```
►►─── GRANT ALL PRIVILEGES ─────────────────────────────────────►

►─── ON table table-name ───────────────────────────────────────►

          ┌──────────────── , ────────────┐
►─── TO ─▼─┬─ PUBLIC ────────────────────┬┴─────────────────────►
           └─ authorization-identifier ──┘

►─┬─────────────────────────────────────────────────────────────►◄
  └─ WITH GRANT OPTION ─┘
```

GRANT All Table Privileges Parameters

ALL PRIVILEGES

Gives the DELETE, INSERT, SELECT, UPDATE, ALTER, CREATE, DROP, and REFERENCES privileges on the table-like object, as applicable, identified in the ON parameter to the users or groups identified in the TO parameter.

ON table table-name

Identifies the table-like object to which the table privileges apply.

Note: Expanded syntax for **table-name** is presented in Notes on Security Statement Syntax.

You can wildcard the *identifier-components* of **table-name**. If you specify *schema-name* in **table-name**, the wildcard character is valid after the period following *schema-name*.

Note: For more information about wildcarding, see Using a Wildcard (see page 76).

If you do not specify *schema-name*, it defaults to the current schema in effect for your session.

TO

Specifies the users or groups to whom you are giving table privileges.

PUBLIC

Specifies all users.

authorization-identifier

Identifies a user or group.

Note: Expanded syntax for **authorization-identifier** is presented in Using a Wildcard (see page 76).

WITH GRANT OPTION

Gives the privilege of granting the all table privileges on **table-name** to the users or groups identified in the TO parameter. The owner of the resource, a holder of the applicable DBADMIN privilege, or a holder of SYSADMIN privilege can specify WITH GRANT OPTION.

A privilege granted with the WITH GRANT OPTION is called a grantable privilege.

GRANT All Table Privileges Usage

Verification of Wildcarded Grants

When you grant all table privileges, you grant a combination of table access and table definition privileges. If you wildcard **table-name**, the verification at runtime of the user's access privilege is handled differently from verification of the user's definition privilege:

- For definition privileges, only the closest matching wildcarded grant is used. If CREATE privilege has been granted on HR.EMP* and HR.EMPV*, then only the grant on HR.EMPV* is used to verify the privilege to create HR.EMPVU_SALARY.
- For access privileges, all matching wildcarded grants are used. If SELECT privilege has been granted on HR.EMP* and HR.EMPV*, then users or groups receiving either the HR.EMP* or the HR.EMPV* grant are authorized to select from EMPVU_SALARY.

Granting All Privileges to All Users

The following GRANT statement gives all users all privileges on all table-like objects in the TEST schema:

```
grant all privileges
  on test.*
  to public;
```

GRANT All Table Privileges More Information

- For more information about table access privileges, see GRANT Table Access Privileges (see page 315).
- For more information about table definition privileges, see GRANT SQL Definition Privileges (see page 311).

GRANT Area Access Privileges

Gives one or more users or groups access to an area of the database.

GRANT Area Access Privileges Authorization

To grant an area access privilege, you must hold one of these privileges:

- The grantable area access privilege on the area (you can grant the privilege, but you cannot specify WITH GRANT OPTION)
- DBADMIN on DB *segment-name*
- SYSADMIN

You must be connected to the system dictionary when you issue the statement.

GRANT Area Access Privileges Syntax

```
            ┌──────── , ───────┐
►►── GRANT ─▼─┬─ DBAREAD ──┬───┴──────────────────────────────────────►
              ├─ DBAWRITE ─┤
              └─ USE ──────┘

►── ON AREA segment-name.area-name ───────────────────────────────────►

          ┌──────────────── , ───────────────┐
►── TO ─▼─┬─ PUBLIC ────────────────────┬────┴────────────────────────►
          └─ authorization-identifier ──┘

►─┬──────────────────────┬────────────────────────────────────────────►◄
  └─ WITH GRANT OPTION ──┘
```

GRANT Area Access Privileges Parameters

DBAREAD

Specifies that you are giving DBAREAD privilege on the area identified in the ON parameter to the users or groups specified in the TO parameter.

A user with DBAREAD privilege can execute database utilities that perform read-only functions in the specified area.

DBAWRITE

Specifies that you are giving DBAWRITE privilege on the area identified in the ON parameter to the users or groups specified in the TO parameter.

A user with DBAWRITE privilege can execute database utilities that perform read-write functions in the specified area.

Note: DBAWRITE privilege does *not* imply DBAREAD privilege. You must give both privileges to users or groups who need to execute all utilities.

USE

Specifies that you are giving USE privilege on the area identified in the ON parameter to the users or groups specified in the TO parameter.

A user with USE privilege can create an SQL table or index in the specified area.

ON AREA *segment-name.area-name*

Identifies the area to which the specified area access privileges apply.

You can wildcard *area-name*. You cannot wildcard *segment-name*. The wildcard character is valid after the period following *segment-name*.

Note: For more information about wildcarding, see Using a Wildcard (see page 76).

TO

Specifies the users or groups to whom you are giving area access privileges.

PUBLIC

Specifies all users.

authorization-identifier

Identifies a user or group.

Note: Expanded syntax for **authorization-identifier** is presented in Notes on Security Statement Syntax (see page 225).

WITH GRANT OPTION

Gives the privilege of granting the specified area access privileges to the users or groups identified in the TO parameter. Only a user with DBADMIN privilege on *segment-name* or with SYSADMIN privilege can specify WITH GRANT OPTION.

A privilege granted with the WITH GRANT OPTION is called a grantable privilege.

Granting All Area Access Privileges

The following statement grants all area access privileges to the specified users:

```
on area gl."account-area"
to matt, alex;
```

GRANT Area Access Privileges More Information

For more information about revoking the privilege to access an area, see REVOKE Area Access Privileges (see page 323).

GRANT Non-SQL Definition Privilege

Gives one or more users or groups the privilege of referencing a non-SQL-defined schema in an SQL schema definition.

GRANT Non-SQL Definition Privilege Authorization

To grant the USE privilege on a non-SQL-defined schema, you must hold one of these privileges:

- Grantable privilege on the non-SQL-defined schema (you can grant the privilege, but you cannot specify WITH GRANT OPTION).
- DBADMIN on the dictionary containing the non-SQL-defined schema definition.
- SYSADMIN.

You must be connected to the dictionary containing the non-SQL-defined schema when you issue the statement.

GRANT Non-SQL Definition Privilege Syntax

```
►►── GRANT ─┬─ USE ─────┬──────────────────────────────────────────────►
            └─ DISPLAY ─┘

►── ON NONSQL SCHEMA Vnnnn.nonsql-schema-name ─────────────────────────►

            ┌───────────── , ────────────┐
►── TO ─▼─┬─ PUBLIC ───────────────────┬─┴──────────────────────────►
          └─ authorization-identifier ─┘

►─┬──────────────────────┬───────────────────────────────────────────►◄
  └─ WITH GRANT OPTION ──┘
```

GRANT Non-SQL Definition Privilege Parameters

USE

Gives the USE privilege on the non-SQL-defined schema identified in the ON parameter to the users or groups identified in the TO parameter.

DISPLAY

Gives the DISPLAY privilege on the non-SQL-defined schema identified in the ON parameter to the users or groups identified in the TO parameter.

ON NONSQL SCHEMA

Specifies the non-SQL-defined schema to which the USE privilege applies.

V*nnnn.nonsql-schema-name*

Specifies the version number and name of the non-SQL-defined schema. The version number (*nnnn*) must include leading zeros.

You can wildcard *nonsql-schema-name*. You cannot wildcard V*nnnn*. The wildcard character is valid after the period following V*nnnn*.

Note: For more information about wildcarding, see Using a Wildcard (see page 76).

TO

Specifies the users or groups to whom you are giving the USE privilege.

PUBLIC

Specifies all users.

authorization-identifier

Identifies a user or group.

Note: Expanded syntax for **authorization-identifier** is presented in Notes on Security Statement Syntax (see page 225).

WITH GRANT OPTION

Gives the privilege of granting the USE privilege on the named resource to the users or groups identified in the TO parameter.

A privilege granted with the WITH GRANT OPTION is called a grantable privilege.

Granting USE On a Non-SQL-Defined Schema

The following statement grants the privilege of referencing a non-SQL-defined schema when creating an SQL schema:

```
grant use
  on nonsql schema v0001.ap
  to sal, sam;
```

GRANT Non-SQL Definition Privilege More Information

For more information about revoking the privilege to use a non-SQL-defined schema, see REVOKE Non-SQL Definition Privilege (see page 325).

GRANT Physical Database Definition Privileges

Gives one or more users or groups the privilege of issuing DMCL, DBTABLE, and SEGMENT physical DDL statements.

GRANT Physical Database Definition Privileges Authorization

To grant a definition privilege on a DMCL or DBTABLE, you must hold one of these privileges:

- The corresponding grantable privilege
- DBADMIN on DB SYSTEM
- SYSADMIN

To grant a physical definition privilege on a database, you must hold one of the following privileges:

- The corresponding grantable privilege
- DBADMIN on the specified DB
- SYSADMIN on the specified DB

You must be connected to the system dictionary when you issue the statement.

GRANT Physical Database Definition Privileges Syntax

```
►►─── GRANT ─┬─ DEFINE ─────────────────┬──────────────────────────────────►
             │  ┌──────── , ─────────┐ │
             └─▼┬─── ALTER ──────────┼─┘
                ├─── CREATE ─────────┤
                ├─── DISPLAY ────────┤
                ├─── DROP ───────────┤
                └─── USE ────────────┘

 ►─── ON ─┬─ DMCL dmcl-name ───────────┬───────────────────────────────────►
          ├─ DBTABLE dbtable-name ─────┤
          └─ DB database-name ─────────┘

 ►─── TO ─▼┬─ PUBLIC ─────────────────────┬────────────────────────────────►
           │ ┌──────────── , ──────────┐  │
           └─ authorization-identifier ─┘

 ►─┬──────────────────────────────────────────────────────────────────────►◄
   └─ WITH GRANT OPTION ─┘
```

GRANT Physical Database Definition Privileges Parameters

DEFINE

Gives the ALTER, CREATE, DISPLAY, DROP, and USE privileges, as applicable, on the resource identified in the ON parameter to the users or groups identified in the TO parameter.

ALTER

Gives the ALTER privilege on the resource identified in the ON parameter to the users or groups identified in the TO parameter.

The ALTER privilege on a resource allows a user to modify the definition of the resource. The ALTER privilege on a DMCL or database name table also allows a user to generate a load module from the definition.

CREATE

Gives the CREATE privilege on the resource identified in the ON parameter to the users or groups identified in the TO parameter.

The CREATE privilege on a resource allows a user to define the resource.

DISPLAY

Gives the DISPLAY privilege on the resource identified in the ON parameter to the users or groups identified in the TO parameter.

The DISPLAY privilege allows the user to issue a DISPLAY RESOURCE statement on the named resource. The grantable DISPLAY privilege allows a user to issue a DISPLAY PRIVILEGES statement on the named resource.

The DISPLAY privilege on a DBTABLE resource is required for a user to produce a DBTABLE listing using IDMSRPTS. The DISPLAY privilege on a DMCL resource is required for a user to produce a DMCL listing using IDMSRPTS. The DISPLAY privilege on a DB resource is required for a user to produce a segment listing using IDMSRPTS.

DROP

Gives the DROP privilege on the resource identified in the ON parameter to the users or groups identified in the TO parameter.

The DROP privilege on a resource allows a user to delete the definition of the resource.

USE

Gives the USE privilege on the resource identified in the ON parameter to the users or groups identified in the TO parameter.

- **DMCL**—The USE privilege allows a user to format, print, archive, and fix the journal files defined by the DMCL and punch the DMCL load module.
- **Database name table**—The USE privilege allows a user to punch the database name table load module and specify the database name table in the DBTABLE parameter of a DMCL definition.
- **Segment**—The USE privilege allows a user to associate the segment with an SQL schema.

ON

Specifies the resource to which the definition privileges apply.

DMCL *dmcl-name*

Identifies a DMCL.

The scope of a privilege granted on a DMCL resource includes these physical database definition statements:

- DMCL
- BUFFER
- JOURNAL BUFFER
- ARCHIVE JOURNAL
- DISK JOURNAL
- TAPE JOURNAL

You can wildcard *dmcl-name*.

Note: For more information about wildcarding, see Using a Wildcard (see page 76).

DBTABLE *dbtable-name*

Identifies a database name table.

The scope of a privilege granted on a DBTABLE resource includes these physical database definition statements:

- DBTABLE
- DBNAME

You can wildcard *dbtable-name*.

Note: For more information about wildcarding, see Using a Wildcard (see page 76).

DB *database-name*

Identifies a segment or a name in the database name table.

The scope of a privilege granted on a DB resource includes these physical database definition statements:

- SEGMENT
- FILE
- AREA

You can wildcard *database-name*.

Note: For more information about wildcarding, see Using a Wildcard (see page 76).

TO

Specifies the users or groups to whom you are giving definition privileges.

PUBLIC

Specifies all users.

authorization-identifier

Identifies a user or group.

Note: Expanded syntax for **authorization-identifier** is presented in chapter Notes on Security Statement Syntax (see page 225).

WITH GRANT OPTION

Gives the authority to grant the specified definition privileges on the named resource to the users or groups identified in the TO parameter. Only a holder of the applicable DBADMIN privilege or a holder of SYSADMIN privilege can specify WITH GRANT OPTION.

A privilege granted with the WITH GRANT OPTION is called a grantable privilege.

GRANT Physical Database Definition Privileges Usage

The DEFINE Keyword

When you use the DEFINE keyword with a GRANT statement, you grant a set of definition privileges to one or more users or groups.

When you use the DEFINE keyword with a REVOKE statement, you revoke all of the privileges in the set that have been previously granted to the specified users or groups.

This means that if you GRANT CREATE privilege on a resource, you can revoke the privilege with either a REVOKE CREATE statement or a REVOKE DEFINE statement. Using REVOKE DEFINE is an efficient technique when you intend to revoke all definition privileges from a user or group, whether the privileges were granted singly or as a set.

Similarly, you can GRANT DEFINE on a resource to a user and then REVOKE DROP on the resource from the same user as a way to grant all but one definition privilege.

Security Considerations for IDMSRPTS

If a dictionary named in an IDMSRPTS run has been secured, the user who submits the job must have EXECUTE privilege on the category containing the run unit *dictionary-name*.IDMSNWKG.IDMSRPTS. Additional privileges may be required depending on the reports requested:

Report	**Privilege**
DBTLST (DBTABLE listing)	DBADMIN on the dictionary or DISPLAY on the DBTABLE
DMCLST (DMCL listing)	DBADMIN on the dictionary or DISPLAY on the DMCL
SEGLST (segment listing)	DBADMIN on the dictionary or DISPLAY on the DB
All other reports	Governed by application dictionary security

Note: For more information, see the chapter Securing Application Dictionary Resources (see page 153).

Granting Privilege to Issue DMCL Statements

The following statement gives the users the privilege to issue DMCL definition statements for DMCL99:

```
grant define
  on dmcl dmcl99
  to mike, ryan;
```

GRANT Physical Database Definition Privileges More Information

For more information about revoking privilege to define physical database resources, see REVOKE Physical Database Definition Privileges (see page 326).

GRANT SQL Definition Privileges

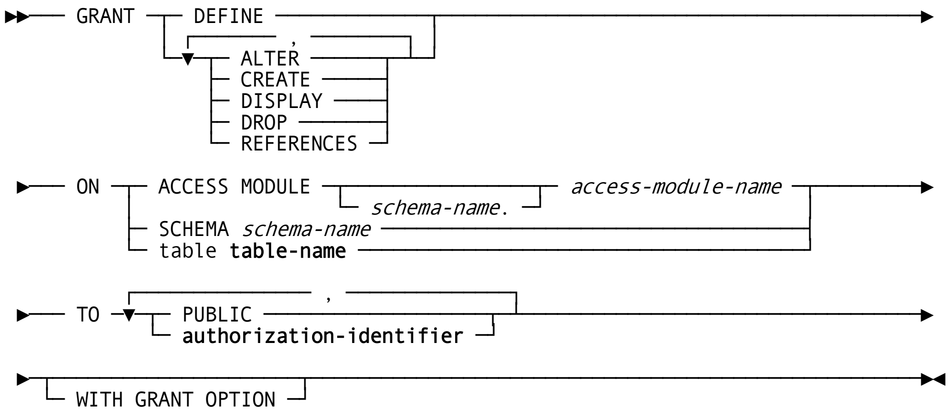
Gives one or more users or groups the privilege of performing definition functions on a specified access module, schema, table, function, procedure, table procedure, or view.

GRANT SQL Definition Privileges Authorization

To grant definition privileges on SQL-defined database resources, one of the following must be true:

- You hold the corresponding grantable privilege on the resource (you can grant the privilege, but you cannot specify WITH GRANT OPTION).
- You own the resource.
- You hold DBADMIN on the dictionary containing the definition.
- You hold SYSADMIN privilege.

GRANT SQL Definition Privileges Syntax



GRANT SQL Definition Privileges Parameters

DEFINE

Gives the ALTER, CREATE, DISPLAY, and DROP privileges, as applicable, on the resource identified in the ON parameter to the users or groups identified in the TO parameter.

ALTER

Gives the ALTER privilege on the resource identified in the ON parameter to the users or groups identified in the TO parameter.

The ALTER privilege on a resource allows a user to modify the definition of the resource.

CREATE

Gives the CREATE privilege on the resource identified in the ON parameter to the users or groups identified in the TO parameter.

The CREATE privilege on a resource allows a user to define the resource.

DISPLAY

Gives the DISPLAY privilege on the resource identified in the ON parameter to the users or groups identified in the TO parameter.

The DISPLAY privilege allows the user to issue a DISPLAY RESOURCE statement on the named resource. The grantable DISPLAY privilege allows a user to issue a DISPLAY PRIVILEGES statement on the named resource.

The DISPLAY privilege on *access-module-name* also allows a user to execute the EXPLAIN statement on the access module.

DROP

Gives the DROP privilege on the resource identified in the ON parameter to the users or groups identified in the TO parameter.

The DROP privilege on a resource allows a user to delete the definition of the resource.

REFERENCES

Gives the REFERENCES privilege on the table identified in the ON parameter to the users or groups identified in the TO parameter.

The REFERENCES privilege on a table allows a user to define referential constraints in which the named table is the referenced table.

Note: The REFERENCES privilege applies only to tables. It does not apply to functions, procedures, table procedures, views, access modules, or schemas.

ON

Specifies the resource to which the definition privileges apply.

ACCESS MODULE

Specifies that the privileges apply to any version of *access-module-name* in the associated schema.

access-module-name

Identifies the access module.

You can wildcard *access-module-name*. If you specify *schema-name*, the wildcard character is valid after the period following *schema-name*.

Note: For more information about wildcarding, see Using a Wildcard (see page 76).

schema-name

Identifies the schema associated with *access-module-name*.

If you do not specify *schema-name*, it defaults to the current schema in effect for your session.

Note: For more information about using a schema name to qualify an access module name, see the *CA IDMS SQL Reference Guide*.

SCHEMA *schema-name*

Identifies an SQL schema.

You can wildcard *schema-name* in the SCHEMA parameter.

Note: For more information about wildcarding, see Using a Wildcard (see page 76).

table table-name

Identifies a table-like object

Note: Expanded syntax for **table-name** is presented in Notes on Security Statement Syntax.

You can wildcard any identifier when you grant definition privileges on **table-name**. If you specify *schema-name* in **table-name**, the wildcard character is valid after the period following *schema-name*.

Note: For more information about wildcarding, see Using a Wildcard (see page 76).

If you do not specify *schema-name*, it defaults to the current schema in effect for your session.

TO

Specifies the users to whom you are giving the definition privileges.

PUBLIC

Specifies all users.

authorization-identifier

Identifies a user or group.

Note: Expanded syntax for **authorization-identifier** is presented in Notes on Security Statement Syntax (see page 225).

WITH GRANT OPTION

Gives the privilege of granting the specified definition privileges on the named resource to the users or groups identified in the TO parameter.

A privilege granted with the WITH GRANT OPTION is called a grantable privilege.

GRANT SQL Definition Privileges Usage

Wildcarding Table-Name

For table definition privileges, only the closest matching wildcarded grant is used. If CREATE privilege has been granted on HR.EMP* and HR.EMPV*, then only the grant on HR.EMPV* is used to verify the privilege to create HR.EMPVU_SALARY.

The DEFINE Keyword

When you use the DEFINE keyword with a GRANT statement, you grant a set of definition privileges to one or more users or groups.

When you use the DEFINE keyword with a REVOKE statement, you revoke all of the privileges in the set that have been previously granted to the specified users or groups.

This means that if you GRANT CREATE privilege on a resource, you can revoke the privilege with either a REVOKE CREATE statement or a REVOKE DEFINE statement. Using REVOKE DEFINE is an efficient technique when you intend to revoke all definition privileges from a user or group, whether the privileges were granted singly or as a set.

Similarly, you can GRANT DEFINE on a resource to a user and then REVOKE DROP on the resource from the same user as a way to grant all but one definition privilege.

Granting Privileges on a Schema

The following GRANT statement gives the ALTER, CREATE, DISPLAY, and DROP privileges on all schemas that begin with 'DSF' to user DSF. The statement also gives user DSF the privilege of granting the same privileges to other users.

```
grant define
  on schema dsf*
  to dsf
  with grant option;
```

GRANT SQL Definition Privileges More Information

For more information about revoking SQL definition privileges, see REVOKE SQL Definition Privileges (see page 330).

GRANT Table Access Privileges

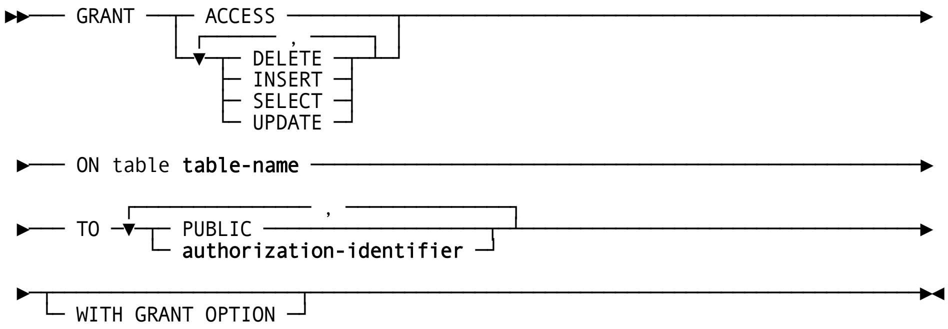
Gives one or more users or groups the privilege of accessing a specified table-like object in a specified way.

GRANT Table Access Privileges Authorization

To grant table-like object access privileges, one of the following must be true:

- You hold the corresponding grantable privilege on the table-like object (you can grant the privilege, but you cannot specify WITH GRANT OPTION).
- You own the table-like object.
- You hold DBADMIN privilege on the database that contains the table-like object data.
- You hold SYSADMIN privilege.

GRANT Table Access Privileges Syntax



GRANT Table Access Privileges Parameters

ACCESS

Gives the DELETE, INSERT, SELECT, and UPDATE privileges on the table-like object identified in the ON parameter to the users or groups identified in the TO parameter.

DELETE

Gives the DELETE privilege on the table-like object identified in the ON parameter to the users or groups identified in the TO parameter.

The DELETE privilege on a table-like object allows a user to delete rows from the table or view.

INSERT

Gives the INSERT privilege on the table-like object identified in the ON parameter to the users or groups identified in the TO parameter.

The INSERT privilege on a table-like object allows a user to insert rows into the table or view.

SELECT

Gives the SELECT privilege on the table-like object identified in the ON parameter to the users or groups identified in the TO parameter.

The SELECT privilege on a table-like object allows a user to the following:

- Retrieve data from the table-like object.
- Name the table-like object in a subquery.
- Define a view derived from the table-like object.

UPDATE

Gives the UPDATE privilege on the table-like object identified in the ON parameter to the users or groups identified in the TO parameter.

The UPDATE privilege on a table-like object allows a user to modify data in the table-like object.

ON table table-name

Specifies the table-like object to which the table access privileges apply.

Note: Expanded syntax for **table-name** is presented in Notes on Security Statement Syntax (see page 225).

You can wildcard any identifier when you grant access on **table-name**. If you specify *schema-name* in **table-name**, the wildcard character is valid after the period following *schema-name*.

Note: For more information about wildcarding, see Using a Wildcard (see page 76).

If you do not specify *schema-name*, it defaults to the current schema in effect for your session.

TO

Specifies the users or groups to whom you are giving table access privileges.

PUBLIC

Specifies all users.

authorization-identifier

Identifies a user or group.

Note: Expanded syntax for **authorization-identifier** is presented in chapter Notes on Security Statement Syntax (see page 225).

WITH GRANT OPTION

Gives the privilege of granting the specified access privileges on the named table or view to the users or groups identified in the TO parameter.

A privilege granted with the WITH GRANT OPTION is called a grantable privilege.

GRANT Table Access Privileges Usage

Verifying Privileges Granted with Wildcards

For table access privileges, all matching wildcarded grants are used. If SELECT privilege has been granted on HR.EMP* and HR.EMPV*, then users or groups receiving either the HR.EMP* or the HR.EMPV* grant are authorized to select from EMPVU_SALARY.

This differs from the use of wildcards for all other types of resources or privileges, where only the closest matching wildcarded grant of privilege is used to verify the user's authorization.

The ACCESS Keyword

When you use the ACCESS keyword with a GRANT statement, you grant a set of access privileges on a table-like object to one or more users or groups.

When you use the ACCESS keyword with a REVOKE statement, you revoke any access privileges that have been previously granted on the table-like object from the specified users or groups.

This means that if you GRANT SELECT privilege on a table, you can revoke the privilege with either a REVOKE SELECT statement or a REVOKE ACCESS statement. Using REVOKE ACCESS is an efficient technique when you intend to revoke all access privileges on a table from a user or group, whether the privileges were granted singly or as a set.

Similarly, you can GRANT ACCESS on a table to a user and then REVOKE DELETE on the table from the same user as a way to grant all but one table access privilege.

Granting Selected Privileges on a Table

The following GRANT statement gives the SELECT and UPDATE privileges on the EMPLOYEE table associated with the current schema to users KRP, SAE, and PGD:

```
grant select, update
  on employee
  to krp, sae, pgd;
```

GRANT Table Access Privileges More Information

For more information about revoking table access privileges, see REVOKE Table Access Privileges (see page 333).

REVOKE Access Module Execution Privilege

Revokes from one or more users or groups the privilege of executing a specified access module.

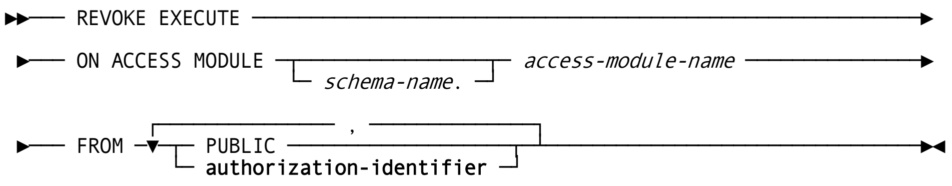
REVOKE Access Module Execution Privilege Authorization

To revoke access module execution privilege, one of the following must be true:

- You hold grantable execution privilege on the access module (you can grant the privilege, but you cannot specify WITH GRANT OPTION).
- You own the schema associated with the access module.
- You hold DBADMIN privilege on the dictionary that contains the access module.
- You hold SYSADMIN privilege.

You must be connected to the application dictionary that contains the access module when you issue the statement.

REVOKE Access Module Execution Privilege Syntax



REVOKE Access Module Execution Privilege Parameters

EXECUTE

Specifies that you are revoking execution privilege on the access module identified in the ON parameter from the users or group identified in the FROM parameter.

ON ACCESS MODULE

Specifies the access module to which execution privilege applies.

access-module-name

Identifies the access module.

You can wildcard *access-module-name*. If you specify *schema-name*, the wildcard character is valid after the period following *schema-name*.

Note: For more information about wildcarding, see Using a Wildcard (see page 76).

schema-name

Identifies the schema with which *access-module-name* is associated.

If you do not specify *schema-name*, it defaults to the current schema associated with your session.

Note: For more information about using a schema name to qualify an access module name, see the *CA IDMS SQL Reference Guide*.

FROM

Specifies the users or groups from whom you are revoking execution privilege.

PUBLIC

Specifies all users.

The privilege must have been previously given to PUBLIC by means of the GRANT statement.

authorization-identifier

Identifies a user or group.

The privilege must have been previously given to *authorization-identifier* by means of the GRANT statement.

Note: Expanded syntax for *authorization-identifier* is presented in chapter Notes on Security Statement Syntax (see page 225).

Revoking Execution Privilege

The following statement revokes execution privilege on all access modules associated with schema HR that begin with 'HR.EMP' from group PER_GRP_2:

```
revoke execute
  on access module hr.emp*
  from per_grp_2;
```

For more information about granting execution privilege, see GRANT Access Module Execution Privilege (see page 295).

REVOKE Administration Privilege

Revokes from one or more users or groups the DBADMIN privilege.

Authorization

To revoke DBADMIN privilege, you must hold one of these privileges:

- DBADMIN on the database
- SYSADMIN

You must be connected to the system dictionary when you issue the statement.

Syntax

```
►►── REVOKE DBADMIN ON DB database-name ─────────────────────────────────►

           ┌────────────────── , ───────────────┐
►── FROM ─▼─┬─ PUBLIC ─────────────────┬────────┴─────────────────────────►◄
            └─ authorization-identifier ┘
```

Parameters

DBADMIN

Specifies that you are revoking DBADMIN privilege on the database identified in the ON parameter from the users or groups identified in the FROM parameter.

ON DB *database-name*

Specifies database to which DBADMIN privilege applies.

Database-name refers to either a segment or a name in the database name table.

Note: For more information about the DBADMIN privilege, see Securing Database Resources (see page 117).

You can wildcard *database-name*.

Note: For more information about wildcarding, see Using a Wildcard (see page 76).

FROM

Specifies the users or groups from whom you are revoking DBADMIN privilege.

PUBLIC

Specifies all users.

The privilege must have been previously given to PUBLIC by means of the GRANT statement.

authorization-identifier

Identifies a user or group.

The privilege must have been previously given to **authorization-identifier** by means of the GRANT statement.

Note: Expanded syntax for **authorization-identifier** is presented in Notes on Security Statement Syntax (see page 225).

Revoking DBADMIN From the DBA

The following statement revokes DBADMIN privilege on database GLDB from the DBA group ID:

```
revoke dbadmin
  on db gldb
   from dba_gldb;
```

More Information

For more information **about granting DBADMIN privilege**, see GRANT Administration Privilege (see page 298).

REVOKE All Table Privileges

Revokes from one or more users or groups all definition and access privileges on a specified table-like object.

Authorization

To revoke all table privileges, one of the following must be true:

- You hold all privileges as grantable privileges on the table-like object.
- You own the table-like object.
- You hold DBADMIN privileges for the dictionary that contains the table-like object definition and the database that contains the table-like object data.
- You hold SYSADMIN privilege.

Syntax

```
►►── REVOKE ALL PRIVILEGES ─────────────────────────────────────────────►

►── ON table table-name ────────────────────────────────────────────────►

                ┌──────────────── , ──────────────┐
►── FROM ─▼─┬─ PUBLIC─────────────────────────┬┴──────────────────────────►◄
             └─ authorization-identifier ─────┘
```

Parameters

ALL PRIVILEGES

Revokes the DELETE, INSERT, SELECT, UPDATE, ALTER, CREATE, DROP, and REFERENCES privileges, as applicable, on the table-like object identified in the ON parameter from the users or groups identified in the FROM parameter.

ON table table-name

Identifies the table-like object to which the privileges apply.

Note: Expanded syntax for **table-name** is presented in Notes on Security Statement Syntax.

You can wildcard the identifier of **table-name**. If you specify *schema-name* in **table-name**, the wildcard character is valid after the period following *schema-name*.

Note: For more information about wildcarding, see Using a Wildcard (see page 76).

If you do not specify *schema-name*, it defaults to the current schema in effect for your session.

FROM

Specifies the users or groups from whom you are revoking table privileges.

PUBLIC

Specifies all users.

The privileges must have been previously given to PUBLIC by means of the GRANT statement.

authorization-identifier

Identifies a user or group.

The privileges must have been previously given to **authorization-identifier** by means of the GRANT statement.

Note: Expanded syntax for **authorization-identifier** is presented in Notes on Security Statement Syntax (see page 225).

Revoking All Privileges From All Users

The following statement revokes all privileges on all table-like objects in the TEST schema from the group PUBLIC:

```
revoke all privileges
  on test.*
  from public;
```

More Information

- For more information about table access privileges, see GRANT Table Access Privileges (see page 315) and REVOKE Table Access Privileges (see page 333).
- For more information about table definition privileges, see GRANT SQL Definition Privileges (see page 311) and REVOKE SQL Definition Privileges (see page 330).

REVOKE Area Access Privileges

Revokes from one or more users or groups access to an area.

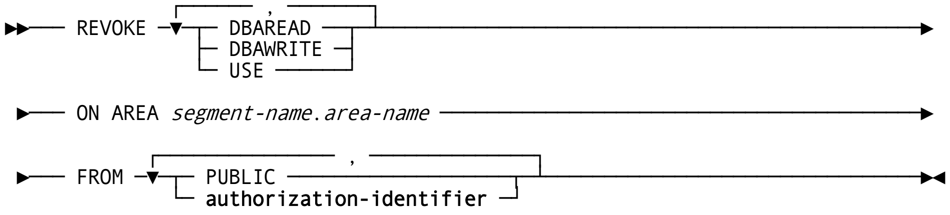
Authorization

To revoke an area access privilege, you must hold one of these privileges:

- The grantable area access privilege on the area
- DBADMIN on DB *segment-name*
- SYSADMIN

You must be connected to the system dictionary when you issue the statement.

Syntax



```
►►── REVOKE ─▼┬─ DBAREAD ──┬───────────────────────────────────────►
              ├─ DBAWRITE ─┤   (repeat, separated by ",")
              └─ USE ──────┘

►── ON AREA segment-name.area-name ────────────────────────────────►

►── FROM ─▼┬─ PUBLIC ──────────────────┬──────────────────────────►◄
           └─ authorization-identifier ┘   (repeat, separated by ",")
```

Parameters

DBAREAD

Specifies that you are revoking DBAREAD privilege on the area named in the ON parameter from the users or groups named in the FROM parameter.

DBAWRITE

Specifies that you are revoking DBAWRITE privilege on the area named in the ON parameter from the users or groups named in the FROM parameter.

USE

Specifies that you are revoking USE privilege on the area named in the ON parameter from the users or groups named in the FROM parameter.

ON AREA *segment-name.area-name*

Identifies the area to which the specified area access privileges apply.

You can wildcard *area-name* when you revoke area access privileges. You cannot wildcard *segment-name*. The wildcard character is valid after the period following *segment-name*.

Note: For more information about wildcarding, see Using a Wildcard (see page 76).

FROM

Specifies the users or groups from whom you are revoking area access privileges.

PUBLIC

Specifies all users.

The privileges must have been previously given to PUBLIC by means of the GRANT statement.

authorization-identifier

Identifies a user or group.

The privileges must have been previously given to **authorization-identifier** by means of the GRANT statement.

Note: Expanded syntax for **authorization-identifier** is presented in Notes on Security Statement Syntax (see page 225).

Revoking Area Access Privileges

The following statement revokes an area access privilege from the specified user:

```
revoke dbawrite
  on area gl."account-area"
  from alex;
```

Note: For more information about granting the privilege to access an area, see GRANT Area Access Privileges (see page 301).

REVOKE Non-SQL Definition Privilege

Revokes from one or more users or groups the privilege of referencing a non-SQL-defined schema in an SQL schema definition.

Authorization

To revoke USE privilege on a non-SQL-defined schema, you must hold one of these privileges:

- Grantable USE privilege on the non-SQL-defined schema
- DBADMIN on the dictionary containing the non-SQL schema definition
- SYSADMIN

You must be connected to the dictionary containing the non-SQL-defined schema when you issue the statement.

Syntax

```
►►─── REVOKE ─┬─ USE ─────┬──────────────────────────────────────────────►
              └─ DISPLAY ─┘

►─── ON NONSQL SCHEMA Vnnnn.nonsql-schema-name ──────────────────────────►

               ┌───────────────── , ─────────────┐
►─── FROM ─▼─┬─ PUBLIC ────────────────────┬─┴──────────────────────────►◄
              └─ authorization-identifier ─┘
```

Parameters

USE

Specifies that you are revoking the USE privilege on the non-SQL-defined schema identified in the ON parameter to the users or groups identified in the FROM parameter.

DISPLAY

Specifies that you are revoking the DISPLAY privilege on the non-SQL-defined schema identified in the ON parameter to the users or groups identified in the FROM parameter.

ON NONSQL SCHEMA

Specifies the non-SQL-defined schema to which the USE privilege applies.

V*nnnn.nonsql-schema-name*

Specifies the version number and name of the non-SQL-defined schema. The version number (*nnnn*) must include leading zeros.

You can wildcard *nonsql-schema-name*. You cannot wildcard V*nnnn*. The wildcard character is valid after the period following V*nnnn*.

Note: For more information about wildcarding, see Using a Wildcard (see page 76).

FROM

Specifies the users or groups from whom you are revoking the USE privilege.

PUBLIC

Specifies all users.

The privilege must have been previously given to PUBLIC by means of the GRANT statement.

authorization-identifier

Identifies a user or group.

The privilege must have been previously given to **authorization-identifier** by means of the GRANT statement.

Note: Expanded syntax for **authorization-identifier** is presented in Notes on Security Statement Syntax (see page 225).

Revoking Use of a Non-SQL-Defined Schema

The following statement revokes the privilege of referencing a non-SQL-defined schema when creating an SQL schema:

```
revoke use
  on nonsql schema v0001.ap
  from sam;
```

More Information

For more information **about granting the privilege to use a non-SQL-defined schema**, see GRANT Non-SQL Definition Privilege (see page 303).

REVOKE Physical Database Definition Privileges

Revokes from one or more users or groups the privilege of executing the DMCL, DBTABLE, and SEGMENT physical DDL statements.

Authorization

To revoke a definition privilege on a DMCL or DBTABLE, you must hold one of these privileges:

- The corresponding grantable privilege
- DBADMIN on DB SYSTEM
- SYSADMIN

To revoke a physical definition privilege on a database, you must hold one of these privileges:

- The corresponding grantable privilege
- DBADMIN on the database
- SYSADMIN

You must be connected to the system dictionary when you issue the statement.

Syntax

```
►►─── REVOKE ─┬─ DEFINE ────────────────┬──────────────────────────────────►
              │ ┌──────── , ────────┐   │
              └─▼─┬─ ALTER ─────────┬─┴───┘
                  ├─ CREATE ────────┤
                  ├─ DISPLAY ───────┤
                  ├─ DROP ──────────┤
                  └─ USE ───────────┘

►─── ON ──┬─ DMCL dmcl-name ─────────┬─────────────────────────────────────►
          ├─ DBTABLE dbtable-name ───┤
          └─ DB database-name ───────┘

               ┌──────────────── , ────────────────┐
►─── FROM ─────▼─┬─ PUBLIC ───────────────────┬────┴──────────────────────►◄
                 └─ authorization-identifier ─┘
```

Parameters

DEFINE

Revokes the ALTER, CREATE, DISPLAY, DROP, and USE privileges on the resource identified in the ON parameter from the users or groups identified in the FROM parameter.

ALTER

Revokes the ALTER privilege on the resource identified in the ON parameter from the users or groups identified in the FROM parameter.

CREATE

Revokes the CREATE privilege on the resource identified in the ON parameter from the users or groups identified in the FROM parameter.

DISPLAY

Revokes the DISPLAY privilege on the resource identified in the ON parameter from the users or groups identified in the FROM parameter.

DROP

Revokes the DROP privilege on the resource identified in the ON parameter from the users or groups identified in the FROM parameter.

USE

Revokes the USE privilege on the resource identified in the ON parameter from the users or groups identified in the FROM parameter.

ON

Specifies the resource to which the definition privileges apply.

DMCL *dmcl-name*

Identifies a DMCL.

You can wildcard *dmcl-name*.

Note: For more information about wildcarding, see Using a Wildcard (see page 76).

DBTABLE *dbtable-name*

Identifies a database name table.

You can wildcard *dbtable-name*.

Note: For more information about wildcarding, see Using a Wildcard (see page 76).

.

DB *database-name*

Identifies a segment or a name in the database name table.

You can wildcard *database-name*.

Note: For more information about wildcarding, see Using a Wildcard (see page 76).

FROM

Specifies the users or groups from whom you are revoking definition privileges.

PUBLIC

Specifies all users.

The privileges must have been previously given to PUBLIC by means of the GRANT statement.

authorization-identifier

Identifies a user or group.

The privileges must have been previously given to **authorization-identifier** by means of the GRANT statement.

Note: Expanded syntax for **authorization-identifier** is presented in Notes on Security Statement Syntax (see page 225).

Usage

The DEFINE Keyword

When you use the DEFINE keyword with a GRANT statement, you grant a set of definition privileges on a resource to one or more users or groups.

When you use the DEFINE keyword with a REVOKE statement, you revoke all definition privileges that have been previously granted on the resource from the specified users or groups.

This means that if you GRANT CREATE privilege on a resource, you can revoke the privilege with either a REVOKE CREATE statement or a REVOKE DEFINE statement. Using REVOKE DEFINE is an efficient technique when you intend to revoke all definition privileges on the resource from a user or group, whether the privileges were granted singly or as a set.

Similarly, you can GRANT DEFINE on a resource to a user and then REVOKE DROP on the resource from the same user as a way to grant all but one definition privilege.

Revoking Privilege to Issue DMCL Statements

The following statement revokes from the user the privilege to issue DMCL definition statements for DMCL99:

```
revoke define
  on dmcl dmcl99
  from ryan;
```

More Information

For more information **about granting privilege to define physical database resources**, see GRANT Physical Database Definition Privileges (see page 305).

REVOKE SQL Definition Privileges

Revokes from one or more users or groups the privilege of performing selected actions on a specified schema, access module, or table-like object.

Authorization

To revoke a definition privilege for an SQL-defined database resource, one of the following must be true:

- You hold the grantable definition privilege on the resource
- You own the resource
- You hold DBADMIN privilege on the dictionary containing the definitions
- You hold SYSADMIN privilege

Syntax

```
►►─── REVOKE ───┬── DEFINE ─────────────────┬──────────────────────────────►
                │ ┌────────── , ─────────┐  │
                └─▼─┬── ALTER ───────────┴──┘
                    ├── CREATE ──────────┤
                    ├── DISPLAY ─────────┤
                    ├── DROP ────────────┤
                    └── REFERENCES ──────┘

 ►─── ON ──┬── ACCESS MODULE ─┬────────────────┬── access-module-name ─┬────►
           │                  └─ schema-name. ─┘                       │
           ├── SCHEMA schema-name ─────────────────────────────────────┤
           └── table table-name ───────────────────────────────────────┘

 ►─── FROM ─▼─┬── PUBLIC ──────────────────┬── , ──┬──────────────────────►◄
            │ └── authorization-identifier ┘       │
            └──────────────────────────────────────┘
```

Parameters

DEFINE

Revokes the ALTER, CREATE, DISPLAY, and DROP privileges on the resource identified in the ON parameter from the users or groups identified in the FROM parameter.

ALTER

Revokes the ALTER privilege on the resource identified in the ON parameter from the users or groups identified in the FROM parameter.

CREATE

Revokes the CREATE privilege on the resource identified in the ON parameter from the users or groups identified in the FROM parameter.

DISPLAY

Revokes the DISPLAY privilege on the resource identified in the ON parameter from the users or groups identified in the FROM parameter.

DROP

Revokes the DROP privilege on the resource identified in the ON parameter from the users or groups identified in the FROM parameter.

REFERENCES

Revokes the REFERENCES privilege on the table identified in the ON parameter from the users or groups identified in the FROM parameter.

ON

Specifies the resource to which the definition privileges apply.

ACCESS MODULE *access-module-name*

Identifies an access module.

Privileges on any version of *access-module-name* in the associated schema are revoked.

You can wildcard *access-module-name*. If you specify *schema-name*, the wildcard character is valid after the period following *schema-name*.

Note: For more information about wildcarding, see Using a Wildcard (see page 76).

schema-name

Identifies the schema associated with *access-module-name*.

Note: For more information about using a schema name to qualify an access module name, see the *CA IDMS SQL Reference Guide*.

If you do not specify *schema-name*, it defaults to the current schema in effect for your session.

SCHEMA *schema-name*

Identifies an SQL schema.

You can wildcard *schema-name*.

Note: For more information about wildcarding, see Using a Wildcard (see page 76).

table table-name

Identifies a table-like object.

You can wildcard any identifier in **table-name** when you grant definition privileges on **table-name**. If you specify *schema-name* in **table-name**, the wildcard character is valid after the period following *schema-name*.

Note: For more information about wildcarding, see Using a Wildcard (see page 76).

If you do not specify *schema-name*, it defaults to the current schema in effect for your session.

FROM

Specifies the users or groups from whom you are revoking definition privileges.

PUBLIC

Specifies all users.

The privileges must have been previously given to PUBLIC by means of the GRANT statement.

authorization-identifier

Identifies a user or group.

The privileges must have been previously given to **authorization-identifier** by means of the GRANT statement.

Note: Expanded syntax for **authorization-identifier** is presented in Notes on Security Statement Syntax (see page 225).

Usage

The DEFINE Keyword

When you use the DEFINE keyword with a GRANT statement, you grant a set of definition privileges on a resource to one or more users or groups.

When you use the DEFINE keyword with a REVOKE statement, you revoke all definition privileges that have been previously granted on the resource from the specified users or groups.

This means that if you GRANT CREATE privilege on a resource, you can revoke the privilege with either a REVOKE CREATE statement or a REVOKE DEFINE statement. Using REVOKE DEFINE is an efficient technique when you intend to revoke all definition privileges on the resource from a user or group, whether the privileges were granted singly or as a set.

Similarly, you can GRANT DEFINE on a resource to a user and then REVOKE DROP on the resource from the same user as a way to grant all but one definition privilege.

Revoking Privileges on a Schema

The following statement revokes the ALTER, CREATE, DISPLAY, and DROP privileges on all schemas that begin with 'DSF' from user DSF:

```
revoke define
  on schema dsf*
  from dsf;
```

More Information

For more information **about granting SQL definition privileges**, see GRANT SQL Definition Privileges (see page 311).

REVOKE Table Access Privileges

Revokes from one or more users or groups the privilege of accessing a specified table-like object in a specified way.

Authorization

To revoke a table-like object access privilege, one of the following must be true:

- You hold the corresponding grantable privilege on the table-like object.
- You own the table-like object.
- You hold DBADMIN privilege on the database that contains the table-like object.
- You hold SYSADMIN privilege.

Syntax

```
►►─── REVOKE ──┬── ACCESS ───────────────────────────────────────────────►
               │  ┌─────── , ─────┐ │
               └─▼┬─ DELETE ─┬────┴─┘
                  ├─ INSERT ─┤
                  ├─ SELECT ─┤
                  └─ UPDATE ─┘

►─── ON table table-name ────────────────────────────────────────────────►

             ┌──────────────── , ────────────┐
►─── FROM ─▼┬─ PUBLIC ───────────────────────┴──────────────────────────►◄
            └─ authorization-identifier ─┘
```

Parameters

ACCESS

Revokes the DELETE, INSERT, SELECT, and UPDATE privileges on the table-like object identified in the ON parameter from the users or groups identified in the FROM parameter.

DELETE

Revokes the DELETE privilege on the table-like object identified in the ON parameter from the users or groups identified in the FROM parameter.

INSERT

Revokes the INSERT privilege on the table-like object identified in the ON parameter from the users or groups identified in the FROM parameter.

SELECT

Revokes the SELECT privilege on the table-like object identified in the ON parameter from the users or groups identified in the FROM parameter.

UPDATE

Revokes the UPDATE privilege on the table-like object identified in the ON parameter from the users or groups identified in the FROM parameter.

ON table table-name

Identifies the table-like object to which the access privileges apply.

Note: Expanded syntax for **table-name** is presented in Notes on Security Statement Syntax.

You can wildcard any identifier in **table-name** when you grant access on **table-name**. If you specify *schema-name* in **table-name**, the wildcard character is valid after the period following *schema-name*.

Note: For more information about wildcarding, see Using a Wildcard (see page 76).

If you do not specify *schema-name*, it defaults to the current schema in effect for your session.

FROM

Specifies the users or groups from whom you are revoking the specified table access privileges.

PUBLIC

Specifies all users.

The privileges must have been previously given to PUBLIC by means of the GRANT statement.

authorization-identifier

Identifies a user or group.

The privileges must have been previously given to **authorization-identifier** by means of the GRANT statement.

Note: Expanded syntax for **authorization-identifier** is presented in Notes on Security Statement Syntax (see page 225).

Usage

The ACCESS Keyword

When you use the ACCESS keyword with a GRANT statement, you grant a set of access privileges on a table-like object to one or more users or groups.

When you use the ACCESS keyword with a REVOKE statement, you revoke any access privileges that have been previously granted on the table or view from the specified users or groups.

This means that if you GRANT SELECT privilege on a table, you can revoke the privilege with either a REVOKE SELECT statement or a REVOKE ACCESS statement. Using REVOKE ACCESS is an efficient technique when you intend to revoke all access privileges on a table from a user or group, whether the privileges were granted singly or as a set.

Similarly, you can GRANT ACCESS on a table to a user and then REVOKE DELETE on the table from the same user as a way to grant all but one table access privilege.

Revoking Selected Privileges on a Table

The following statement revokes the SELECT and UPDATE privileges on the EMPLOYEE table associated with the current schema from users KRP, SAE, and PGD:

```
revoke select, update
  on employee
  from krp, sae, pgd;
```

More Information

For more information **about granting table access privileges**, see GRANT Table Access Privileges (see page 315).

TRANSFER OWNERSHIP

Passes ownership of an SQL schema from one user or group to another user or group.

Authorization

To transfer ownership, one of the following must be true:

- You own the schema
- You hold DBADMIN privilege on the dictionary in which the schema is defined
- You hold SYSADMIN privilege

Syntax

```
►►─── TRANSFER OWNERSHIP OF SCHEMA schema-name ──────────────────────────►

 ►─── TO authorization-identifier ─────────────────────────────────────►◄
```

Parameters

OF SCHEMA *schema-name*

Specifies the schema whose ownership is being transferred to the user or group identified in the TO parameter.

Schema-name must identify a schema defined in the dictionary.

TO authorization-identifier

Identifies a user or group.

Note: Expanded syntax for **authorization-identifier** is presented in Notes on Security Statement Syntax (see page 225).

Usage

Schema Ownership

At any given time, a schema can be owned by one user or group. The initial owner is the user who created the schema. When ownership of a schema is transferred to a group, each user in the group has all the privileges associated with ownership.

Ownership of Other Resources

Technically, schemas are the only database resources that users own. However, by association, the user or group that owns a schema is also said to own the tables, functions, procedures, table procedures, views, and access modules in the schema.

Ownership Privileges

The owner of a schema has all applicable privileges on resources in the schema, as well as the privilege of granting those privileges to other users or groups. If you transfer ownership of a schema to another user or group, you no longer own the resources in the schema.

Transferring Ownership to a Single User

The following TRANSFER OWNERSHIP statement transfers ownership of the PKE_SCH schema to user PKE:

```
transfer ownership of schema pke_sch
   to pke;
```

Transferring Ownership to a Group

The following TRANSFER OWNERSHIP statement transfers ownership of the SALES schema to the SALES_GRP group:

```
transfer ownership of schema sales
   to sales_grp;
```

For more information **about creating a schema**, see the "CREATE SCHEMA" in the *CA IDMS SQL Reference Guide*.

Chapter 16: Syntax for Security Display Statements

This section contains the following topics:

Notes on DISPLAY/PUNCH Statement Syntax

How to Submit Statements

You submit security DISPLAY statements through the CA IDMS Command Facility in online mode or batch mode,

What DISPLAY Statements Do

DISPLAY statements enable you to display resource definitions in a CA IDMS centralized security database and privileges granted on resources.

What PUNCH Statements Do

PUNCH statements submitted online display the output online as DISPLAY statements do.

PUNCH statements submitted in batch mode write the output to the SYSPCH file.

Common Parameters

The following parameters are valid in all DISPLAY statements:

WITh display-option

Specifies that output is to include information represented by *display-option*.

ALSo WITh display-option

Specifies that output is also to include information represented by *display-option*.

WIThout display-option

Specifies that output is to exclude information represented by *display-option*.

AS SYNtax

Specifies that lines of output representing executable syntax do not begin with comment characters.

AS COMments

Specifies that lines of output representing executable syntax begin with comment characters.

Usage

Wildcards

If a wildcarded name was used in the resource definition or the grant of privilege on the resource, the DISPLAY statement must specify the wildcarded name.

Correspondence to DDDL Syntax

The preceding common parameters for DISPLAY statements have the same function that they have in DDDL syntax. In general, the conventions of security DISPLAY statement syntax will be familiar to the DDDL user.

Note: For more information about DDDL, see the *CA IDMS IDD DDDL Reference Guide*.

Specifying AS SYNTAX

This example shows the output of a security DISPLAY statement that specifies AS SYNTAX:

```
DISPLAY USER DICK WITH ALL AS SYNTAX;
   CREATE USER DICK
*+     USER IS ACTIVE
      DESCRIPTION 'CHIEF ANALYST'
      NAME 'RICHARD A. ANALYST'
*+     PASSWORD ASSIGNED
      PROFILE PAYROLL
*+     CREATED 1991-07-18-12.51.17.187373 BY KKS
*+     LAST UPDATED 1991-07-18-12.51.23.000517 BY KKS
*+     WITHIN GROUP DEVTEAM
*+     HOLDS DEFINE PRIVILEGES ON USER TOM
*+     HOLDS DEFINE PRIVILEGES ON SYSTEM SYSTEM71
*+     HOLDS SIGNON PRIVILEGES ON SYSTEM SYSTEM71
*+     HOLDS DBADMIN PRIVILEGES ON DB FRED
*+     HOLDS DBAWRITE PRIVILEGES ON AREA THESEGMENT.CORPTSP
*+     HOLDS USE PRIVILEGES ON DMCL THEDMCL
*+     HOLDS USE PRIVILEGES ON DBTABLE THEDBTABLE
*+     HOLDS USE PRIVILEGES ON DB THEDB
*+     HOLDS EXECUTE PRIVILEGES ON CATEGORY CATE3
*+     HOLDS EXECUTE PRIVILEGES ON ACTIVITY DCMT.N002
*+     HOLDS REFERENCES PRIVILEGES ON ACCESS MODULE THESCHEMA.THEAM
*+     HOLDS EXECUTE PRIVILEGES ON ACCESS MODULE THESCHEMA.THEAM
*+     HOLDS ALL PRIVILEGES ON TABLE THESCHEMA.THE*
*+     HOLDS DISPLAY PRIVILEGES ON ACCESS MODULE THESCHEMA.*
*+     HOLDS DISPLAY PRIVILEGES ON ACCESS MODULE THESCHEMA.FRED*
*+     HOLDS REFERENCES PRIVILEGES ON TABLE THESCHEMA.THETABLE
*+     HOLDS SELECT, INSERT, UPDATE PRIVILEGES ON TABLE THESCHEMA.THETABLE
*+     HOLDS DISPLAY PRIVILEGES ON TABLE THESCHEMA.*
      ;
```

Specifying AS COMMENTS

This example shows the output of a security DISPLAY statement that specifies AS COMMENTS:

```
DISPLAY USER DICK WITH ALL AS COMMENTS;
*+   CREATE USER DICK
*+     USER IS ACTIVE
*+     DESCRIPTION 'CHIEF ANALYST'
*+     NAME 'RICHARD A. ANALYST'
*+     PASSWORD ASSIGNED
*+     PROFILE PAYROLL
*+     CREATED 1991-07-18-12.51.17.187373 BY KKS
*+     LAST UPDATED 1991-07-18-12.51.23.000517 BY KKS
*+     WITHIN GROUP DEVTEAM
*+      HOLDS DEFINE PRIVILEGES ON USER TOM
*+      HOLDS DEFINE PRIVILEGES ON SYSTEM SYSTEM71
*+      HOLDS SIGNON PRIVILEGES ON SYSTEM SYSTEM71
*+      HOLDS DBADMIN PRIVILEGES ON DB FRED
*+      HOLDS DBAWRITE PRIVILEGES ON AREA THESEGMENT.CORPTSP
*+      HOLDS USE PRIVILEGES ON DMCL THEDMCL
*+      HOLDS USE PRIVILEGES ON DBTABLE THEDBTABLE
*+      HOLDS USE PRIVILEGES ON DB THEDB
*+      HOLDS EXECUTE PRIVILEGES ON CATEGORY CATE3
*+      HOLDS EXECUTE PRIVILEGES ON ACTIVITY DCMT.N002
*+      HOLDS REFERENCES PRIVILEGES ON ACCESS MODULE THESCHEMA.THEAM
*+      HOLDS EXECUTE PRIVILEGES ON ACCESS MODULE THESCHEMA.THEAM
*+      HOLDS ALL PRIVILEGES ON TABLE THESCHEMA.THE*
*+      HOLDS DISPLAY PRIVILEGES ON ACCESS MODULE THESCHEMA.*
*+      HOLDS DISPLAY PRIVILEGES ON ACCESS MODULE THESCHEMA.FRED*
*+      HOLDS REFERENCES PRIVILEGES ON TABLE THESCHEMA.THETABLE
*+      HOLDS SELECT, INSERT, UPDATE PRIVILEGES ON TABLE THESCHEMA.THETABLE
*+      HOLDS DISPLAY PRIVILEGES ON TABLE THESCHEMA.*
```

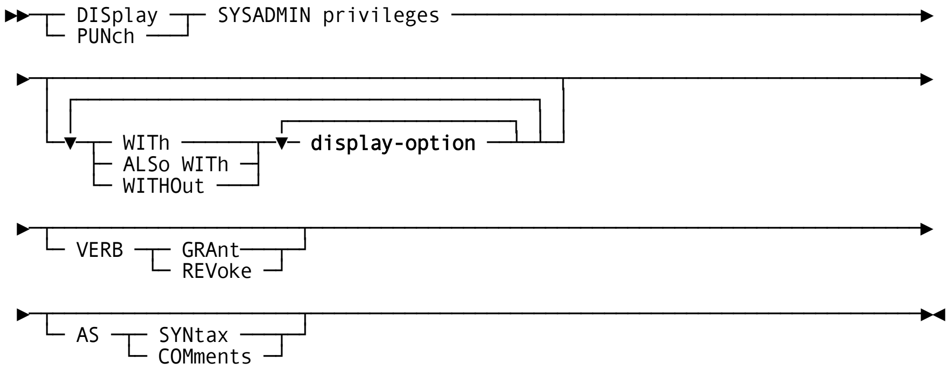
DISPLAY SYSADMIN PRIVILEGES

Displays users who have been granted SYSADMIN privilege for the domain.

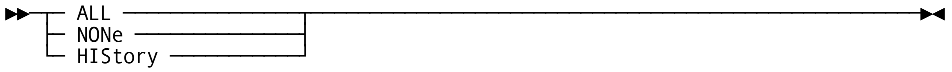
Authorization

To display SYSADMIN privilege, you must hold SYSADMIN privilege.

Syntax



Expansion of Display-Option



Parameters

ALL

Specifies all display options.

NONe

Specifies display of the privilege and the user or group to whom it was granted.

HIStory

Specifies display of the following:

- The date and time the privilege was granted to the user or group, and the ID of the user granted the privilege.
- The date and time the privilege was last updated, and the ID of the user who updated the privilege.

VERB

Specifies the verb used in the display output.

If VERB is not specified, the privilege is displayed in the form of GRANT statements.

GRAnt

Specifies display of SYSADMIN privilege in the form of GRANT statements.

REVoke

Specifies substitution of "REVOKE" for "GRANT" in the display output.

Usage

Specifying WITH VERB REVOKE

By specifying WITH VERB REVOKE and AS SYNTAX, you display output that you can then submit to revoke privilege from those who hold the privilege. This is useful when you need to revoke a privilege from most or all users who hold it.

However, be sure to leave at least one user with SYSADMIN privilege.

DISPLAY PRIVILEGES on a Global Resource

Displays privileges granted on a global resource and the users and groups who hold the privilege.

Authorization

To display privileges on a global resource, you must hold the grantable DISPLAY privilege on the resource or SYSADMIN privilege.

Syntax

```
►►─┬─ DISplay ─┬─ PRIvileges on resource ─┬─ GROup group-identifier ─────┬──────►
   └─ PUNch ───┘                          ├─ USEr user-identifier ──────┤
                                          └─ USEr PROfile profile-name ─┘

 ►─┬─────────────────────────────────────────────────────┬──────────────────────►
   │ ┌──────────────────────────────────────────────┐    │
   │ │                         ┌──────────────────┐ │    │
   └─▼─┬─ WITh ──────────┬──▼─ display-option ─┴─┴────┘
       ├─ ALSo WITh ─────┤
       └─ WITHOut ───────┘

 ►─┬──────────────────────────────┬─────────────────────────────────────────────►
   └─ VERB ─┬─ GRAnt ────────┤
            └─ REVoke ───────┘

 ►─┬──────────────────────────────┬────────────────────────────────────────────►◄
   └─ AS ─┬─ SYNtax ─────────┤
          └─ COMments ───────┘
```

Expansion of Display-Option

```
►►─┬─ ALL ─────────────────┬───────────────────────────────────────────────────►◄
   ├─ NONe ────────────────┤
   └─ HIStory ─────────────┘
```

Parameters

GROup *group-identifier*

Identifies a group.

USEr *user-identifier*

Identifies a user.

USEr PROfile *profile-name*

Identifies a user profile.

ALL

Specifies all display options.

NONe

Specifies display of only the privileges and the authorization identifier to which they have been granted.

HIStory

Specifies display of the following:

- The date and time privilege on the resource was granted to the user or group, and the ID of the user granted the privilege.
- The date and time privilege on the resource was last updated, and the ID of the user who updated the privilege.

VERB

Specifies the verb to be used in the display output.

If VERB is not specified, privileges are displayed in the form of GRANT statements.

GRAnt

Specifies that privileges are displayed in the form of GRANT statements.

REVoke

Specifies substitution of "REVOKE" for "GRANT" in the display output.

Usage

Specifying WITH VERB REVOKE

By specifying WITH VERB REVOKE and AS SYNTAX, you display output that you can then submit to revoke privilege from those who hold the privilege. This is useful when you need to revoke a privilege from most or all users who hold it.

DISPLAY GROUP

Displays information about a group.

Authorization

To display group information, you must hold SYSADMIN privilege.

Syntax

```
►►─┬─ DISplay ─┬─ GROup group-identifier ─────────────────────────────────►
   └─ PUNch ───┘

 ►─┬────────────────────────────────────────────┬─────────────────────────►
   │ ┌────────────────────────────────────────┐ │
   │ │                  ┌─────────────────┐   │ │
   └─▼─┬─ WITh ────────┬─▼─ display-option ─┴─┴─┘
       ├─ ALSo WITh ───┤
       └─ WITHOut ─────┘

 ►─┬──────────────────────┬───────────────────────────────────────────────►◄
   └─ AS ─┬─ SYNtax ───┬──┘
          └─ COMments ─┘
```

Expansion of Display-Option

```
►►─┬─ ALL ──────────────────────────┬─────────────────────────────────────►◄
   ├─ NONe ─────────────────────────┤
   ├─ DETails ──────────────────────┤
   ├─ HIStory ──────────────────────┤
   ├─ USErs ────────────────────────┤
   ├─ all PRIvileges ───────────────┤
   ├─ ACCess MODule privileges ─────┤
   ├─ ACTivity privileges ──────────┤
   ├─ AREa privileges ──────────────┤
   ├─ CATegory privileges ──────────┤
   ├─ DB privileges ────────────────┤
   ├─ DBADMin privileges ───────────┤
   ├─ DBTable privileges ───────────┤
   ├─ DCADMin privileges ───────────┤
   ├─ DMCl privileges ──────────────┤
   ├─ GROup privileges ─────────────┤
   ├─ NONSQL SCHema privileges ─────┤
   ├─ SCHema privileges ────────────┤
   ├─ SYSADMin privileges ──────────┤
   ├─ SYStem privileges ────────────┤
   ├─ SYStem PROfile privileges ────┤
   ├─ TABle privileges ─────────────┤
   ├─ USEr PRIvileges ──────────────┤
   └─ USEr PROfile privileges ──────┘
```

Parameters

ALL

Specifies all display options.

NONe

Specifies display of the group identifier and its status (active or logically deleted).

DETails

Specifies display of the status of the group and a description of the group, if one is included in the definition.

HIStory

Specifies display of the following:

- The date and time the group definition was created, and the ID of the user who created the definition.
- The date and time the group definition was last updated, and the ID of the user who updated the definition.

USErs

Specifies display of all users who are members of the group.

all PRIvileges

Specifies display of all privileges held by the group.

ACCess MODule privileges

Specifies display of privileges on access modules, if any are held by the group.

ACTivity privileges

Specifies display of privileges on activities, if any are held by the group.

AREa privileges

Specifies display of privileges on areas, if any are held by the group.

CATegory privileges

Specifies display of privileges on categories, if any are held by the group.

DB privileges

Specifies display of privileges on databases, if any are held by the group.

DBADMin privileges

Specifies display of DBADMIN privilege, if it is held by the group.

DBTable privileges

Specifies display of privileges on database name tables, if any are held by the group.

DCADMin privileges

Specifies display of DCADMIN privilege, if it is held by the group.

DMCl privileges

Specifies display of privileges on DMCLs, if any are held by the group.

GROup privileges

Specifies display of privileges on groups, if any are held by the group.

NONSQL SCHema privileges

Specifies display of privileges on non-SQL-defined schemas, if any are held by the group.

SCHema privileges

Specifies display of privileges on SQL-defined schemas, if any are held by the group.

SYSADMin privileges

Specifies display of SYSADMIN privilege, if it is held by the group.

SYStem privileges

Specifies display of privileges on systems, if any are held by the group.

SYStem PROfile privileges

Specifies display of privileges on system profiles, if any are held by the group.

TABle privileges

Specifies display of privileges on tables, if any are held by the group.

USEr PRIvileges

Specifies display of privileges on users, if any are held by the group.

Note that at least the first three characters of the keyword PRIVILEGES must be specified to display users on which the group holds privileges. This removes any potential ambiguity with specification of the preceding USERS display option.

USEr PROfile privileges

Specifies display of privileges on user profiles, if any are held by the group.

Usage

Privileges Assigned in More Than One Dictionary

It may be necessary to connect to more than one dictionary to see all privileges assigned to a group. For example, if you issue the DISPLAY statement while connected to the system dictionary, the output will not include the SQL table privileges assigned in an application dictionary.

DISPLAY USER

Displays information about a user.

Authorization

To display privileges granted to a user you must hold SYSADMIN privilege.

Syntax

```
►►─┬─ DISplay ─┬─ USEr user-identifier ───────────────────────────────────►
   └─ PUNch ───┘

►──┬───────────────────────────────────────────────────────┬───────────────►
   │ ┌─────────────────────────────────────────────────┐   │
   │ │                 ┌───────────────────────────┐   │   │
   └─▼──┬─ WITh ───────┬─▼─ display-option ───────────┴───┴───┘
        ├─ ALSo WITh ──┤
        └─ WITHOut ────┘

►──┬─────────────────────────┬───────────────────────────────────────────►◄
   └─ AS ─┬─ SYNtax ───┬─────┘
          └─ COMments ─┘
```

Expansion of Display-Option

```
►►─┬─ ALL ──────────────────────────────┬───────────────────────────────►◄
   ├─ NONe ─────────────────────────────┤
   ├─ DETails ──────────────────────────┤
   ├─ HIStory ──────────────────────────┤
   ├─ GROups ───────────────────────────┤
   ├─ all PRIvileges ───────────────────┤
   ├─ ACCess MODule privileges ─────────┤
   ├─ ACTivity privileges ──────────────┤
   ├─ AREa privileges ──────────────────┤
   ├─ CATegory privileges ──────────────┤
   ├─ DB privileges ────────────────────┤
   ├─ DBADMin privileges ───────────────┤
   ├─ DBTable privileges ───────────────┤
   ├─ DCADMin privileges ───────────────┤
   ├─ DMCl privileges ──────────────────┤
   ├─ GROup PRIvileges ─────────────────┤
   ├─ NONSQL SCHema privileges ─────────┤
   ├─ SCHema privileges ────────────────┤
   ├─ SYSADMin privileges ──────────────┤
   ├─ SYStem privileges ────────────────┤
   ├─ SYStem PROfile privileges ────────┤
   ├─ TABle privileges ─────────────────┤
   ├─ USEr privileges ──────────────────┤
   └─ USEr PROfile privileges ──────────┘
```

Parameters

ALL

Specifies all display options.

NONe

Specifies display of only the user identifier and its status (active or logically deleted).

DETails

Specifies display of the following information about the user:

- The status of the user identifier.
- The user's name.
- Whether a password has been assigned to the user.
- The user profile, if any, associated with the user.
- The description, if any, of the user.

HIStory

Specifies display of the following:

- The date and time the user definition was created, and the ID of the user who created the definition.
- The date and time the user definition was last updated and the ID of the user who updated the definition.

GROups

Specifies display of all groups of which the user is a member.

all PRIvileges

Specifies display of all privileges held by the user.

ACCess MODule privileges

Specifies display of privileges on access modules, if any are held by the user.

ACTivity privileges

Specifies display of privileges on activities, if any are held by the user.

AREa privileges

Specifies display of privileges on areas, if any are held by the user.

CATegory privileges

Specifies display of privileges on categories, if any are held by the user.

DB privileges

Specifies display of privileges on databases, if any are held by the user.

DBADMin privileges

Specifies display of DBADMIN privilege, if it is held by the user.

DBTable privileges

Specifies display of privileges on database name tables, if any are held by the user.

DCADMin privileges

Specifies display of DCADMIN privilege, if it is held by the user.

DMCl privileges

Specifies display of privileges on DMCLs, if any are held by the user.

GROup PRIvileges

Specifies display of privileges on groups, if any are held by the user.

Note that at least the first three characters of the keyword PRIVILEGES must be specified to display groups on which the user holds privileges. This removes any potential ambiguity with specification of the preceding GROUPS display option.

NONSQL SCHema privileges

Specifies display of privileges on non-SQL-defined schemas, if any are held by the user.

SCHema privileges

Specifies display of privileges on SQL-defined schemas, if any are held by the user.

SYSADMin privileges

Specifies display of SYSADMIN privilege, if it is held by the user.

SYStem privileges

Specifies display of privileges on systems (including SIGNON), if any are held by the user.

SYStem PROfile privileges

Specifies display of privileges on system profiles, if any are held by the user.

TABle privileges

Specifies display of privileges on tables, if any are held by the user.

USEr privileges

Specifies display of privileges on users, if any are held by the user.

USEr PROfile privileges

Specifies display of privileges on user profiles, if any are held by the user.

Usage

Privileges Assigned in More Than One Dictionary

It may be necessary to connect to more than one dictionary to see all privileges assigned to a user. For example, if you issue the DISPLAY statement while connected to the system dictionary, the output will not include the SQL table privileges assigned in an application dictionary.

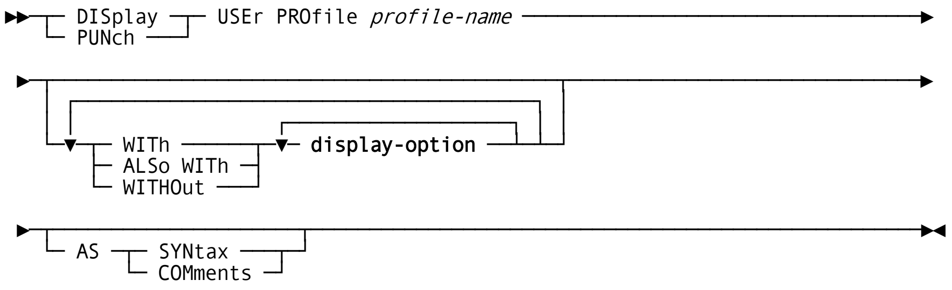
DISPLAY USER PROFILE

Displays information about a user profile.

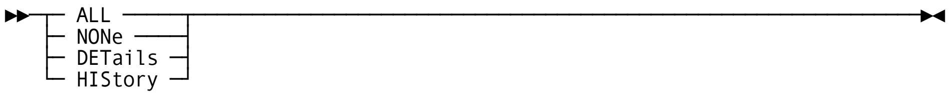
Authorization

To display user profile information, you must hold SYSADMIN privilege.

Syntax



Expansion of Display-Option



Parameters

ALL

Specifies all display options.

NONe

Specifies display of the profile name.

DETails

Specifies display of all attributes defined in the profile.

HIStory

Specifies display of the following:

- The date and time the user profile definition was created, and the ID of the user who created the definition.
- The date and time the user profile definition was last updated, and the ID of the user who updated the definition.

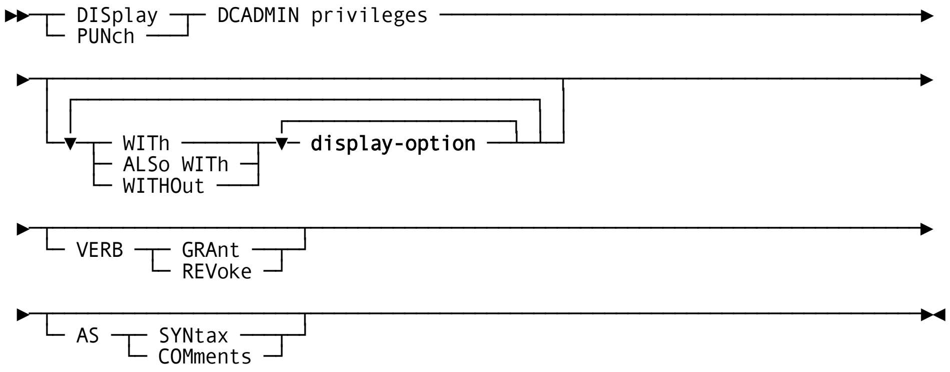
DISPLAY DCADMIN PRIVILEGES

Displays users and groups who hold DCADMIN privilege for the system.

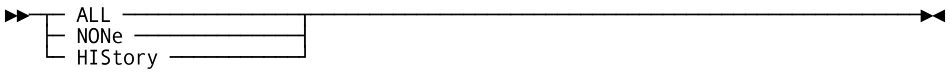
Authorization

To display DCADMIN privilege, you must hold DCADMIN privilege or SYSADMIN privilege.

Syntax



Expansion of Display-Option



Parameters

ALL

Specifies all display options.

NONe

Specifies display of the users or groups to whom the privilege was granted.

HIStory

Specifies display of the following:

- The date and time the privilege was granted to the user or group, and the ID of the user granted the privilege.
- The date and time the privilege was last updated, and the ID of the user who updated the privilege.

VERB

Specifies the verb used in the display output.

If VERB is not specified, the privilege is displayed in the form of GRANT statements.

GRAnt

Specifies that the privilege is displayed in the form of GRANT statements.

REVoke

Specifies substitution of "REVOKE" for "GRANT" in the display output.

Usage

Specifying WITH VERB REVOKE

By specifying WITH VERB REVOKE and AS SYNTAX, you display output that you can then submit to revoke privilege from those who hold the privilege. This is useful when you need to revoke a privilege from most or all users who hold it.

DISPLAY PRIVILEGES on a System Resource

Displays privileges granted on a system resource and the users and groups who hold the privileges.

Authorization

To display privileges on a system resource, you must hold one of these privileges:

- DCADMIN privilege
- SYSADMIN privilege

You can display privileges on a system or system profile if you hold DISPLAY privilege WITH GRANT OPTION on the resource.

Syntax

```
►►─┬─ DISplay ─┬─ PRIvileges on resource ──────────────────────────────────►
   └─ PUNch ───┘

►─┬─ ACTivity application-name.activity-name ─┬────────────────────────────►
  ├─ CATegory category-name ──────────────────┤
  ├─ SYStem system-identifier ────────────────┤
  └─ SYStem PROfile profile-name ─────────────┘

►─┬────────────────────────────────────────────┬───────────────────────────►
  │  ┌──────────────────────────────────────┐  │
  └─▼─┬─ WITh ──────┬─▼─ display-option ─┴─┴─┘
      ├─ ALSo WITh ─┤  └────────────────────┘
      └─ WITHOut ───┘

►─┬──────────────────────┬──────────────────────────────────────────────────►
  └─ VERB ─┬─ GRAnt ──┬──┘
           └─ REVoke ─┘

►─┬────────────────────────┬────────────────────────────────────────────────►◄
  └─ AS ─┬─ SYNtax ─────┬──┘
         └─ COMments ───┘
```

Expansion of Display-Option

```
►►─┬─ ALL ──────────┬─────────────────────────────────────────────────────►◄
   ├─ NONe ─────────┤
   └─ HIStory ──────┘
```

Parameters

ACTivity

Specifies that the type of resource is activity.

application-name.activity-name

Identifies the application in which the activity is defined and the name of the activity.

CATegory

Specifies that the type of resource is category.

category-name

Identifies the category.

SYStem

Specifies that the type of resource is system.

system-identifier

Identifies the system.

SYStem PROfile

Specifies that the type of resource is system profile.

profile-name

Identifies the system profile.

ALL

Specifies all display options.

NONe

Specifies display of only the privileges and the authorization identifier to which they have been granted.

HIStory

Specifies display of the following:

- The date and time privilege on the resource was granted to the user or group, and the ID of the user who granted the privilege.
- The date and time privilege on the resource was last updated, and the ID of the user who updated the privilege.

VERB

Specifies the verb to be used in the display output.

If VERB is not specified, the privilege is displayed in the form of GRANT statements.

GRAnt

Specifies that privileges are to be displayed in the form of GRANT statements.

REVoke

Specifies substitution of "REVOKE" for "GRANT" in the display output.

Usage

Specifying WITH VERB REVOKE

By specifying WITH VERB REVOKE and AS SYNTAX, you display output that you can then submit to revoke privilege from those who hold the privilege. This is useful when you need to revoke a privilege from most or all users who hold it.

DISPLAY RESOURCE (System)

Displays information about a system resource.

Authorization

To display a system resource, you must hold one of these privileges:

- DISPLAY privilege WITH GRANT OPTION on the resource
- DCADMIN privilege
- SYSADMIN privilege

Syntax

```
►►─┬─ DISplay ─┬─ RESource ──────────────────────────────────────────►
   └─ PUNch ───┘

 ►─┬─ ACTivity application-name.activity-name ─┬─────────────────────►
   ├─ CATegory category-name ──────────────────┤
   └─ SYStem system-identifier ────────────────┘

 ►─┬─────────────────────────────────────────────┬───────────────────►
   │ ┌────────────────────────────────────────┐  │
   │ │                  ┌─────────────────┐   │  │
   └─▼─┬─ WITh ───────┬─▼─ display-option ─┴───┴──┘
       ├─ ALSo WITh ──┤
       └─ WITHOut ────┘

 ►─┬─────────────────────┬──────────────────────────────────────────►◄
   └─ AS ─┬─ SYNtax ───┤
          └─ COMments ─┘
```

Expansion of Display-Option

```
►►─┬─ ALL ─────┬─────────────────────────────────────────────────────►◄
   ├─ NONe ────┤
   ├─ DETails ─┤
   └─ HIStory ─┘
```

Note: The DETAILS option is meaningful only for the CATEGORY resource type.

Parameters

ACTivity

Specifies that the type of resource is activity.

application-name.activity-name

Identifies the application in which the activity is defined and the activity name.

CATegory

Specifies that the type of resource is category.

category-name

Identifies the category.

SYStem

Specifies that the type of resource is system.

system-identifier

Identifies the system.

SYStem PROfile

Specifies that the type of resource is system profile.

profile-name

Identifies the system profile.

ALL

Specifies all display options.

NONe

Specifies display of only the resource type such as:

- For category, the resource name
- For activity, the activity name and number
- For system, the system identifier

DETails

For a category, specifies display of the resources in the category.

For a system profile, specifies display of the attributes defined in the profile.

HIStory

Specifies display of the following:

- The date and time the resource was created, and the ID of the user who created it.
- The date and time the resource was last altered, and the ID of the user who altered it.

DISPLAY PRIVILEGES on a Database Resource

Displays privileges granted on a database resource and the users and groups who hold the privileges.

Authorization

To display privileges on a database resource, you must hold DBADMIN privilege on the current dictionary, SYSADMIN privilege, or the privilege listed in this table:

Resource type	Privilege to display privileges
Access Module	DISPLAY (WITH GRANT OPTION) on the access module
Area	DISPLAY (WITH GRANT OPTION) on the segment containing the area
Database	DISPLAY (WITH GRANT OPTION) on the database identified as a DB resource
Database name table	DISPLAY (WITH GRANT OPTION) on the database name table
DMCL	DISPLAY (WITH GRANT OPTION) on the DMCL
Non-SQL-defined schema	DISPLAY (WITH GRANT OPTION) on the non-SQL-defined schema
SQL-defined schema	DISPLAY (WITH GRANT OPTION) on the schema
Table	DISPLAY (WITH GRANT OPTION) on the table

Syntax

```
►►─┬─ DISplay ─┬─ PRIvileges on resource ──────────────────────────►
   └─ PUNch ───┘

►─┬─ ACCess MODule schema-name.access-module-name ─┬────────────────►
  ├─ AREa segment-name.area-name ──────────────────┤
  ├─ DB database-name ─────────────────────────────┤
  ├─ DBTable dbtable-name ─────────────────────────┤
  ├─ DMCl dmcl-name ───────────────────────────────┤
  ├─ NONSQL SCHema Vnnnn.nonsql-schema-name ───────┤
  ├─ SCHema schema-name ───────────────────────────┤
  └─ TABle schema-name.table-name ─────────────────┘

►─┬──────────────────────────────────────────────────┬──────────────►
  │ ┌────────────────────────────────────────────┐   │
  │ │                   ┌──────────────────────┐ │   │
  └─▼─┬─ WITh ──────┬───▼─ display-option ─────┴─┴───┘
      ├─ ALSo WITh ─┤
      └─ WITHOut ───┘

►─┬──────────────────────────┬──────────────────────────────────────►
  └─ VERB ─┬─ GRAnt ──┬──────┘
           └─ REVoke ─┘

►─┬──────────────────────────┬──────────────────────────────────────►◄
  └─ AS ─┬─ SYNtax ───┬──────┘
         └─ COMments ─┘
```

Expansion of Display-Option

```
►►─┬─ ALL ─────────────┬──────────────────────────────────────────────►◄
   ├─ NONe ────────────┤
   └─ HIStory ─────────┘
```

Parameters

ACCess MODule

Specifies that the type of resource is access module.

schema-name.access-module-name

Identifies the access module and its schema-name qualifier.

AREa

Specifies that the type of resource is area.

segment-name.area-name

Identifies the segment containing the area and the area name.

DB

Specifies that the type of resource is database.

database-name

Identifies a segment or a name in the database name table.

DBTable

Specifies that the type of resource is database name table.

dbtable-name

Identifies the database name table.

DMCL

Specifies that the type of resource is DMCL.

dmcl-name

Identifies the DMCL.

NONSQL SCHema

Specifies that the type of resource is non-SQL-defined schema.

V*nnnn.nonsql-schema-name*

Identifies the non-SQL-defined schema and its version (*nnnn*).

SCHema

Specifies that the type of resource is an SQL-defined schema.

schema-name

Identifies the schema.

TABle

Specifies that the type of resource is table.

schema-name.table-name

Identifies the table and its schema-name qualifier.

ALL

Specifies all display options.

NONe

Specifies display of only the privileges and the authorization identifier to which they have been granted.

HIStory

Specifies display of the following:

- The date and time privilege on the resource was granted to the user or group, and the ID of the user who granted the privilege.
- The date and time privilege on the resource was last updated, and the ID of the user who updated the privilege.

VERB

Specifies the verb to be used in the display output.

If VERB is not specified, the privilege is displayed in the form of GRANT statements.

GRAnt

Specifies that privileges are displayed in the form of GRANT statements.

REVoke

Specifies substitution of "REVOKE" for "GRANT" in the display output.

Usage

Specifying WITH VERB REVOKE

By specifying WITH VERB REVOKE and AS SYNTAX, you display output that you can then submit to revoke privilege from those who hold the privilege. This is useful when you need to revoke a privilege from most or all users who hold it.

DISPLAY RESOURCE (Database)

Displays information about a database resource.

Authorization

To display a database resource, you must hold DBADMIN privilege on the current dictionary, SYSADMIN privilege, or the privilege listed in this table:

Resource type	Privilege to display privileges
Access Module	DISPLAY on the access module
Area	DBADMIN on the segment containing the area
Database	DBADMIN on the database identified as a DB resource
Database name table	DISPLAY on the database name table
DMCL	DISPLAY on the DMCL
Non-SQL-defined schema	DISPLAY on the non-SQL-defined schema
SQL-defined schema	DISPLAY on the schema
Table	DISPLAY on the table

Syntax

```
►►─┬─ DISplay ─┬─ RESource ─────────────────────────────────────────────────────►
   └─ PUNch ───┘

►─┬─ ACCess MODule schema-name.access-module-name ─┬─────────────────────────────►
  ├─ AREa segment-name.area-name ──────────────────┤
  ├─ DB database-name ─────────────────────────────┤
  ├─ DBTable dbtable-name ─────────────────────────┤
  ├─ DMCl dmcl-name ───────────────────────────────┤
  ├─ NONSQL SCHema Vnnnn.nonsql-schema-name ───────┤
  ├─ SCHema schema-name ───────────────────────────┤
  └─ TABle schema-name.table-name ─────────────────┘

►─┬──────────────────────────────────────────────┬───────────────────────────────►
  │   ┌──────────────────────────────────────┐   │
  └───▼─┬─ WITh ──────┬──▼─┬─ ALL ─────┬──────┴──┘
        ├─ ALSo WITh ─┤    ├─ NONe ────┤
        └─ WITHOut ───┘    └─ HIStory ─┘

►─┬────────────────────────────┬────────────────────────────────────────────────►◄
  └─ AS ─┬─ SYNtax ─────┬──────┘
         └─ COMments ───┘
```

Parameters

ACCess MODule

Specifies that the type of resource is access module.

schema-name.access-module-name

Identifies the access module and its schema-name qualifier.

AREa

Specifies that the type of resource is area.

segment-name.area-name

Identifies the segment containing the area and the area name.

DB

Specifies that the type of resource is database.

database-name

Identifies a segment or a name in the database name table.

DBTable

Specifies that the type of resource is database name table.

dbtable-name

Identifies the database name table.

DMCL

Specifies that the type of resource is DMCL.

dmcl-name

Identifies the DMCL.

NONSQL SCHema

Specifies that the type of resource is non-SQL-defined schema.

V*nnnn.nonsql-schema-name*

Identifies the non-SQL-defined schema and its version (*nnnn*).

SCHema

Specifies that the type of resource is schema.

schema-name

Identifies the schema.

TABle

Specifies that the type of resource is table.

schema-name.table-name

Identifies the table and its schema-name qualifier.

ALL

Specifies all display options.

NONe

Specifies display of the resource type and name only.

HIStory

Specifies display of the following:

- The date and time the resource was created, and the ID of the user who created it.
- The date and time the resource was last updated, and the ID of the user who updated it.
- For a table or schema, the authorization identifier of the owner.

Chapter 17: DISPLAY/PUNCH ALL Syntax for Security Definitions

This section contains the following topics:

Using DISPLAY/PUNCH ALL Syntax

In addition to using DISPLAY and PUNCH syntax for specific resource definitions in a CA IDMS security database, you can issue either a DISPLAY ALL or PUNCH ALL statement for an entity type to display or punch all occurrences defined within that entity type. For example, you can issue a DISPLAY ALL RESOURCE AREAS to see the security definitions for all areas secured in a security database. You can also select occurrences of an entity type to display or punch by specifying the following:

- Conditional expressions
- First occurrence of the entity type
- Last occurrence of the entity type
- A specific number of occurrences

Choosing Which Entity Occurrences to Display

The DISPLAY and PUNCH ALL syntax supports a variety of selection criteria to select occurrences to DISPLAY or PUNCH. You can use a conditional expression with Boolean criteria to select occurrences, including a mask comparison. The mask comparison supports the use of different keywords for each entity type. A table of keywords by entity type is presented in Usage (see page 371).

Issue Statements from CA IDMS Command Facility

You can issue DISPLAY/PUNCH statements from either the Online (OCF) or Batch (BCF) Command Facility.

Syntax

```
►►─┬─ DISplay ─┬──┬─ ALL ─────────────────────────────┬─ entity-type ──────►
   └─ PUNch ───┘  └─┬─ FIRst ─┬──┬──────────────────┬─┘
                    └─ LASt ──┘  ├─ 1 ◄─────────────┤
                                 └─ entity-count ───┘

►─┬──────────────────────────────────┬──────────────────────────────────►
  └─ WHEre conditional-expression ──┘

►─┬──────────────────────────┬──────────────────────────────────────────►
  └─ VERB ─┬─ DISplay ◄─┬────┘
           ├─ PUNch ────┤
           ├─ CREate ───┤
           ├─ ALTer ────┤
           └─ DROp ─────┘

►─┬─────────────────────────┬───────────────────────────────────────────►
  └─ AS ─┬─ COMments ─┬─────┘
         └─ SYNtax ───┘
```

Expansion of Conditional-Expression

```
►►─┬─ mask-comparison ──────────────────────────────────┬──────────────►
   ├─ value-comparison ─────────────────────────────────┤
   └─┬───────┬── ( ─┬─ mask-comparison ──┬─ ) ──────────┘
     └─ NOT ─┘      └─ value-comparison ─┘

►─┬───────────────────────────────────────────────────────────────┬───►◄
  │ ┌───────────────────────────────────────────────────────────┐ │
  └─▼─┬─ AND ─┬──┬─ mask-comparison ────────────────────────────┤─┘
      └─ OR ──┘  ├─ value-comparison ───────────────────────────┤
                 └─┬───────┬── ( ─┬─ mask-comparison ──┬─ ) ────┘
                   └─ NOT ─┘      └─ value-comparison ─┘
```

Expansion of Mask-Comparison

```
►►── entity-option-keyword ─────────────────────────────────────────────►

►─┬─ CONTAINs ─┬─ 'mask-value' ────────────────────────────────────────►◄
  └─ MATCHES ──┘
```

Expansion of Value-Comparison

```
►►─┬─ 'character-string-literal' ─┬─────────────────────────────────────►
   ├─ numeric-literal ────────────┤
   └─ entity-option-keyword ──────┘

►─┬─ IS ─┬───────┬───────┬─┬─ 'character-string-literal' ─┬────────────►◄
  │      └─ NOT ─┘       │ ├─ numeric-literal ────────────┤
  ├─ NE ─────────────────┤ └─ entity-option-keyword ──────┘
  └─┬───────┬─┬─┬─ EQ ─┬─┤
    └─ NOT ─┘ │ └─ = ──┘ │
              ├─┬─ GT ─┬─┤
              │ └─ > ──┘ │
              ├─┬─ LT ─┬─┤
              │ └─ < ──┘ │
              ├─ GE ─────┤
              └─ LE ─────┘
```

Parameters

ALL

Lists all occurrences of the requested entity type that the current user is authorized to display.

Online users: With a large number of entity occurrences, ALL may have a slow response time.

FIRst

Lists the first occurrence of the named entity type.

LASt

Lists the last occurrence of the named entity type.

entity-count

Specifies the number of occurrences of the named entity type to list. 1 is the default.

entity-type

Identifies the entity type that is the object of the DISPLAY/PUNCH ALL request. Valid values appear in the table in Usage (see page 371).

VERB DISplay/PUNch/CREate/ALTer/DROp

Specifies the verb that is to accompany DISPLAY/PUNCH output. DISPLAY is the default.

AS SYNtax

Specifies that the text output by the DISPLAY/PUNCH verb is to appear as syntax. In an online session, text displayed as syntax can be edited and resubmitted to the command facility. If the PUNCH command is issued in batch mode, the batch command facility directs the output to the SYSPCH file, where it can be edited and subsequently resubmitted.

AS COMments

Specifies that the text output by the DISPLAY/PUNCH verb be formatted as comments; comments are preceded by *+ and are ignored by the command facility.

WHEre conditional-expression

Specifies criteria to be used in selecting occurrences of the requested entity type.

The outcome of a test for the condition determines which occurrences of the named entity type are displayed.

mask-comparison

Compares an entity type operand with a mask value.

entity-option-keyword

Identifies the left operand as a syntax option associated with the named entity type. The table in Usage (see page 371) lists valid options for each entity type.

CONTAINs

Searches the left operand for an occurrence of the right operand. The length of the right operand must be less than or equal to the length of the left operand. If the right operand is not contained entirely in the left operand, the outcome of the condition is false.

MATCHES

Compares the left operand with the right operand one character at a time, beginning with the leftmost character in each operand. When a character in the left operand does not match a character in the right operand, the outcome of the condition is false.

'*mask-value*'

Identifies the right operand as a character string; the specified value must be enclosed in quotation marks. *Mask-value* can contain the following special characters:

@	Matches any alphabetic character in *entity-option-keyword*.
#	Matches any numeric character in *entity-option-keyword*.
*	Matches any character in *entity-option-keyword*.

value-comparison

Compares values contained in the left and right operands based on the specified comparison operator.

'*character-string-literal*'

Identifies a character string enclosed in quotes.

numeric-literal

Identifies a numeric value.

entity-option-keyword

Identifies a syntax option associated with the named entity type; valid options for each entity type are listed in the table presented in Usage (see page 371).

IS

Specifies that the left operand must equal the right operand for the condition to be true.

NE

Specifies that the left operand must *not* equal the right operand for the condition to be true.

EQ/=

Specifies that the left operand must equal the right operand for the condition to be true.

GT/>

Specifies that the left operand must be greater than the right operand for the condition to be true.

LT/<

Specifies that the left operand must be less than the right operand for the condition to be true.

GE

Specifies that the left operand must be greater than or equal to the right operand for the condition to be true.

LE

Specifies that the left operand must be less than or equal to the right operand for the condition to be true.

NOT

Specifies that the opposite of the condition fulfills the test requirements. If NOT is specified, the condition must be enclosed in parentheses.

AND

Indicates the expression is true only if the outcome of both test conditions is true.

OR

Indicates the expression is true if the outcome of either one or both test conditions is true.

Usage

Output Contains Only Enough Information to Display/Punch Entity

Output produced by DISPLAY or PUNCH ALL consists only of the information necessary to execute a DISPLAY/PUNCH request for each entity occurrence. For example, Resource DMCL occurrences are displayed with their name, and AREA occurrences with their fully qualified name (that is, segmentname.areaname). In an online session, the user can execute the displayed statements by pressing [Enter]. This two-step process allows the user to scan the names of entity occurrences related to the database in which the statement is issued.

Valid Entity Types and Option Keywords for Conditional Expressions

The following table lists valid entity types and keywords that you can specify as *entity-type* and *entity-option-keyword* in the DISPLAY ALL and PUNCH ALL syntax.

Entity type	Entity-option keyword	Selects based on
All Security components		
	NAMe	Unqualified Name (1)
	FULl NAMe	Qualified Name (1)
	RESource NAMe	Unqualified Name (Resources only) (1)
	CREated by	User who created occurrence
	PREpared by	
	LASt UPDated by	User who created occurrence
	REVised by	User who last updated occurrence
	DATe last UPDated	User who last updated occurrence
	MONth last UPDated	Date (MM/DD/YY) occurrence last updated
	DAY last UPDated	
	YEAr last UPDated	Month occurrence last updated
	DATe CREated	
		Day occurrence last updated
	MONth CREated	Year occurrence last updated
	DAY CREated	Date (MM/DD/YY) occurrences created
	YEAr CREated	
		Month occurrence created
		Day occurrence created
		Year occurrence created
		.tabreak
Global Security components		
GROups	GROup name	Name (ID) of Group
	STAtus	Status of GROUP (ACTIVE, INACTIVE, LOGICALLY DELETED)

Entity type	Entity-option keyword	Selects based on
USErs	USEr name	Name (ID) of User
	STAtus	Status of USER (ACTIVE, INACTIVE, LOGICALLY DELETED)
	FULl NAMe	Full Name of User
	PROfile	Profile assigned to User
USEr PROfiles	USEr PROfile name	Profile Name
	PROfile name	Profile Name
Physical Database Security components		
RESource AREas	resource AREa NAMe	Unqualified AREA name (1)
	SEGment name	Area's segment name
RESource DBs	resource DB NAMe	Name of Database
RESource DBTables	resource DBTable NAMe	Name of DBTable
RESource DMCls	resource DMCL NAMe	Name of DMCL
RESource NONsql SCHEmas	resource NONSQL SCHEma NAME	Name of NON SQL Schema
SQL Security Components		
RESource ACCess MODules	resource ACCess MODule NAMe	Unqualified Name of Access Module (1)
or	resource AM NAMe	Unqualified Name of Access Module (1)
RESource AMS		
	AM NAMe	Unqualified Name of Access Module (1)
	SCHema name	Schema Name of Access Module
RESource SCHemas	resource SCHema NAMe	Name of SQL Schema
RESource TABles	resource TABle NAMe	Unqualified Name of Table (1)
	SCHema NAMe	Schema Name of Table
System Security Components		
RESource ACTivities	resource ACTivity name	Name of Activity
	NUMber	Activity Number
RESource CATegories	resource CATegory NAMe	Name of Category
	NUMber	Category Number

Entity type	Entity-option keyword	Selects based on
RESource SYStems	resource SYStem NAMe	Name of System
SYStem PROfiles	system PROfile NAMe	Profile Name
The following **Resource Category Components** can be selected using the specified entity-option keyword (in addition to those specified in the preceding Resource Categories).		
RESource CATegory ACCess MODules or RESource CATegory AMS	ACCess MODule name	Unqualified Access Module Name (1)
	DICTName	Dictionary Name
	DICtionary name	Dictionary Name
	SCHema name	SQL Schema Name
RESource category LOAd MODules	LOAd MODule name	Unqualified Load Module Name (1)
	DICTName	Dictionary Name
	DICtionary name	Dictionary Name
	Version	Version Number (in V*nnnn* format)
RESource category PROgrams	PROgram name	Unqualified Program name
	FILe name	File Name (CDMSLIB)
	Version	Version Number (in V*nnnn* format)
RESource category QUEues	QUEue name	Name of Queue
RESource category RUNunits	RUNunit name	Unqualified Rununit name (1)
	DATabase NAMe	Database Name
	DBName	Database Name
	SUBschema name	Subschema Name
	PROgram name	Program Name
RESource category TASks	TASk name	Name of task

(1) Unqualified name selections are based on the primary name of the entity occurrence only. To select based on the fully qualified occurrence name, token FULL NAME must be specified. Security components with qualified names are specified in the following table.

Fully Qualified Names of Security Components

The fully qualified names of security components are listed in the following table.

Resource	Fully qualified name
ACCESS MODULE	*schema-name.access-module-name*
AREA	*segment-name.area-name*
TABLE	*schema-name.table-name*
CATEGORY ACCESS MODULE	*dictname.schema-name. access-module-name*
CATEGORY LOAD MODULE	*dictname.Vnnnn.load-module-name*
CATEGORY RUNUNIT	*dbname.subschema-name.program-name*
CATEGORY PROGRAM	CDMSLIB.*program-name* or *Vnnnn.program-name*

For all other security components, unqualified and qualified names are the same.

Date and Year 2000 support

You can use date selection criteria and year 2000 support in DISPLAY/PUNCH ALL statements to display security entities.

You implement date selection criteria in these WHERE clause options:

- DATE CREATED
- DATE LAST UPDATED

You can specify the date as a *value-comparison* string in the form 'MM/DD/YY' in the right side of the conditional expression. CA IDMS extracts it in CCYYMMDD form to accurately determine the relationship of dates. For example, this DISPLAY ALL statement:

```
DISPLAY ALL USERS WHERE DATE CREATED > '01/01/96';
```

establishes a search criteria to identify the USERS whose DATE CREATED values are greater than the specified string. The DISPLAY ALL process determines that the date '01/01/96' is greater than the date '12/31/95'.

Alternatively, you may specify the *value-comparison* string on either side of the conditional expression in the form 'CCYYMMDD' to achieve the same results.

You can also substitute day, month, or year for each of these WHERE clause options. For example, this DISPLAY ALL statement specifies a search condition that is based on month and year:

```
DISPLAY ALL RESOURCE AREAS
  WHERE MONTH CREATED = '01'
  AND YEAR CREATED > '95';
```

Default Order of Precedence Applied to Logical Operators

Conditional expressions can contain a single condition, or two or more conditions combined with the logical operators AND or OR. The logical operator NOT specifies the opposite of the condition. The command facility evaluates operators in a conditional expression one at a time, from left to right, in order of precedence. The default order of precedence is as follows:

- MATCHES or CONTAINS keywords
- EQ, NE, GT, LT, GE, LE operators
- NOT
- AND
- OR

If parentheses are used to override the default order of precedence, the command facility evaluates the expression within the innermost parentheses first.

The following examples show sample DISPLAY statements for security definitions.

DISPLAY ALL GROUPS WHERE STATUS IS 'ACTIVE'

```
        OCF rr.r IDMS  PAGE 1 LINE 1  DICT=SYSTEM      1/8 cv-name
DISPLAY ALL GROUPS WHERE STATUS IS 'ACTIVE';
*+ Status = 0       SQLSTATE = 00000
*+   DISPLAY GROUP "TESTGROUP" ;
*+   DISPLAY GROUP "PUBLIC" ;
*+   DISPLAY GROUP "MIS" ;
*+   DISPLAY GROUP "HR" ;
*+   DISPLAY GROUP "ACCOUNTING" ;
*+ I DC601157  NO MORE ENTITY OCCURRENCES FOUND                    WORD  1
```

DISPLAY ALL USERS WHERE USER NAME MATCHES 'SP'

```
        OCF rr.r IDMS  PAGE 1 LINE 1  DICT=SYSTEM      1/5 cv-name
DISPLAY ALL USERS WHERE USER NAME MATCHES 'SP'
*+ Status = 0       SQLSTATE = 00000
*+   DISPLAY USER "SPILL01" ;
*+   DISPLAY USER "SPANL01" ;
*+ I DC601157  NO MORE ENTITY OCCURRENCES FOUND                    WORD  1
```

Appendix A: Security Macro JCL

This section contains the following topics:

z/OS JCL

Using SMP/E

Any modifications to CA IDMS load libraries in z/OS should be applied by SMP/E.

Note: For instructions on how to assemble and link edit a module using SMP/E, see the *CA IDMS Installation and Maintenance Guide for z/OS*.

#CTABGEN

The #CTABGEN macro can be assembled using the following JCL.

```
//          EXEC HLASMCL
//ASM.SYSLIB   DD   DSN=yourHLQ.CAGJMAC,DISP=SHR
//          DD   DSN=zOS.maclib,DISP=SHR
//ASM.SYSIN    DD   *
       #CTABGEN macro
       END
/*
//LKED.SYSLMOD DD   DSN=yourHLQ.custom.loadlib,DISP=SHR
//LKED.SYSIN   DD   *
 ENTRY   CTABEP1
 MODE    AMODE(31),RMODE(ANY)
 NAME    IDMSCTAB(R)
/*
//*
```

yourHLQ.CAGJMAC

Data set name of the CA IDMS macro library.

zOS.maclib

Data set name of the operating system macro library.

yourHLQ.custom.loadlib

Data set name of the CA IDMS custom load library.

#GTABGEN

The #GTABGEN macro can be assembled using the following JCL.

```
//          EXEC HLASMCL
//ASM.SYSLIB   DD   DSN=yourHLQ.CAGJMAC,DISP=SHR
//          DD   DSN=zOS.maclib,DISP=SHR
//ASM.SYSIN    DD   *
       #GTABGEN macro
       END
/*
//LKED.SYSLMOD DD   DSN=yourHLQ.custom.loadlib,DISP=SHR
//LKED.SYSIN   DD   *
 ENTRY   GTABEP1
 MODE    AMODE(31),RMODE(ANY)
 NAME    IDMSGTAB(R)
/*
//*
```

yourHLQ.CAGJMAC

Data set name of the CA IDMS macro library.

zOS.maclib

Data set name of the operating system macro library.

yourHLQ.custom.loadlib

Data set name of the CA IDMS custom load library.

#SECRTT

```
//          EXEC HLASMCL
//ASM.SYSLIB   DD   DSN=yourHLQ.CAGJMAC,DISP=SHR
//          DD   DSN=zOS.maclib,DISP=SHR
//ASM.SYSIN    DD   *
       #SECRTT TYPE=INITIAL,SVCNUM=svcnumber
       #SECRTT macros
         ...
       #SECRTT TYPE=FINAL
       END
```

```
/*
//LKED.SYSLMOD DD   DSN=yourHLQ.custom.loadlib,DISP=SHR
//LKED.SYSIN   DD  *
 ENTRY   SRTTEP1
 MODE    AMODE(31),RMODE(ANY)
 NAME    RHDCSRTT(R)
/*
//*
```

yourHLQ.CAGJMAC

Data set name of the CA IDMS macro library

zOS.maclib

Data set name of the operating system macro library

yourHLQ.custom.loadlib

Data set name of the CA IDMS custom load library

#UTABGEN

The #UTABGEN macro can be assembled using the following JCL.

```
//           EXEC HLASMCL
//ASM.SYSLIB   DD   DSN=yourHLQ.CAGJMAC,DISP=SHR
//           DD   DSN=zOS.maclib,DISP=SHR
//ASM.SYSIN    DD   *
       #UTABGEN macro
       END
/*
//LKED.SYSLMOD DD   DSN=yourHLQ.custom.loadlib,DISP=SHR
//LKED.SYSIN   DD  *
 ENTRY   UTABEP1
 MODE    AMODE(31),RMODE(ANY)
 NAME    IDMSUTAB(R)
/*
//*
```

yourHLQ.CAGJMAC

Data set name of the CA IDMS macro library

zOS.maclib

Data set name of the operating system macro library

yourHLQ.custom.loadlib

Data set name of the CA IDMS custom load library

z/VSE JCL

Using MSHP

Any modifications to CA IDMS load libraries in z/VSE should be applied by MSHP.

Note: For instructions on how to assemble and link edit a module using MSHP, see the *CA IDMS Installation and Maintenance Guide—z/VSE*.

#CTABGEN

```
// DLBL    userlib
// EXTENT  ,nnnnnn
// LIBDEF  SOURCE,SEARCH=(userlib.idmslib)
// OPTION  CATAL
   PHASE   IDMSCTAB,*
// EXEC   ASMA90
          #CTABGEN macro
          END
/*
   ENTRY   CTABEP1
// DLBL    userlib
// EXTENT  ,nnnnnn
// LIBDEF  PHASE,TO=(userlib.idmslib)
// EXEC    LNKEDT
```

userlib

Filename of the user library.

nnnnnn

Volume serial number of the library.

userlib.idmslib

File identifier of the CA IDMS sublibrary.

#GTABGEN

```
// DLBL   userlib
// EXTENT ,nnnnnn
// LIBDEF SOURCE,SEARCH=(userlib.idmslib)
// OPTION CATAL
 PHASE  IDMSGTAB,*
// EXEC   ASMA90
    #GTABGEN macro
    END
/*
 ENTRY GTABEP1
// DLBL   userlib
// EXTENT ,nnnnnn
// LIBDEF PHASE,TO=(userlib.idmslib)
// EXEC   LNKEDT
```

userlib

Filename of the user library.

nnnnnn

Volume serial number of the library.

userlib.idmslib

File identifier of the CA IDMS sublibrary.

#SECRTT

```
// DLBL   userlib
// EXTENT ,nnnnnn
// LIBDEF SOURCE,SEARCH=(userlib.idmslib)
// OPTION CATAL
  PHASE  RHDCSRTT,*
// EXEC   ASMA90
         #SECRTT TYPE=INITIAL,SVCNUM=svcnumber
         #SECRTT macro
            ...
         #SECRTT TYPE=FINAL
         END
/*
  ENTRY SRTTEP1
// DLBL   userlib
// EXTENT ,nnnnnn
// LIBDEF PHASE,TO=(userlib.idmslib)
// EXEC   LNKEDT
```

userlib

Filename of the user library.

nnnnnn

Volume serial number of the library.

userlib.idmslib

File identifier of the CA IDMS sublibrary.

#UTABGEN

```
// DLBL    userlib
// EXTENT  ,nnnnnn
// LIBDEF  SOURCE,SEARCH=(userlib.idmslib)
// OPTION  CATAL
  PHASE   IDMSUTAB,*
// EXEC   ASMA90
          #UTABGEN macro
          END
/*
  ENTRY   UTABEP1
// DLBL    userlib
// EXTENT  ,nnnnnn
// LIBDEF  PHASE,CATALOG=userlib.idmslib
// EXEC    LNKEDT
```

userlib

Filename of the user library.

nnnnnn

Volume serial number of the library.

userlib.idmslib

File identifier of the CA IDMS sublibrary.

Sample IDMSUTAB

A sample IDMSUTAB.A module id provided with zero security. You may modify and assemble this source, or code your own.

CMS Commands

#CTABGEN

```
GLOBAL MACLIB idmslib OSMACRO
FILEDEF SYSLIN DISK IDMSUTAB TEXT A
ASMAHL idmsctab (NODECK OBJECT PRINT
CAE5LNKB IDMSCTAB SYSLIN userload RENT
```

idmslib

Filename of the CA IDMS MACLIB library.

idmsctab

Filename of the file containing the #CTABGEN macro source. Filetype ASSEMBLE.

userload

Filename of the output LOADLIB where IDMSCTAB will be placed.

IDMSCTAB SYSLIN is provided with the install.

#GTABGEN

```
GLOBAL MACLIB idmslib OSMACRO
FILEDEF SYSLIN DISK DBUGSTAB TEXT A
ASMAHL idmsgtab (NODECK OBJECT PRINT

CAE5LNKB IDMSGTAB SYSLIN userload RENT
```

idmslib

Filename of the CA IDMS MACLIB library

idmsgtab

Filename of the file containing the #GTABGEN macro source. Filetype ASSEMBLE.

userload

Filename of the output LOADLIB where IDMSGTAB will be placed.

IDMSCTAB SYSLIN is provided with the install.

#SECRTT

```
GLOBAL MACLIB idmslib OSMACRO
FILEDEF SYSLIN DISK RHDCSRTT TEXT A
ASMAHL rhdcsrtt (NODECK OBJECT PRINT

CAE5LNKB RHDCSRTT SYSLIN userload RENT
```

idmslib

Filename of the CA IDMS MACLIB library

rhdcsrtt

Filename of the file containing the #SECRTT macro source. Filetype ASSEMBLE

userload

Filename of the output LOADLIB where the relinked RHDCSRTT will be placed

RHDCSRTT SYSLIN is provided with the install.

#UTABGEN

```
GLOBAL MACLIB idmslib OSMACRO
FILEDEF SYSLIN DISK IDMSUTAB TEXT A
ASMAHL idmsutab (NODECK OBJECT PRINT

CAE5LNKB IDMSUTAB SYSLIN userload RENT
```

idmslib

Filename of the CA IDMS MACLIB library

idmsutab

Filename of the file containing the #UTABGEN macro source. Filetype ASSEMBLE

userload

Filename of the output LOADLIB library where IDMSUTAB will be placed

Sample IDMSUTAB

A sample IDMSUTAB ASSEMBLE module is provided with zero security. You may modify and assemble this source, or code your own.

IDMSUTAB SYSLIN is provided with the install.

Appendix B: Security Database Information and DSECTs

This section contains the following topics:

About Security Information

Dictionary Connection for Storing Information

When you define resources or manipulate privileges in CA IDMS internal security, your session should be connected to the proper dictionary:

- For global resources—The system dictionary or an application dictionary
- For system resources—The system dictionary
- For non-SQL-defined database resources:
  - DMCL, database, database name table, area—The system dictionary
  - Non-SQL-defined schema—The application dictionary
- For SQL-defined database resources—The application dictionary or system dictionary

Note: If your session is connected to an application dictionary and you issue security statements specifying resources that must be defined in the system dictionary, the statements will be processed and the security information will be stored in the application dictionary. However, at runtime, the information will not be used.

Where Security Information is Maintained

CA IDMS maintains security definitions in these areas:

- The user catalog (SYSUSER.DDLSEC area)
- The DDLDML, DDLCAT, and DDLCATX areas of the system dictionary and application dictionaries

Information about privileges on a resource is maintained in the same area as the resource definition.

User Catalog

Security information about global resources is maintained in these user catalog records:

- ATTRIBUTE
- DELUSER
- PROFILE
- RESOURCE
- RESOURCEAUTH

- USER
- USERGROUP

User catalog records reside in the SYSUSER.DDLSEC area and are accessible through subschema IDMSSECU, which is defined in dictionaries against which IDMSDIRL has been run.

System Dictionary

Security information about system resources and non-SQL-defined database resources is maintained in these system dictionary records:

- ATTRIBUTE
- PROFILE
- RESGROUPAUTH
- RESOURCE
- RESOURCEAUTH
- RESOURCEGROUP
- USERDATA

System dictionary security records reside in the SYSTEM.DDLDML area and are accessible through subschema IDMSSECS, which is defined in dictionaries against which IDMSDIRL has been run.

Security information about SQL-defined database resources is maintained in these tables of the catalog component of the dictionary:

- SYSTEM.RESOURCE
- SYSTEM.RESOURCEAUTH
- SYSTEM.RESOURCEGROUP
- SYSTEM.RESGROUPAUTH

Security tables for SQL-defined database resources reside in the DDLCAT area, and indexes on security tables reside in the DDLCATX area.

Application Dictionary

Security information about SQL-defined database resources is maintained in these tables of the catalog component of the dictionary:

- SYSTEM.RESOURCE
- SYSTEM.RESOURCEAUTH

- SYSTEM.RESOURCEGROUP
- SYSTEM.RESGROUPAUTH

Security tables for SQL-defined database resources reside in the DDLCAT area, and indexes on security tables reside in the DDLCATX area.

Security information about non-SQL-defined schemas is maintained in these system dictionary records:

- RESOURCE
- RESOURCEAUTH

These records are stored in the DDLDML area and are accessible through subschema IDMSSECS, which is defined in dictionaries against which IDMSDIRL has been run.

Summary

This table summarizes security information about CA IDMS resources, including where the information is stored and the privileges that apply to each resource:

Resource	Keyword	Location	Privileges
Global resources			
SYSADMIN	SYSA	SYSUSER.DDLSEC	SYSADMIN
User	USER	SYSUSER.DDLSEC	DEFINE (ALTER, CREATE, DISPLAY, DROP, USE(2))
Group	GROU	SYSUSER.DDLSEC	DEFINE (ALTER, CREATE, DISPLAY, DROP, USE(2))
User profile	UPRF	SYSUSER.DDLSEC	DEFINE (ALTER, CREATE, DISPLAY, DROP, USE(2))
System resources			
DCADMIN	DCA	System dictionary	DCADMIN
System	SYST	System dictionary	DEFINE (ALTER, CREATE, DISPLAY, DROP, USE(2)), SIGNON
System profile	SPRF	System dictionary	DEFINE (ALTER, CREATE, DISPLAY, DROP, USE(2))
Application(1)	SAPP	System dictionary	EXECUTE
Activity	ACTI	System dictionary	(EXECUTE on the associated 'SAPP')
Category(1)	CATE	System dictionary	EXECUTE

Resource	Keyword	Location	Privileges
Access module (runtime)	SACC	System dictionary	(EXECUTE on the associated 'CATE')
Dictionary load module	SLOD	System dictionary	(EXECUTE on the associated 'CATE')
Program (load module)	SPGM	System dictionary	(EXECUTE on the associated 'CATE')
Queue	QUEU	System dictionary	(EXECUTE on the associated 'CATE')
Run unit	NRU	System dictionary	(EXECUTE on the associated 'CATE')
Task	TASK	System dictionary	(EXECUTE on the associated 'CATE')
Non-SQL-defined database resources			
Area	AREA	System dictionary	DBAREAD, DBAWRITE, USE
Database	DB	System dictionary	DEFINE (ALTER, CREATE, DISPLAY, DROP, USE)
Database name table	DBTB	System dictionary	DEFINE (ALTER, CREATE, DISPLAY, DROP, USE)
DMCL	DMCL	System dictionary	DEFINE (ALTER, CREATE, DISPLAY, DROP, USE)
Non-SQL-defined schema	NSCH	Application dictionary (DDLDML area)	USE
SQL-defined database resources			
Schema(3)	QSCH	Application dictionary (DDLCAT area)	DEFINE (ALTER, CREATE, DISPLAY, DROP, USE(2)), OWNER
Access module (definition)	DACC	Application dictionary (DDLCAT area)	EXECUTE, DEFINE (ALTER, CREATE, DISPLAY, DROP, USE(2)), REFERENCES(2)
Table	TABL	Application dictionary (DDLCAT area)	ALL [DEFINE (ALTER, CREATE, DISPLAY, DROP, USE(2)), REFERENCES, ACCESS (DELETE, INSERT, SELECT, UPDATE), OWNER(2)]

(1)Resource group.

(2)Privilege not meaningful for resource.

(3)Resource and resource group.

Accessing Security Information

You can access security information in the following ways:

- Issuing DISPLAY statements through the CA IDMS Command Facility
- Issuing SQL DML statements through the CA IDMS Command Facility
- Issuing navigational or SQL DML statements to access the data from CA technologies OLQ, CA technologies Culprit, or a user-written program

Security DISPLAY statements

What You Can Do

You can submit to the CA IDMS Command Facility DISPLAY or PUNCH statements to display this security information:

- About global resources:
  - Holders of SYSADMIN privilege
  - User information, including definitions, groups, and privileges
  - Group information, including definitions, users, and privileges
  - User profile information, including attributes
- About system resources:
  - Holders of DCADMIN privilege
  - Resource definitions
  - Privileges granted on resources
- About database resources:
  - Resource definitions
  - Privileges granted on resources

Documentation of DISPLAY Statements

For documentation of DISPLAY statement syntax, see Syntax for Security Display Statements (see page 339).

DML Access

Navigational DML

For navigational DML access to a security database, you specify the appropriate security subschema and issue DML statements to navigate the security records that are described later in this appendix.

You can issue navigational DML statements through CA OLQ, CA Culprit, or a user-written program to access the user catalog through subschema IDMSSECU and other security definitions in the system dictionary through subschema IDMSSECS. You cannot access SQL-defined database resources through navigational DML.

More Information

- For more information about using CA OLQ, see the *CA OLQ User Guide*.
- For more information about using CA Culprit, see the *CA Culprit for CA IDMS User Guide.*
- For more information about programming with navigational DML, see the *CA IDMS Navigational DML programming Guide* and the language-specific reference manual.

SQL DML

For SQL access to information about global and system resources, you define SQL schemas for the non-SQL-defined security schemas (IDMSSECU and IDMSSECS) and issue SQL statements that specify records in the security databases as tables.

Security information about SQL-defined database resources resides in the SQL-defined tables in the SYSTEM schema that are described later in this appendix.

More Information

- For more information about using the Command Facility to issue SQL statements, see the *CA IDMS Common Facilities Guide*.
- For more information about accessing a non-SQL-defined database with SQL, see the *CA IDMS SQL Reference Guide*.
- For more information about programming with SQL, see the *CA IDMS SQL Programming Guide*.

Global and System Resource Security Records

This section contains the descriptions of records containing security information about global resources, system resources, and non-SQL-defined database resources.

ATTRIBUTE

Purpose

An occurrence of this record represents an attribute defined in a profile.

Access

- IDMSSECU subschema (user profiles)
- IDMSSECS subschema (system profiles)
- Location mode: VIA PROFILE-ATTR
- Member of PROFILE-ATTR set

Record elements

Element name	Picture and usage	Description of contents
PROFILENAME	PIC X(18)	Name of the profile.
KEYWORD	PIC X(8)	The keyword that identifies the attribute.
LENGTH	PIC S9(4) USAGE COMP	Number of characters in the attribute value.
VALUE	PIC X(32)	A user-assigned value for the attribute.
OVERRIDE	PIC X(1)	Defines whether the value may be overridden at runtime: ■ 'Y'—Value may be overridden at runtime ■ 'N'—Value is static and may not be overridden
CTIME	PIC X(64) USAGE BIT	Time created.
UTIME	PIC X(64) USAGE BIT	Time of the last update.
CUSER	PIC X(18)	ID of the user who created the attribute.
UUSER	PIC X(18)	ID of the user who last updated the attribute.
FILLER	PIC X(13)	(Reserved—initialized to spaces.)

DELUSER

Purpose

An occurrence of this record represents a user or group that has been logically deleted from the user catalog. The SDEL task uses this information to determine which resource and resource group authorizations should be physically deleted.

Access

- IDMSSECU subschema
- Location mode: VIA IX-DELUSER
- Member of IX-DELUSER set

Record Elements

Element name	Picture and usage	Description of contents
AUTHID	PIC X(18)	The authorization ID of a user or group that has been logically deleted.
CTIME	PIC X(64) USAGE BIT	Time created.
UTIME	PIC X(64) USAGE BIT	Time of the last update.
CUSER	PIC X(18)	ID of the user who created the record occurrence.
UUSER	PIC X(18)	ID of the user who last updated the record occurrence.
FILLER	PIC X(10)	(Reserved—initialized to spaces.)

PROFILE

Purpose

An occurrence of this record represents a profile definition.

Access

- Subschema IDMSSECU (user profiles)
- Subschema IDMSSECS (system profiles)
- Location mode: CALC on PROFILENAME
- Owner of PROFILE-ATTR set

Record Elements

Element name	Picture and usage	Description of contents
PROFILENAME	PIC X(18)	The name of the profile.
TYPE	PIC X(1)	The profile type: ■ 'U'—User profile ■ 'S'—System profile
CTIME	PIC X(64) USAGE BIT	Time created.
UTIME	PIC X(64) USAGE BIT	Time of the last update.
CUSER	PIC X(18)	ID of the user who created the profile.
UUSER	PIC X(18)	ID of the user who last updated the profile.
FILLER	PIC X(17)	(Reserved—initialized to spaces.)

RESGROUPAUTH

Purpose

An occurrence of this record represents an execute privilege granted to an authorization ID on a category or on application activities.

Access

- Subschema IDMSSECS
- Location mode: CALC on AUTHID
- Member of RESGROUP-AUTH set

Record Elements

Element name	Picture and usage	Description of contents
AUTHID	PIC X(18)	The authorization ID of the user or group that holds EXECUTE privilege.
RESOURCETYPE	PIC X(4)	The code for the resource group to which the privilege applies: ■ 'SAPP'—Application ■ 'CATE'—Category

Element name	Picture and usage	Description of contents
RESOURCENAME	PIC X(60)	The name of the category or the name of the application with which the activities are associated.
RUNTIMEAUTH	PIC S9(4) USAGE COMP SYNC	Runtime privilege that has been granted on the category or activity: ■ 1—EXECUTE
RUNTIMEAUTHW	PIC S9(4) USAGE COMP SYNC	(Not applicable—always 0)
DEFNAUTH	PIC S9(4) USAGE COMP SYNC	(Not applicable—always 0)
DEFNAUTHW	PIC S9(4) USAGE COMP SYNC	(Not applicable—always 0)
OTHERAUTH	PIC S9(4) USAGE COMP SYNC	(Not applicable—always 0)
FUNCTIONS	PIC X(32)	A bit map that designates the activities within the application in RESOURCENAME to which the EXECUTE privilege applies.
CTIME	PIC X(64) USAGE BIT	Time created.
UTIME	PIC X(64) USAGE BIT	Time of the last update.
CUSER	PIC X(18)	ID of the user who created the resource group authorization.
UUSER	PIC X(18)	ID of the user who last updated the resource group authorization.
FILLER	PIC X(16)	(Reserved—initialized to spaces.)

RESOURCE

Purpose

An occurrence of this record represents a resource.

Access

- Subschema IDMSSECU (global resources)
- Subschema IDMSSECS (system and non-SQL-defined resources)
- Location mode: CALC on RESOURCETYPE and RESOURCENAME
- Member of IX-RESOURCE set
- Member of RESGROUP-RES set (IDMSSECS subschema only)
- Owner of RESOURCE-AUTH set

Record Elements

Element name	Picture and usage	Description of contents
RESOURCETYPE	PIC X(4)	Resource type keyword. Using IDMSSECS subschema: 'ACTI' 'QUEU' 'AREA' 'SACC' 'DB' 'SLOD' 'DBTB' 'SPGM' 'DCA' 'SPRF' 'DMCL' 'SYST' 'NRU' 'TASK' 'NSCH' Using IDMSSECU subschema: 'GROU' 'UPRF' 'SYSA' 'USER'
RESOURCENAME	PIC X(60)	The resource name. For RESOURCETYPE 'ACTI', the value is the fully qualified activity name (for example, DCMT.SHUTDOWN).
GROUPTYPEIND	PIC X(8) USAGE BIT	Null value indicator for the GROUPTYPE field: ■ B'11111111'—Resource does not belong to a group
GROUPTYPE	PIC X(4)	The group code of the resource group (if there is one) to which the resource belongs: ■ 'CATE'—Category ■ 'SAPP'—Application ■ ' '—Resource does not belong to a group
GROUPNAMEIND	PIC X(8) USAGE BIT	Null value indicator for the GROUPNAME field: ■ B'11111111'—No value in GROUPNAME
GROUPNAME	PIC X(60)	The name of the resource group (if there is one) to which the resource belongs.

Element name	Picture and usage	Description of contents
NUMBER	PIC S9(4) USAGE COMP SYNC	For a resource assigned to a category (NRU, QUEU, SACC, SLOD, SPGM, or TASK), the unique number that identifies the category with which the resource is associated. For an activity, the unique activity number within the application. For a resource that is not an activity or does not belong to a category, the value is 0.
AMRUNNABLE	PIC X(1)	(Reserved—initialized to 'N'.)
STATUS	PIC X(1)	(Reserved—initialized to 'A'.)
AMGRANTABLE	PIC X(1)	(Reserved—initialized to 'N'.)
AMLASTCHANGE	PIC X(8)	(Reserved—initialized to binary zeros.)
CTIME	PIC X(64) USAGE BIT	Time created.
UTIME	PIC X(64) USAGE BIT	Time of the last update.
CUSER	PIC X(18)	ID of the user who created the resource.
UUSER	PIC X(18)	ID of the user who last updated the resource.
FILLER	PIC X(21)	(Reserved—initialized to spaces.)

RESOURCEAUTH

Purpose

An occurrence of this record represents the privileges that have been granted to an authorization ID on a resource.

Access

- Subschema IDMSSECU (privileges on global resources)
- Subschema IDMSSECS (privileges on system resources and non-SQL-defined database resources)
- CALC on AUTHID
- Member of RESOURCE-AUTH set

Record Elements

Element name	Picture and usage	Description of contents
AUTHID	PIC X(18)	The authorization ID of the user or group to whom the privileges have been granted.
RESOURCETYPE	PIC X(4)	The resource type keyword. Using IDMSSECS subschema: 'AREA' 'DMCL' 'DB' 'NSCH' 'DBTB' 'SPRF' 'DCA' 'SYST' Using IDMSSECU subschema: 'GROU' 'UPRF' 'SYSA' 'USER'
RESOURCENAME	PIC X(60)	The resource name.
RUNTIMEAUTH	PIC S9(4) USAGE COMP SYNC	(Not applicable—initialized to binary zeros. EXECUTE privileges on categories and activities are stored in the RESGROUPAUTH record.)
RUNTIMEAUTHW	PIC S9(4) USAGE COMP SYNC	(Not applicable—initialized to binary zeros.)
DEFNAUTH	PIC S9(4) USAGE COMP SYNC	Definition privileges that have been granted on the resource: ■ 1—CREATE ■ 2—ALTER ■ 4—DROP ■ 8—DISPLAY ■ 16—USE ■ *nn*—(The sum of two or more of the above, representing multiple privileges) ■ 31—DEFINE (all definition privileges) Note: DEFINE privilege always includes USE; however, USE is meaningful only for AREA, DB, DBTB, DMCL, and NSCH.
DEFNAUTHW	PIC S9(4) USAGE COMP SYNC	Definition privileges a user represented by AUTHID may grant to other authorization IDs.

Element name	Picture and usage	Description of contents
OTHERAUTH	PIC S9(4) USAGE COMP SYNC	Special privileges: ■ 1—SYSADMIN (Resource type SYSA only) ■ 4—DBADMIN (Resource type DB only) ■ 8—DCADMIN (Resource type DCA only) ■ 16—SIGNON (Resource type SYST only) ■ 32—DBAREAD (Resource type AREA only) ■ 64—DBAWRITE (Resource type AREA only) ■ 96—DBAREAD and DBAWRITE
CTIME	PIC X(64) USAGE BIT	Time created.
UTIME	PIC X(64) USAGE BIT	Time of the last update.
CUSER	PIC X(18)	ID of the user who created the resource authorization.
UUSER	PIC X(18)	ID of the user who last updated the resource authorization.
FILLER	PIC X(16)	(Reserved—initialized to spaces.)

RESOURCEGROUP

Purpose

An occurrence of this record represents an application or category that has been defined with a CREATE RESOURCE statement. Applications are defined when the first CREATE RESOURCE ACTIVITY *application-name.activity-name* statement is issued.

Access

- Subschema IDMSSECS
- Location mode: CALC on RESOURCETYPE and RESOURCENAME
- Member of IX-RESGROUP set; owner of RESGROUP-AUTH and RESGROUP-RES sets

Record Elements

Element name	Picture and usage	Description of contents
RESOURCETYPE	PIC X(4)	The code for the resource group: ■ 'CATE'—Category ■ 'SAPP'—Application
RESOURCENAME	PIC X(60)	The name of the application or category.
OWNER	PIC X(18)	(Not applicable—initialized to spaces.)
CATEGORYNO	PIC S9(4) USAGE COMP SYNC	For categories, the unique number assigned to the category. For applications, the value is 0.
FUNCTIONASSIGNMT	PIC X(1)	How activity numbers are assigned: ■ 'S'—System-assigned ■ 'U'—User-assigned ■ ' '—(Resource type is 'CATE')
CTIME	PIC X(64) USAGE BIT	Time created.
UTIME	PIC X(64) USAGE BIT	Time of the last update.
CUSER	PIC X(18)	ID of the user who created the resource group.
UUSER	PIC X(18)	ID of the user who last updated the resource group.
FILLER	PIC X(19)	(Reserved—initialized to spaces.)

USER

Purpose

An occurrence of this record represents a user or group defined within a CA IDMS security domain. Users and groups are defined by the same record so that privileges on resources—whether to a user or a group—can be granted and enforced in the same manner. Note that if the occurrence represents a group, there is no associated password or user profile.

Access

- IDMSSECU subschema
- Location mode: CALC on AUTHID
- Owner of the USER-GROUP and GROUP-USER sets

Record elements

Element name	Picture and usage	Description of contents
AUTHID	PIC X(18)	The authorization ID of a user or group.
TYPE	PIC X(1)	Identifies the authorization ID as representing a user or group: ■ 'G'—Group
STATUS	PIC X(1)	Current status of the authorization ID: ■ 'A'—Active ■ 'D'—Logically deleted
PASSWORD	PIC X(8)	The encrypted form of the plain text password, or: ■ Binary zeros, when TYPE field is 'U' and the password unassigned or null ■ Spaces, when TYPE field is 'G' or when TYPE field is 'U' and AUTHID is 'SYSTEM'
NAME	PIC X(32)	The full name associated with the user, if specified; initialized to spaces when not specified for the user or when the authorization ID represents a group.

Element name	Picture and usage	Description of contents
DESCRIPTION	PIC X(40)	Textual information that the security administrator may associate with the authorization ID but that has no special meaning to the security system; initialized to spaces when not specified for the user or group.
FILLER	PIC X(2)	(Reserved—initialized to spaces.)
PROFILENAME	PIC X(18)	The name of the user profile assigned to this authorization ID; initialized to spaces for all groups and when not specified for a user.
FILLER	PIC X(4)	(Reserved—initialized to spaces.)
FILLER	PIC X(8)	(Reserved—initialized to binary zeros.)
GTIME	PIC X(64) USAGE BIT	(Reserved—initialized to binary zeros.)
PTIME	PIC X(64) USAGE BIT	(Reserved—initialized to binary zeros.)
CTIME	PIC X(64) USAGE BIT	Time created.
UTIME	PIC X(64) USAGE BIT	Time of the last update.
CUSER	PIC X(18)	ID of the user who created the user definition.
UUSER	PIC X(18)	ID of the user who last updated the user definition.
FILLER	PIC X(16)	(Reserved—initialized to spaces.)

Special Record Occurrences

These are special occurrences of the USER record:

- Group 'PUBLIC'—created on the first grant of privilege to PUBLIC (group 'PUBLIC' cannot be dropped):
  - AUTHID: 'PUBLIC'
  - TYPE: 'G'
  - STATUS: (space)
  - PASSWORD: (spaces)
  - NAME: (spaces)
  - DESCRIPTION: 'System Category bit map owner'
  - PROFILENAME: (spaces)
  - CUSER: 'SYSTEM'
  - UUSER: 'SYSTEM'
  - Other fields as noted in the USER record description above
- User 'SYSTEM'—created by the first CREATE RESOURCE CATEGORY statement (user 'SYSTEM' cannot be dropped):
  - AUTHID: 'SYSTEM'
  - TYPE: 'U'
  - STATUS: (space)
  - PASSWORD: (spaces)
  - NAME: (spaces)
  - DESCRIPTION: 'PUBLIC Group'
  - PROFILENAME: (spaces)
  - CUSER: 'SYSTEM'
  - UUSER: 'SYSTEM'
  - Other fields as noted in the USER record description above

USERDATA

Purpose

An occurrence of this record can represent:

- The association of a user or group with all categories on which the user or group holds execution privilege
- In one case, an occurrence, with the value 'SYSTEM' in the AUTHID field and 'C' in the TYPE field, that represents all categories defined to the system
- The association of a user and a system profile, as specified in a GRANT SIGNON statement

Access

- IDMSSECS subschema
- Location mode: CALC on AUTHID
- Sets: None

Record Elements

Element name	Picture and usage	Description of contents
AUTHID	PIC X(18)	The authorization ID of the user.
LENGTH	PIC S9(4) USAGE COMP	The number of bytes (a multiple of 32) currently allocated to the category bit map; initialized to 0 when TYPE is 'P'.
TYPE	PIC X(1)	Type of user data: ■ 'C'—Category ■ 'P'—Profile
SYSTEM	PIC X(8)	The identifier of a CA IDMS system; initialized to spaces when TYPE is 'C'.
PROFILENAME	PIC X(18)	The name of the system profile; initialized to spaces when TYPE is 'C'.
CTIME	PIC X(64) USAGE BIT	Time created.
UTIME	PIC X(64) USAGE BIT	Time of the last update.
CUSER	PIC X(18)	ID of the user who created the record occurrence.
UUSER	PIC X(18)	ID of the user who last updated the record occurrence.
FILLER	PIC X(5)	(Reserved—initialized to spaces.)

Element name	Picture and usage	Description of contents
CATEGORIES	PIC X OCCURS 0 TO 4096 DEPENDING ON LENGTH	The bit map used to identify categories. Each bit in the bit map corresponds to a category number from 1 to 32,767.

Special Record Occurrence

The following occurrence of the USERDATA record is created by the first CREATE RESOURCE CATEGORY statement and is used to manage category number assignment:

- AUTHID: 'SYSTEM'
- LENGTH: (variable—initially 32)
- TYPE: 'C'
- SYSTEM: (spaces)
- PROFILENAME: (spaces)
- CUSER: 'SYSTEM'
- UUSER: 'SYSTEM'
- CATEGORIES: (variable—initially 32 bytes with high-order bit on and all other bits off)
- Other fields as noted in the USERDATA record description above

USERGROUP

Purpose

An occurrence of this record represents the association of a user with a group.

Access

- IDMSSECU subschema
- Location mode: VIA USER-GROUP
- Member of the USER-GROUP and GROUP-USER sets

Record Elements

Element name	Picture and usage	Description of contents
GROUPID	PIC X(18)	The authorization ID of the group of which the user is a member.
USERID	PIC X(18)	The authorization ID of the user who is a member of the group.

Element name	Picture and usage	Description of contents
DEFAULTGROUP	PIC X(1)	Default group indicator: ■ 'D'—Default group ■ ' '—Not the default group
CTIME	PIC X(64) USAGE BIT	Time created.
UTIME	PIC X(64) USAGE BIT	Time of the last update.
CUSER	PIC X(18)	ID of the user who created the user and group association.
UUSER	PIC X(18)	ID of the user who last updated the user and group association.
FILLER	PIC X(11)	(Reserved—initialized to spaces.)

Database Resource Security Tables

This section contains the descriptions of tables maintained in the catalog component of the dictionary for use by CA IDMS centralized security.

SYSTEM.RESGROUPAUTH

Purpose

A row of this table represents the privileges that have been granted to an authorization ID on an SQL-defined database resource group. A row of this table is stored only when a CREATE SCHEMA statement is issued.

Columns

Column name	Data type	Description of contents
AUTHID	CHAR(18)	The authorization ID of the user or group that holds one or more privileges on the resource group.
RESOURCETYPE	CHAR(4)	The code for the resource group to which the privileges apply: ■ 'QSCH'—SQL Schema
RESOURCENAME	CHAR(60)	The schema name.

Column name	Data type	Description of contents
RUNTIMEAUTH	BINARY(2)	Runtime privileges that have been granted on the schema: ■ 143—ALL (ACCESS and OWNER privileges) When a CREATE SCHEMA statement is executed, a row is inserted into SYSTEM.RESGROUPAUTH. AUTHID is the ID of the user executing the statement, and RUNTIMEAUTH is 143 (ALL). If ownership of the schema is transferred, AUTHID is modified but RUNTIMEAUTH remains 143.
RUNTIMEAUTHW	BINARY(2)	The runtime privileges a user represented by AUTHID may grant to other authorization IDs: ■ 143—ALL ALL includes schema ownership, which can be transferred, and these access privileges, which can be granted as appropriate on tables and access modules: ■ SELECT ■ EXECUTE ■ INSERT ■ UPDATE ■ DELETE
DEFNAUTH	BINARY(2)	Definition privileges that have been granted on the schema and the resources associated with the schema: ■ 191—ALL (DEFINE, REFERENCES, and OWNER)

Column name	Data type	Description of contents
DEFNAUTHW	BINARY(2)	The definition privileges a user represented by AUTHID may grant to other authorization IDs: ■ 191—ALL (definition privileges, REFERENCES, and OWNER) ALL includes schema ownership, which can be transferred, and these privileges, which can be granted as appropriate on tables and access modules: ■ CREATE ■ ALTER ■ DROP ■ DISPLAY ■ USE ■ REFERENCES
OTHERAUTH	BINARY(2)	(Not applicable—initialized to binary zeros.)
FUNCTIONS	BINARY(32)	(Not applicable—initialized to binary zeros.)
CTIME	TIMESTAMP	Time created.
UTIME	TIMESTAMP	Time of the last update.
CUSER	CHAR(18)	ID of the user who created the resource group authorization.
UUSER	CHAR(18)	ID of the user who last updated the resource group authorization.
FILLER	BINARY(16)	(Reserved—initialized to spaces.)

SYSTEM.RESOURCE

Purpose

A row of this table represents a resource (a table or access module) associated with a schema, or the schema itself.

Columns

Column name	Data type	Description of contents
RESOURCETYPE	CHAR(4)	Resource type keyword: ■ 'DACC'—Access module ■ 'QSCH'—SQL Schema ■ 'TABL'—Table
RESOURCENAME	CHAR(60)	The resource name; for a table or access module, the name is fully qualified with *schema-name*.
GROUPTYPE	CHAR(4)	The resource group to which the resource belongs: ■ 'QSCH'—SQL schema ■ Null when RESOURCETYPE is 'QSCH'
GROUPNAME	CHAR(60)	The name of the resource group (SQL Schema) to which the resource belongs; null when RESOURCETYPE is 'QSCH'
NUMBER	SMALLINT	(Not applicable—initialized to zero.)
AMRUNNABLE	CHAR(1)	(Reserved—initialized to 'N'.)
STATUS	CHAR(1)	(Reserved—initialized to 'A'.)
AMGRANTABLE	CHAR(1)	(Reserved—initialized to 'N'.)
AMLASTCHANGE	TIMESTAMP	(Reserved—initialized to binary zeros.)
CTIME	TIMESTAMP	Time created.
UTIME	TIMESTAMP	Time of the last update.
CUSER	CHAR(18)	ID of the user who created the resource.
UUSER	CHAR(18)	ID of the user who last updated the resource.
FILLER	BINARY(21)	(Reserved—initialized to spaces.)

SYSTEM.RESOURCEAUTH

Purpose

A row of this table represents the privileges that have been granted to an authorization ID on an SQL-defined database resource.

Columns

Column name	Data type	Description of contents
AUTHID	CHAR(18)	The authorization ID of the user or group to whom the privileges have been granted.
RESOURCETYPE	CHAR(4)	The resource type keyword: ■ 'DACC'—Access module ■ 'QSCH'—Schema ■ 'TABL'—Table
RESOURCENAME	CHAR(60)	The resource name.
RUNTIMEAUTH	BINARY(2)	Runtime privileges that have been granted on the resource. For RESOURCETYPE 'DACC': ■ 1—EXECUTE(1) For RESOURCETYPE 'QSCH'(2) and 'TABL': ■ 1—SELECT ■ 2—INSERT ■ 4—UPDATE ■ 8—DELETE ■ *nn*—(The sum of two or more of the above, representing multiple privileges) ■ 15—ACCESS (all of the above) ■ 128—OWNER(3) ■ 143—ALL (ACCESS and OWNER)
RUNTIMEAUTHW	BINARY(2)	The runtime privileges a user represented by AUTHID may grant to other authorization IDs.(4)

Column name	Data type	Description of contents
DEFNAUTH	BINARY(2)	Definition privileges that have been granted on the resource: ■ 1—CREATE ■ 2—ALTER ■ 4—DROP ■ 8—DISPLAY ■ 16—USE ■ *nn*—(The sum of two or more of the above, representing multiple privileges) ■ 31—DEFINE (CREATE, ALTER, DROP, DISPLAY, USE) ■ 32—REFERENCES ■ 63—DEFINE and REFERENCES ■ 128—OWNER(5) ■ 191—ALL (DEFINE, REFERENCES, and OWNER)(6)
DEFNAUTHW	BINARY(2)	The definition privileges a user represented by AUTHID may grant to other authorization IDs.(5) (6)
OTHERAUTH	BINARY(2)	(Not applicable—initialized to binary zeros.)
CTIME	TIMESTAMP	Time created.
UTIME	TIMESTAMP	Time of the last update.
CUSER	CHAR(18)	ID of the user who created the resource authorization.
UUSER	CHAR(18)	ID of the user who last updated the resource authorization.
FILLER	BINARY(16)	(Reserved—initialized to spaces.)

(1) An occurrence of a row with RESOURCETYPE 'DACC' and RUNTIMEAUTH 1 (EXECUTE) results when privilege is granted on the access module directly as an SQL-defined database resource, not as part of a category, which is a system resource.

(2) An occurrence of a row with RESOURCETYPE 'QSCH' and a nonzero value in RUNTIMEAUTH results only when a schema is created. RUNTIMEAUTH is 143 (ALL) and remains 143 even if schema ownership is transferred because the runtime privileges cannot be revoked. All other occurrences of a row with RESOURCETYPE 'QSCH' refer to definition privileges only and, therefore, contain zero in RUNTIMEAUTH and RUNTIMEAUTHW.

(3) The OWNER privilege for a table can be granted only with a GRANT ALL PRIVILEGES statement and revoked only with a REVOKE ALL PRIVILEGES statement. However, it is possible for the value of RUNTIMEAUTH (and RUNTIMEAUTHW) to be 128 (OWNER) if all privileges are granted on a table and then access privileges are revoked.

(4) Runtime privileges for RESOURCETYPE 'QSCH' are grantable only in the sense that they are automatically transferred when ownership of the schema is transferred.

(5) OWNER privilege is never assigned to DEFNAUTH (or DEFNAUTHW) for resource types 'DACC' and 'TABL'.

(6) When an SQL schema is created, the value of RESOURCETYPE is 'QSCH' and the value of DEFNAUTH is 191 (ALL: DEFINE, REFERENCES, and OWNER). REFERENCES and definition privileges can be revoked; ownership can be transferred but not revoked.

SYSTEM.RESOURCEGROUP

Purpose

A row of this table represents an SQL schema resource group. A row of this table is stored when one of these events occurs:

- The first GRANT statement is issued for an access module or table associated with an undefined schema.
- A CREATE SCHEMA statement is issued before any GRANT statements for associated access modules or tables.

Columns

Column name	**Data type**	**Description of contents**
RESOURCETYPE	CHAR(4)	The code for the resource group: ■ 'QSCH'—SQL schema
RESOURCENAME	CHAR(60)	The schema name.

Column name	Data type	Description of contents
OWNER	CHAR(18)	The authorization ID of the user or group who owns the schema.
CATEGORYNO	SMALLINT	(Not applicable—initialized to zero.)
FUNCTIONASSIGNMT	BINARY(1)	(Not applicable—initialized to space.)
CTIME	TIMESTAMP	Time created.
UTIME	TIMESTAMP	Time of the last update.
CUSER	CHAR(18)	ID of the user who created the resource group.
UUSER	CHAR(18)	ID of the user who last updated the resource group.
FILLER	BINARY(19)	(Reserved—initialized to spaces.)

#SATTDS

```
      COPY  #SATTDS
*********************************************************************
***                                             ***
***    SATT DSECT                                         ***
***                                             ***
***    PROFILE ATTRIBUTE                                        ***
***                                             ***
***     COPYRIGHT (C) 2007 CA technologies. ALL RIGHTS RESERVED.            ***
***                                             ***
***                                             ***
***                                             ***
*********************************************************************
```

Offset Value

```
000000       SATT     DSECT                              11/16/90
000000       SATTPROF DS    CL18        PROFILE NAME
000012       SATTKEYW DS    CL8         ATTRIBUTE KEYWORD
00001A       SATTLTH  DS    H           ATTRIBUTE LENGTH
00001C       SATTVALU DS    CL32        ATTRIBUTE VALUE
00003C       SATTOVER DS    CL1         ATTRIBUTE OVERRIDE FLAG
          *                   * 'Y' IF RUNTIME OVERRIDE ALLOWED
          *                   * 'N' IF OTHERWISE
00003D       SATTCTIM DS    CL8         TIME/DATE ATTRIBUTE WAS CREATED
000045       SATTUTIM DS    CL8         TIME/DATE ATTRIBUTE WAS LAST UPDATED
00004D       SATTCAID DS    CL18        CREATOR OF THE ATTRIBUTE
00005F       SATTUAID DS    CL18        LAST UPDATOR OF THE ATTRIBUTE
000071                DS    CL13        * UNUSED
     0007E  SATTDSLN EQU   *-SATT       * LENGTH OF DSECT
          *******************************************************************
```

#SDUSDS

```
         COPY  #SDUSDS
***********************************************************************
***                                             ***
***    SDELUDS DSECT                                  ***
***                                             ***
***    USER                                          ***
***                                             ***
***     COPYRIGHT (C) 2007 CA technologies. ALL RIGHTS RESERVED.         ***
***                                             ***
***                                             ***
***                                             ***
***********************************************************************
```

Offset Value

```
000000       SDUS     DSECT                              06/24/91
000000       SDUSAUID DS    CL18        AUTHORIZATION ID
000012       SDUSCTIM DS    CL8         TIME/DATE AUTH. ID WAS CREATED
00001A       SDUSUTIM DS    CL8         TIME/DATE AUTH. ID WAS LAST UPDATED
000022       SDUSCAID DS    CL18        CREATOR OF THE AUTH. ID.
000034       SDUSUAID DS    CL18        LAST UPDATOR OF THE AUTH. ID
000046                DS    XL10        * UNUSED
     00050  SDUSDSLN EQU   *-SDUS       * LENGTH OF DSECT
          *********************************************************************
```

#SECACAB

```
      COPY  #SECACAB
***********************************************************************
***     SECACAB - SECURITY ACCESS AUTHORITY BYTES               ***
*** -------------------------------------------------------------- ***
***     THIS DSECT COVERS THE 6 BYTES OF ACCESS AUTHORITY        ***
***     FLAGS FOUND IN THE RESOURCE-TYPE RECORD/BLOCK, THE       ***
***     RESOURCE RECORD, THE AUTHORITY RECORD, AND VARIOUS FIELDS ***
***     IN THE SECURITY REQUEST BLOCK (#SECRB)                   ***
***                                                             ***
***      COPYRIGHT (C) 2007 CA technologies. ALL RIGHTS RESERVED.  ***
***                                                             ***
***                                                             ***
***                                                             ***
***********************************************************************
```

Offset Value

```
000000        SECACAB  DSECT                     06/08/89 13:35:22
          ******** RUN-TIME AUTHORITIES ********
          SAARSEL  #FLAG SACRSEL            SELECT
000000        SAARSELI DS    0XL1
       00001  SAARSELM EQU   SACRSEL
          SAAREXE  #FLAG SACREXE            EXECUTE
000000        SAAREXEI DS    0XL1
       00001  SAAREXEM EQU   SACREXE
          SAARINS  #FLAG SACRINS            INSERT
000000        SAARINSI DS    0XL1
       00002  SAARINSM EQU   SACRINS
          SAARUPD  #FLAG SACRUPD            UPDATE
000000        SAARUPDI DS    0XL1
       00004  SAARUPDM EQU   SACRUPD
          SAARDEL  #FLAG SACRDEL            DELETE
000000        SAARDELI DS    0XL1
       00008  SAARDELM EQU   SACRDEL
          SAAAOWN  #FLAG SACROWN            OWNER
000000        SAAAOWNI DS    0XL1
       00080  SAAAOWNM EQU   SACROWN
          SAARALL  #FLAG SACRALL            ALL
000000        SAARALLI DS    0XL1
       0008F  SAARALLM EQU   SACRALL
000000        SAARUN   DS    XL1            RUN-TIME
000001        SAARUNWG DS    XL1            WITH GRANT
          ******** DEFINITION AUTHORITIES ********
          SAADCRE  #FLAG SACDCRE            CREATE
000002        SAADCREI DS    0XL1
       00001  SAADCREM EQU   SACDCRE
          SAADALT  #FLAG SACDALT            ALTER
```

```
000002     SAADALTI DS   0XL1
    00002  SAADALTM EQU  SACDALT
       SAADDRO  #FLAG SACDDRO       DROP
000002     SAADDROI DS   0XL1
    00004  SAADDROM EQU  SACDDRO
       SAADDIS  #FLAG SACDDIS       DISPLAY
000002     SAADDISI DS   0XL1
    00008  SAADDISM EQU  SACDDIS
       SAADREF  #FLAG SACDREF       REFERENCE
000002     SAADREFI DS   0XL1
    00020  SAADREFM EQU  SACDREF
       SAADUSE  #FLAG SACDUSE       RUN
000002     SAADUSEI DS   0XL1
    00010  SAADUSEM EQU  SACDUSE
       SAADDEF  #FLAG SACDDEF       DEFINE
000002     SAADDEFI DS   0XL1
```

Offset Value

```
    0001F  SAADDEFM EQU  SACDDEF
000002     SAADEF   DS   XL1       DEFINITIONAL
000003     SAADEFWG DS   XL1           WITH GRANT
       ******** ADMINISTRATIVE AUTHORITIES ********
       SAAASYS  #FLAG SACASYS       SYSADMIN
000004     SAAASYSI DS   0XL1
    00001  SAAASYSM EQU  SACASYS
       SAAADDA  #FLAG SACADDA       DDADMIN
000004     SAAADDAI DS   0XL1
    00002  SAAADDAM EQU  SACADDA
       SAAASIG  #FLAG SACASIG       SIGNON
000004     SAAASIGI DS   0XL1
    00010  SAAASIGM EQU  SACASIG
       SAAADCA  #FLAG SACADCA       DC ADMIN
000004     SAAADCAI DS   0XL1
    00008  SAAADCAM EQU  SACADCA
       SAAADBA  #FLAG SACADBA       DBA ADMIN
000004     SAAADBAI DS   0XL1
    00004  SAAADBAM EQU  SACADBA
       SAAADBW  #FLAG SACADBW       DBA WRITE
000004     SAAADBWI DS   0XL1
    00040  SAAADBWM EQU  SACADBW
       SAAADBR  #FLAG SACADBR       DBA READ
000004     SAAADBRI DS   0XL1
    00020  SAAADBRM EQU  SACADBR
       SAAAALL  #FLAG SACAALL       ALL ADMIN AUTHORITIES
000004     SAAAALLI DS   0XL1
    0007F  SAAAALLM EQU  SACAALL
000004     SAAADM   DS   XL1
       ******** UNUSED AUTHORITIES ********
000005          DS   XL1
```

```
00006  SAALNG   EQU   *-SECACAB    LENGTH OF ACCESS AUTHORITY BYTES
```

#SECEQU

```
      COPY  #SECEQU
*********************************************************************
***                                             ***
***   SECEQU - GENERALLY USEFUL EQUATES FOR THE ENTIRE SECURITY  ***
***                                             ***
***    COPYRIGHT (C) 2007 CA technologies. ALL RIGHTS RESERVED.          ***
***                                             ***
***                                             ***
***                                             ***
*********************************************************************
*---------------------------------------------------------------------
*  PRIMARY FUNCTION CODES
*---------------------------------------------------------------------
*           1                Reserved by security central
```

Offset Value

```
00002  SFCSINIT EQU   2              System initialization
00003  SFCCHECK EQU   3              Authorization check
00004  SFCSGNON EQU   4              User SIGNON -
00005  SFCSGNOF EQU   5              User SIGNOFF
00006  SFCBLDNM EQU   6              Build fully qualified res name
00007  SFCXTRCT EQU   7              Get job card user-id
00008  SFCSTATS EQU   8              Get ESM status
00009  SFCCRYPT EQU   9              Password encryption
0000A  SFCPWVER EQU   10             Password reverification
0000B  SFCAPMAP EQU   11             Get/build application bitmap
0000C  SFCBULK  EQU   12             Bulk check - access mod security
0000D  SFCAMCHK EQU   13             Access module runnability check
     *---------------------------------------------------------------------
     *  SECONDARY FUNCTION CODES (FOR UTILITY ROUTINES)
     *---------------------------------------------------------------------
00002  SUTLFLDL EQU   2              Backscan variable length field
00004  SUTLBLDN EQU   4              Build fully qualified resname
     *---------------------------------------------------------------------
     *  ACCESS AUTHORITY BITS
     *---------------------------------------------------------------------
00001  SACRSEL  EQU   X'01'          RUN-TIME - SELECT
00001  SACREXE  EQU   X'01'                   - EXECUTE
00002  SACRINS  EQU   X'02'                   - INSERT
00004  SACRUPD  EQU   X'04'                   - UPDATE
00008  SACRDEL  EQU   X'08'                   - DELETE
0000F  SACRACC  EQU   X'0F'                   - ACCESS
00080  SACROWN  EQU   X'80'                   - OWNER
0008F  SACRALL  EQU   X'8F'
00001  SACDCRE  EQU   X'01'          DEFINITION - CREATE
00002  SACDALT  EQU   X'02'                   - ALTER
```

```
00004  SACDDRO  EQU   X'04'                    - DROP
00008  SACDDIS  EQU   X'08'                    - DISPLAY
00010  SACDUSE  EQU   X'10'                    - USE
0001F  SACDDEF  EQU   X'1F'                    - DEFINE
00020  SACDREF  EQU   X'20'                    - REFERENCES
00080  SACDOWN  EQU   X'80'                    - OWNER
000BF  SACDALL  EQU   X'BF'
00001  SACASYS  EQU   X'01'         ADMINISTRATIVE - SYSADMIN
00002  SACADDA  EQU   X'02'                        - DDADMIN
00004  SACADBA  EQU   X'04'                        - DBADMIN
00008  SACADCA  EQU   X'08'                        - DCADMIN
00010  SACASIG  EQU   X'10'                        - SIGNON
00020  SACADBR  EQU   X'20'                        - DBAREAD
00040  SACADBW  EQU   X'40'                        - DBAWRITE
0007F  SACAALL  EQU   X'7F'
       *-------------------------------------------------------------------
```

Offset Value

```
       *  RESOURCE TYPE NUMBERS
       *-------------------------------------------------------------------
08000  SRTNUDEF EQU   32768        Number for user defined resource
       **                  NUMBERS 1-10 = SPECIAL
       **
00001  SRTNSPEC EQU   01            SPECIAL
00002  SRTNDCA  EQU   02           DC Admin
00003  SRTNSYSA EQU   03            System admin
0000A  SRTNLSP  EQU   10           Highest 'SPECIAL' resource
       **
       **                  NUMBERS 11-30 = Dictionary resources
0000B  SRTNDDA  EQU   11            DD Admin
0000C  SRTNNSUB EQU   12             Subschema
0000D  SRTNDAPP EQU   13             ADS application
0000E  SRTNDMSG EQU   14             Message
0000F  SRTNDIAL EQU   15           ADS Dialog
00010  SRTNDATT EQU   16            Attribute
00011  SRTNDCLA EQU   17            Class
00012  SRTNDELE EQU   18            Element
00013  SRTNDLOD EQU   19            Load module
00014  SRTNDMAP EQU   20             Map
00015  SRTNDMOD EQU   21             Module
00016  SRTNDPGM EQU   22             Program
00017  SRTNDREC EQU   23            Record
00018  SRTNDUDE EQU   24            User defined entity
00019  SRTNDFIL EQU   25           File
0001A  SRTNDPAN EQU   26             Panel
0001B  SRTNDSYS EQU   27            IDD system
0001C  SRTNDUSR EQU   28             IDD user
       **
       **                  These are SYSGEN definition restypes
```

```
0001D  SRTNDDES EQU   29              Destination
0001E  SRTNDLIN EQU   30              Line
0001F  SRTNDLTE EQU   31              Logical terminal
00020  SRTNDPTE EQU   32              Physical terminal
00021  SRTNDQUE EQU   33              Queue
00022  SRTNDTSK EQU   34              Task
00023  SRTNDACT EQU   35              COBOL syntax, Culprit over
00028  SRTNLDC  EQU   40              LAST DICTIONARY RESOURCE TYPE NO.
       **
       **                   Numbers 31-50 = Database resource types
0002A  SRTNDACC EQU   42              Access module
0002C  SRTNDMCL EQU   44              DMCL
0002D  SRTNTABL EQU   45              Table
0002E  SRTNAREA EQU   46              Area
0002F  SRTNDBTB EQU   47              Database name table
00030  SRTNQSCH EQU   48              SQL schema
00031  SRTNNSCH EQU   49              Non-SQL schema
00034  SRTNDB   EQU   52              Database
0003C  SRTNLCG  EQU   60              Last database resource type
       **
       **                   NUMBERS 51-70 = User resource types
0003D  SRTNUSER EQU   61              User
0003E  SRTNGROU EQU   62              Group
0003F  SRTNUPRF EQU   63              User profile
00046  SRTNLUS  EQU   70              Last user resource type no.
       **
       **                   Numbers 71-90 = System resource types
00047  SRTNSYST EQU   71              System
00048  SRTNCATE EQU   72              Category
00049  SRTNACTI EQU   73              Activity
0004A  SRTNTASK EQU   74              Task
0004B  SRTNQUEU EQU   75              Queue
0004C  SRTNSPGM EQU   76              Program
0004D  SRTNSLOD EQU   77              Load module (as loadable entity)
0004E  SRTNSACC EQU   78              Access module (run time)
```

Offset Value

```
0004F  SRTNNRU  EQU   79              Run unit
00050  SRTNSAPP EQU   80              Application
00051  SRTNSPRF EQU   81              System profile
00052  SRTNSGON EQU   82              SIGNON
0005A  SRTNLSY  EQU   90              Last system resource type no.
       *-----------------------------------------------------------------
       *  AUTHORIZATION ID TYPE
       *-----------------------------------------------------------------
000E4  SATUSER  EQU   C'U'            USER
000C7  SATGRP   EQU   C'G'            GROUP
       *-----------------------------------------------------------------
       *  AUTHORIZATION ID FLAGS
```

```
     *-------------------------------------------------------------------
000C9  SAFINAC  EQU   C'I'            INACTIVE
000C1  SAFACTV  EQU   C'A'            ACTIVE
000C4  SAFLDEL  EQU   C'D'            LOGICALLY DELETED
     *-------------------------------------------------------------------
     *  DEFAULT GROUP FLAG
     *-------------------------------------------------------------------
00040  SDGNDEFG EQU   C' '            NOT A DEFAULT GROUP INDICATOR
000C4  SDGDEFG  EQU   C'D'            DEFAULT GROUP INDICATOR
     *-------------------------------------------------------------------
     *  FUNCTION NUMBER ASSIGNMENT FLAG
     *-------------------------------------------------------------------
000E4  SFNUSER  EQU   C'U'            FUNCTION NOS. ASSIGNED BY USER
000E2  SFNSYS   EQU   C'S'            FUNCTION NOS. ASSIGNED BY SYSTEM
     *-------------------------------------------------------------------
     *  SECURITY METHOD
     *-------------------------------------------------------------------
000C9  SMEIDMS  EQU   C'I'            IDMS SECURITY
000C5  SMEEXT   EQU   C'E'            EXTERNAL SECURITY
000D6  SMEOFF   EQU   C'O'            NO SECURITY
     *-------------------------------------------------------------------
     *  SECURITY MODE
     *-------------------------------------------------------------------
000D6  SMOOFF   EQU   C'O'            SECURITY MODE IS OFF
000E6  SMOWARN  EQU   C'W'            SECURITY MODE IS WARN
000C5  SMOENF   EQU   C'E'            SECURITY MODE IS ENFORCE
     *-------------------------------------------------------------------
     *  MULTIPLE SIGNON
     *-------------------------------------------------------------------
000F0  SMSNO    EQU   C'0'            MULTIPLE SIGNON NO
000F1  SMSYES   EQU   C'1'            MULTIPLE SIGNON YES
     *-------------------------------------------------------------------
     *  WITH GRANT FLAG
     *-------------------------------------------------------------------
00001  SWGOPT   EQU   X'01'           WITH GRANT OPTION
     *-------------------------------------------------------------------
     *  GROUP/AUTH-ID LENGTH EQUATE
     *-------------------------------------------------------------------
00012  SGAUIDL  EQU   18
     *-------------------------------------------------------------------
     *  RESOURCE TYPE FLAG
     *-------------------------------------------------------------------
00040  SLRLIB   EQU   C' '            LIBRARY RESOURCE
000D5  SLRNLIB  EQU   C'N'            NON-LIBRARY RESOURCE
     *-------------------------------------------------------------------
     *  RUNNABLE ACCESS MODULE FLAG
     *-------------------------------------------------------------------
000E8  SRAMYES  EQU   C'Y'            ACCESS MODULE RUNNABLE
000D5  SRAMNO   EQU   C'N'            ACCESS MODULE NOT RUNNABLE
```

```
     *-----------------------------------------------------------------------
     *  GRANTABLE ACCESS MODULE FLAG
     *-----------------------------------------------------------------------
000E8  SGAMYES  EQU   C'Y'           ACCESS MODULE GRANTABLE
000D5  SGAMNO   EQU   C'N'           ACCESS MODULE NOT GRANTABLE
```

Offset Value

```
     *-----------------------------------------------------------------------
     *  RESOURCEGROUP TYPE FLAG
     *-----------------------------------------------------------------------
000C3  SRGTCAT  EQU   C'C'           CATEGORY
00040  SRGTNCAT EQU   C' '           NON-CATEGORY
     *-----------------------------------------------------------------------
     *  USERDATA TYPE FLAG
     *-----------------------------------------------------------------------
000C3  SUDTCAT  EQU   C'C'           CATEGORY
000D7  SUDTPRF  EQU   C'P'           PROFILE
     *-----------------------------------------------------------------------
     *  PROFILE TYPE FLAG
     *-----------------------------------------------------------------------
000E4  SPTUSER  EQU   C'U'           USER PROFILE
000E2  SPTSYS   EQU   C'S'           SYSTEM PROFILE
     *-----------------------------------------------------------------------
     *  ATTRIBUTE OVERRIDE FLAG
     *-----------------------------------------------------------------------
000E8  SAOYES   EQU   C'Y'           ATTRIBUTE MAY BE OVERRIDDEN
000D5  SAONO    EQU   C'N'           ATTRIBUTE MAY NOT BE OVERRIDDEN
```

#SECRB

```
            COPY  #SECRB
        *********************************************************************
        ***                                              ***
        ***     Security Request Block                               ***
        ***                                              ***
        ***     COPYRIGHT (C) 2007 CA technologies. ALL RIGHTS RESERVED.          ***
        ***                                              ***
        ***                                              ***
        ***                                              ***
        *********************************************************************
```

Offset Value

```
000000       SECRB    DSECT                                04/05/94
000000       SRBFUNC  DS    XL1              Function code
000001       SRBSCREL DS    XL1              SRB release level
        *
        SRBPINT  #FLAG X'80'             Processing is INTERNAL
000002       SRBPINTI DS    0XL1
     00080  SRBPINTM EQU   X'80'
        SRBPEXT  #FLAG X'40'             Processing is EXTERNAL
000002       SRBPEXTI DS    0XL1
     00040  SRBPEXTM EQU   X'40'
        SRBUSRR  #FLAG X'20'             Requester is in USER-MODE
000002       SRBUSRRI DS    0XL1
     00020  SRBUSRRM EQU   X'20'
        SRBSECR  #FLAG SRBPINTM+SRBPEXTM     Entity secured somehow
000002       SRBSECRI DS    0XL1
     000C0  SRBSECRM EQU   SRBPINTM+SRBPEXTM
000002       SRBPFLG  DS    XL1              Processing flag
        *
        SRBXFPR  #FLAG X'80'             Preprocessing exit in control
000003       SRBXFPRI DS    0XL1
     00080  SRBXFPRM EQU   X'80'
        SRBXFPO  #FLAG X'40'             Postprocessing exit in control
000003       SRBXFPOI DS    0XL1
     00040  SRBXFPOM EQU   X'40'
        SRBXFAB  #FLAG X'20'             Call aborted by exit
000003       SRBXFABI DS    0XL1
     00020  SRBXFABM EQU   X'20'
        SRBXFS3  #FLAG X'10'             Exit says skip EXIT3
000003       SRBXFS3I DS    0XL1
     00010  SRBXFS3M EQU   X'10'
        SRBXSGN  #FLAG X'08'             External signon call to exit 29
000003       SRBXSGNI DS    0XL1
     00008  SRBXSGNM EQU   X'08'
000003       SRBXFLG  DS    XL1              Exit flags
```

```
          *
000004      SRBERMSG DS    PL4            Error message id
000008      SRBERTXT DS    CL80           Full message text
000058            ORG   SRBERTXT        Redefine for use during SIGNON
000008      SRBCRYOP DS    XL8             Encrypted old password
000010      SRBCRYNP DS    XL8             Encrypted new password
000018            ORG
          *
000058      SRBXRTNC DS    0XL4           Return/reason codes
000058      SRBXR15  DS    XL2           R15 return code
     00004  SRBXURES EQU   4               Resource unknown or undefined
     00004  SRBXUUSR EQU   4               User id unknown
     00008  SRBXNACC EQU   8               Access denied
     0000C  SRBXINVP EQU   12              Invalid parm list
     00010  SRBXPWVF EQU   16               Password validation failure
          *
```

Offset Value

```
00005A      SRBXR0   DS    XL2            R0 reason code
     00004  SRBXDBE  EQU   4              Database access error
     00008  SRBXNSRT EQU   8               SRTT missing
     0000C  SRBXNMS  EQU   12              Multiple SIGNON not allowed
     00010  SRBXNNPW EQU   16               Password cannot be changed
00005C      SRBXINST DS    A            Pointer to feedback area
     00060  SRBLNG2  EQU   *-SECRB          Length of the fixed portion of SRB
     00018  SRBLNG2F EQU   ((SRBLNG2)+3)/4
000060      SRBOPP   DS    0F           Start of SRB extensions
          *-----------------------------------------------------------------------
          *
          *  SRB extension for SIGNON/SIGNOFF
          *
          *-----------------------------------------------------------------------
000060      SRBSGSON DS    A              Pointer to SIGNON element
000064      SRBSGLTF DS    A             Address of signed on LTE (XFR SGON)
000068      SRBSGLTT DS    A             Address of LTE signing on
00006C      SRBSGUSL DS    XL1           Length of user ID
00006D      SRBSGUSR DS    CL18            User ID
00007F      SRBSGGRL DS    XL1           Length of group
000080      SRBSGGRP DS    CL18           Group
000092      SRBSGPSL DS    XL1           Length of password
000093      SRBSGPSW DS    CL8             password
00009B      SRBSGNPL DS    XL1           Length of new password
00009C      SRBSGNPS DS    CL8            New password
0000A4      SRBSGACC DS    CL32            Accounting information
          *
          SRBSGPC  #FLAG X'80'         Skip password checking
0000C4      SRBSGPCI DS    0XL1
     00080  SRBSGPCM EQU   X'80'
          SRBSGPT  #FLAG X'40'         Password already encrypted
```

```
0000C4       SRBSGPTI DS    0XL1
      00040  SRBSGPTM EQU   X'40'
             SRBSGSM  #FLAG X'20'           Suppress SIGNOFF message
0000C4       SRBSGSMI DS    0XL1
      00020  SRBSGSMM EQU   X'20'
             SRBSGCP  #FLAG X'10'           Copy SIGNON from specified LTE
0000C4       SRBSGCPI DS    0XL1
      00010  SRBSGCPM EQU   X'10'
             SRBSGJU  #FLAG X'08'           Signon with user-id from jobcard
0000C4       SRBSGJUI DS    0XL1
      00008  SRBSGJUM EQU   X'08'
0000C4       SRBSGFG1 DS    XL1             FLAG 1
             *-----------------------------------------------------------------------
             *  The following fields are returned by SIGNON
             *-----------------------------------------------------------------------
0000C5       SRBSGSPR DS    CL18            System profile module name
0000D7       SRBSGUPR DS    CL18            User profile module name
      000E9  SRBSGLEN EQU   *-SECRB         Length of SRB for SIGNON/SIGNOFF
      0003B  SRBSGLNF EQU   (SRBSGLEN+3)/4
0000E9                ORG   SRBOPP      RESET
             *-----------------------------------------------------------------------
             *
             *  SRB extension for password encryption
             *
             *-----------------------------------------------------------------------
000060       SRBCRUSL DS    XL1             User ID length
000061       SRBCRUSR DS    CL18            User ID
000073       SRBCRPSL DS    XL1             Plaintext password length
000074       SRBCRPSW DS    CL8             Plaintext password
00007C       SRBCRYPS DS    CL8             Encrypted password
      00084  SRBCRLEN EQU   *-SECRB         Length of SRB for pswd encryption
      00021  SRBCRLNF EQU   (SRBCRLEN+3)/4
000084                ORG   SRBOPP      RESET
             *-----------------------------------------------------------------------
             *
```

Offset Value

```
             *  SRB extension for security check, bulk check and AM check
             *
             *-----------------------------------------------------------------------
000060       SRBSCSON DS    A               Pointer to SIGNON element
             SRBSCNL  #FLAG X'80'           LOG=NO requested
000064       SRBSCNLI DS    0XL1
      00080  SRBSCNLM EQU   X'80'
000064       SRBSCFLG DS    X               Option flags
000065                DS    XL3             Reserved
000068       SRBRSTTA DS    A               Pointer to SRTT entry
00006C       SRBACAHA DS    0A              Pointer to AM header
00006C       SRBSCRLA DS    A               Pointer to resource list
```

```
000070        SRBSCLCT DS    H              Entry count (if list request)
000072        SRBSCFNU DS    0H             Application function number
000072        SRBSCATG DS    H              Security category
000074        SRBSCAUT DS    XL6            Desired authorities (see #SECACAB)
00007A        SRBRSTYP DS    CL4            Resource type
00007E        SRBSCRNL DS    XL1            Resource name length
00007F        SRBSCRNM DS    CL32           Resource name
00009F        SRBSCDVL DS    XL1            Length of version or ddname
0000A0        SRBSCVER DS    0CL8           Version number (character)
0000A0        SRBSCDDN DS    CL8            DDname - programs only
0000A8        SRBSCDBL DS    XL1            Length of database name
0000A9        SRBSCDBN DS    CL8            Database name
0000B1        SRBSCSCL DS    XL1            Length of SQL schema name
0000B2        SRBSCSCM DS    CL18           SQL schema name
0000C4        SRBSCSSL DS    XL1            Length of subschema name
0000C5        SRBSCSSN DS    CL8
    000CD  SRBSCLEN EQU   *-SECRB           Length of SRB for SECHECK
    00034  SRBSCLNF EQU   (SRBSCLEN+3)/4
0000CD              ORG   SRBOPP    RESET
         *-----------------------------------------------------------------------
         *  ADD NEXT FUNCTION HERE
         *-----------------------------------------------------------------------
```

#SECRLST

```
        COPY  #SECRLST
***********************************************************************
***                                                   ***
***     SECURITY RESOURCE LIST                                ***
*** -------------------------------------------------------------- ***
*** Resource list entry.  Note that all resources in the list     ***
*** must be of the same class (as specified on the #SECHECK) and, ***
*** for internal checking, must be defined in the same            ***
*** dictionary or catalog.                              ***
***                                                   ***
***     COPYRIGHT (C) 2007 CA technologies. ALL RIGHTS RESERVED.            ***
***                                                   ***
***                                                   ***
***                                                   ***
***********************************************************************
```

Offset Value

```
000000        SECRLST  DSECT                                10/30/90
000000        SRLSCRNL DS    XL1              Length of resource name
000001        SRLSCRNM DS    CL60              Resource name
00003D        SRLSCLIB DS    XL1              Length of library name
00003E        SRLSCLIL DS    CL8              Library name
000046        SRLRAUTH DS    XL6              Requested authorities
       0004C  SRLRSLEN EQU   *-SECRLST          Length of resource id
00004C        SRLRTNMI DS    CL2              Minor code returned for entry
       0004E  SRLLNG   EQU   *-SECRLST          Total length of entry
```

#SECRTTD

```
                          COPY  #SECRTTD
***********************************************************************
*                                                                     *
*  The Resource Type Table (SRTT) is used to select external vs       *
*  internal processing for resource types and to provide external     *
*  resource classes for those resource types that are secured         *
*  externally.                                                        *
*                                                                     *
***********************************************************************
***********************************************************************
***                                                                 ***
***     SECRTTHD - Security Resource Type Table header              ***
***                                                                 ***
***     COPYRIGHT (C) 2010 CA. ALL RIGHTS RESERVED.                 ***
***                                                                 ***
***                                                                 ***
***                                                                 ***
***********************************************************************


Offset  Value

000000        SECRTTHD DSECT                        06/19/96
000000        SRTGRPLA DS    A          Address of group list
000004        SRTSGRTN DS    0F         SIGNON retention (before init)
000004        SRTSONTA DS    A          Address of SON table (after init)
000008        SRTCBMPA DS    A          Address of category bitmap pool
00000C        SRTABMPA DS    A          Address of activity bitmap pool
000010        SRTSCC   DS    0CL4       Startup completion code
000010        SRTSR15  DS    CL2        Return code (R15)
000012        SRTSR0   DS    CL2        Reason code (R0) if applicable
000014        SRTAPPIL DS    XL1        Length of system name
000015        SRTAPPID DS    CL8        System name
00001D        SRTENVNL DS    XL1        Length of environment name
00001E        SRTENVNM DS    CL8        Environment name
000026        SRTSVCNO DS    XL1        SVC number for local security
              SRTOINT  #FLAG X'01'      Internally secured resources
000027        SRTOINTI DS    0XL1
       00001  SRTOINTM EQU   X'01'
              SRTOEXT  #FLAG X'02'      Externally secured resources
000027        SRTOEXTI DS    0XL1
       00002  SRTOEXTM EQU   X'02'
              SRTDFSN  #FLAG X'04'      Default SIGNON active
000027        SRTDFSNI DS    0XL1
       00004  SRTDFSNM EQU   X'04'
              SRTOMXD  #FLAG SRTOINTM+SRTOEXTM   Mixed security if both on
000027        SRTOMXDI DS    0XL1
       00003  SRTOMXDM EQU   SRTOINTM+SRTOEXTM
              SRTESP   #FLAG X'10'      Process system profiles for ERUs
```

```
000027      SRTESPI  DS    0XL1
      00010 SRTESPM  EQU   X'10'
           SRTEUP   #FLAG X'20'           Process user profiles for ERUs
000027      SRTEUPI  DS    0XL1
      00020 SRTEUPM  EQU   X'20'
           SRTUCA   #FLAG X'40'           USER catalog available
000027      SRTUCAI  DS    0XL1
      00040 SRTUCAM  EQU   X'40'
           SRTSCA   #FLAG X'80'           SYSTEM catalog available
000027      SRTSCAI  DS    0XL1
      00080 SRTSCAM  EQU   X'80'
000027      SRTOPTNS DS    XL1               Option flags
000028      SRTTBH   DS    XL(TBHDSLEN)      Standard table header (see #TBHDS)
000034      SRTTBHN  DS    XL(TBHDSLEN)      Standard table header (see #TBHDS)
000040      SRTTLENT DS    F                 Total length of table
000044      SRTTOTAL DS    F                 Total entries
000048      SRTUSRTK DS    H                 Keyword token
00004A      SRTUSPRF DS    CL18              Default USER profile
00005C      SRTSYSTK DS    H                 Keyword token
00005E      SRTSYPRF DS    CL18              Default SYSTEM profile
000070      SRTERUTK DS    H                 Keyword token
000072      SRTERURF DS    CL18              Default USER profile for ERUs
000084      SRTERSTK DS    H                 Keyword token
000086      SRTERSRF DS    CL18              Default SYSTEM profile for ERUs
           *                                  R140
000098      SRTVERS  DS    XL1               SRTT version indicator     R140
      00000 SRTV12   EQU   X'00'             Up to release 12.01        R140
      00001 SRTV14   EQU   X'01'             Release 14.0 and above     R140
           *                                  R140
           SRTDUST  #FLAG X'01'           Default userid in SRTDUID field R140
000099      SRTDUSTI DS    0XL1
      00001 SRTDUSTM EQU   X'01'
           SRTDUVN  #FLAG X'02'           VTAM node name             R140
000099      SRTDUVNI DS    0XL1
      00002 SRTDUVNM EQU   X'02'
           SRTDUPT  #FLAG X'04'           PTERM-id                   R140
000099      SRTDUPTI DS    0XL1
      00004 SRTDUPTM EQU   X'04'
           SRTDULT  #FLAG X'08'           LTERM-id                   R140
000099      SRTDULTI DS    0XL1
      00008 SRTDULTM EQU   X'08'
000099      SRTDUFLG DS    XL1               Default userid flag        R140
00009A               DS    XL2           Reserved                   R140
00009C      SRTCBMHA DS    A                 Save A(category bitmap model)  R140
0000A0      SRTSCTOT DS    F                 Total calls to security    R140
0000A4      SRTSCFAI DS    F                 Failing security calls     R140
0000A8      SRT#SNON DS    F                 Total signon counter       R140
0000AC      SRTMSNON DS    F                 Multiple signon counter    R140
0000B0      SRTSONLK DS    F                 SON's lock counter         R140
```

```
0000B4     SRTDUID  DS    CL18          Default userid           R140
0000C6     SRTEUID  DS    CL18          Extract userid           R140
       *                                  R140
0000D8     SRTD#USR DS    F             Number of users signed on
0000DC     SRTD#USX DS    F             HWM of Number of users signed on
0000E0     SRTSTRTN DS    CL1           External security system ID    R171
   000E3  SRTSCATS EQU   C'T'           CA TopSecret             R171
   000C1  SRTSCAAC EQU   C'A'           CA-ACF2                  R171
   000E2  STRSRACF EQU   C'S'           RACF or SAF-compatible        R171
   00004  STRSNONE EQU   X'04'          No external security system    R171
0000E1             DS    XL3        Reserved                 R171
0000E4             DS    17F        Reserved                 R140
000128     SRTENT   DS    0F            Start of entries
   00128  SRTHLNG  EQU   *-SECRTTHD        Header length
       ***********************************************************************
       *                                   *
       *      SECRTTED - Security Resource Type Table Entry           *
       *                                   *
       ***********************************************************************
000000     SECRTTED DSECT
000000     SRTRSGPA DS    A             Pointer to owning resource group
000004     SRTSTYPA DS    A             Pointer to owning super class
000008     SRTEOSNA DS    A             Next in entity occurrence list
00000C     SRTCTABA DS    A             Pointer to category table
000010     SRTAMTAB DS    F             Access module table
000014     SRTABBRL DS    X             Length of internal resource type
000015     SRTABBR  DS    CL4           Internal resource type
000019     SRTNAMEL DS    X             Length of syntax/occurrence name
00001A     SRTNAME  DS    CL18          Syntax resource type
       SRTF1IN  #FLAG X'01'         Secured internally
00002C     SRTF1INI DS    0XL1
   00001  SRTF1INM EQU   X'01'
       SRTF1EX  #FLAG X'02'         Secured externally
00002C     SRTF1EXI DS    0XL1
   00002  SRTF1EXM EQU   X'02'
       SRTF1OF  #FLAG SRTF1EXM+SRTF1INM   Unsecured if both bits are off
00002C     SRTF1OFI DS    0XL1
   00003  SRTF1OFM EQU   SRTF1EXM+SRTF1INM
       SRTF1EF  #FLAG X'80'         External initialization failed
00002C     SRTF1EFI DS    0XL1
   00080  SRTF1EFM EQU   X'80'
00002C     SRTFLAG1 DS    XL1           Flag byte 1
00002D     SRTNBR   DS    0XL2          Number
00002D              DS    XL1
00002E     SRTNBRB  DS    XL1
00002F     SRTEFRTN DS    XL2           External init return code
000031     SRTCLASS DS    CL8           External resource class
000039     SRTINCOP DS    XL6           Internal name contruction tokens
00003F     SRTXNCOP DS    XL6           External name contruction tokens
```

```
000045       SRTSRBOF DS   AL2             Offset in SRB (generic only)
000047       SRTRUTYP DS   XL1             Run unit type index
     00048  SRTLNG   EQU   ((*-SECRTTED+3)/4)*4  Entry length
```

#SPRFDS

```
          COPY  #SPRFDS
***********************************************************************
***                                        ***
***    SPRF DSECT                                   ***
***                                        ***
***    PROFILE                                   ***
***                                        ***
***     COPYRIGHT (C) 2007 CA technologies. ALL RIGHTS RESERVED.          ***
***                                        ***
***                                        ***
***                                        ***
***********************************************************************
```

Offset Value

```
000000       SPRF     DSECT                              03/05/90
000000       SPRFNAME DS    CL18         PROFILE NAME
000012       SPRFTYPE DS    CL1          TYPE
          *                     * 'U' - USER PROFILE
          *                     * 'S' - SYSTEM PROFILE
000013       SPRFCTIM DS    CL8          TIME/DATE PROFILE WAS CREATED
00001B       SPRFUTIM DS    CL8          TIME/DATE PROFILE WAS LAST UPDATED
000023       SPRFCAID DS    CL18         CREATOR OF THE PROFILE
000035       SPRFUAID DS    CL18         LAST UPDATOR OF THE PROFILE
000047             DS    XL17        * UNUSED
     00058  SPRFDSLN EQU   *-SPRF       * LENGTH OF DSECT
          **********************************************************************
```

#SRESDS

```
         COPY  #SRESDS
     ***********************************************************************
     ***                                               ***
     ***    SRES DSECT                                          ***
     ***                                               ***
     ***    RESOURCE                                            ***
     ***                                               ***
     ***     COPYRIGHT (C) 2007 CA technologies. ALL RIGHTS RESERVED.          ***
     ***                                               ***
     ***********************************************************************
```

Offset Value

```
000000       SRES     DSECT
000000       SRESRTYP DS    CL4         RESOURCE TYPE
000004       SRESRNAM DS    CL60        RESOURCE NAME
000040       SRESGTNL DS    XL1         Null column indicator
000041       SRESGTYP DS    CL4         RESOURCE GROUP TYPE
000045       SRESGNNL DS    XL1         Null column indicator
000046       SRESGNAM DS    CL60        RESOURCE GROUP NAME
000082       SRESCFNO DS    H           CATEGORY OR ACTIVITY NUMBER
        *                   * CATEGORY NO IF SRESRTYP='CATA'
        *                   * ACTIVITY NO IF SRESRTYP='ACTI'
000084       SRESAMRN DS    CL1         RUNNABLE ACCESS MODULE FLAG
        *                   * 'N' IF NOT RUNNABLE
        *                   * 'Y' IF RUNNABLE
000085       SRESSTAT DS    CL1         STATUS FLAG
        *                   * 'A' FOR ACTIVE
        *                   * 'D' FOR LOGICALLY DELETED
000086       SRESAMGT DS    CL1         GRANTABLE ACCESS MODULE FLAG
        *                   * 'N' IF NOT GRANTABLE
        *                   * 'Y' IF GRANTABLE
000087       SRESAMLC DS    CL8         TIME/DATE OF LAST AM-RELATED CHANGE
00008F       SRESCTIM DS    CL8         TIME/DATE RESOURCE WAS CREATED
000097       SRESUTIM DS    CL8         TIME/DATE RESOURCE WAS LAST UPDATED
00009F       SRESCAID DS    CL18        CREATOR OF THE RESOURCE
0000B1       SRESUAID DS    CL18        LAST UPDATOR OF THE RESOURCE
0000C3                DS    XL21        * UNUSED
    000D8  SRESDSLN EQU   *-SRES      * LENGTH OF DSECT
        ***********************************************************************
```

#SRGADS

```
      COPY  #SRGADS
*************************************************************************
***                                                  ***
***    SRGA DSECT                                                ***
***                                                  ***
***    RESOURCE GROUP AUTHORITY                                  ***
***                                                  ***
***     COPYRIGHT (C) 2007 CA technologies. ALL RIGHTS RESERVED.             ***
***                                                  ***
***                                                  ***
***                                                  ***
*************************************************************************
```

Offset Value

```
000000       SRGA     DSECT                                    10/29/90
000000       SRGAAUID DS    CL18        AUTH. ID THE AUTHORITY IS GRANTED TO
000012       SRGARTYP DS    CL4         RESOURCE TYPE THE AUTHORITY IS GRANTED ON
000016       SRGARNAM DS    CL60        RESOURCE NAME THE AUTHORITY IS GRANTED ON
000052       SRGARA   DS    H           RUNTIME AUTHORITIES
            *                      *  1 FOR SELECT
            *                      *  1 FOR EXECUTE/RUN
            *                      *  2 FOR INSERT
            *                      *  4 FOR UPDATE
            *                      *  8 FOR DELETE
            *                      *128 FOR OWNER
000054       SRGARAWG DS    H           RUNTIME AUTHORITIES (WITH GRANT OPTION)
            *                      *  1 FOR SELECT (W/GRANT)
            *                      *  1 FOR EXECUTE/RUN (W/GRANT)
            *                      *  2 FOR INSERT (W/GRANT)
            *                      *  4 FOR UPDATE (W/GRANT)
            *                      *  8 FOR DELETE (W/GRANT)
            *                      *128 FOR OWNER (W/GRANT)
000056       SRGADA   DS    H           DEFINITION AUTHORITIES
            *                      *  1 FOR CREATE
            *                      *  2 FOR ALTER
            *                      *  4 FOR DROP
            *                      *  8 FOR DISPLAY
            *                      * 16 FOR USE
            *                      * 31 FOR DEFINE
            *                      * 32 FOR REFERENCE
            *                      *128 FOR OWNER
000058       SRGADAWG DS    H           DEFINITION AUTHORITIES (WITH GRANT OPTION)
            *                      *  1 FOR CREATE (W/GRANT)
            *                      *  2 FOR ALTER (W/GRANT)
            *                      *  4 FOR DROP (W/GRANT)
            *                      *  8 FOR DISPLAY (W/GRANT
        *                    * 16 FOR USE (W/GRANT)
        *                    * 31 FOR DEFINE (W/GRANT)
        *                    * 32 FOR REFERENCE (W/GRANT)
        *                    *128 FOR OWNER (W/GRANT)
00005A      SRGAOA   DS    H          OTHER AUTHORITIES
        *                    *  1 FOR SYSADMIN
        *                    *  2 FOR DDADMIN
        *                    *  4 FOR DBADMIN
        *                    *  8 FOR DCADMIN
        *                    * 16 FOR SIGNON
        *                    * 32 FOR DBAREAD
        *                    * 64 FOR DBAWRITE
00005C      SRGAFUNC DS    XL32       FUNCTION BIT MAP
00007C      SRGACTIM DS    CL8        TIME/DATE GROUP AUTH. WAS CREATED
000084      SRGAUTIM DS    CL8        TIME/DATE GROUP AUTH. WAS LAST UPDATED
00008C      SRGACAID DS    CL18       CREATOR OF GROUP AUTHORITY
```

Offset Value

```
00009E      SRGAUAID DS    CL18       LAST UPDATOR OF GROUP AUTHORITY
0000B0               DS    XL16       * UNUSED
    000C0  SRGADSLN EQU   *-SRGA      * LENGTH OF DSECT
        ********************************************************************
```

#SRGPDS

```
          COPY  #SRGPDS
      ***********************************************************************
      ***                                            ***
      ***    SRGP DSECT                                         ***
      ***                                            ***
      ***    RESOURCE GROUP                                           ***
      ***                                            ***
      ***     COPYRIGHT (C) 2007 CA technologies. ALL RIGHTS RESERVED.           ***
      ***                                            ***
      ***                                            ***
      ***                                            ***
      ***********************************************************************
```

Offset Value

```
000000        SRGP     DSECT                                    10/29/90
000000        SRGPRTYP DS    CL4          RESOURCE TYPE
000004        SRGPRNAM DS    CL60         RESOURCE NAME
000040        SRGPOWNR DS    CL18         AUTH. ID OF THE RES. GROUP OWNER
000052        SRGPCATN DS    H            CATEGORY NUMBER
000054        SRGPFASG DS    CL1          FUNCTION NUMBER ASSIGNMENT FLAG
         *                    * 'S' FOR ASSIGNED BY SYSTEM
         *                    * 'U' FOR ASSIGNED BY USER
000055        SRGPCTIM DS    CL8          TIME/DATE AUTH. ID WAS CREATED
00005D        SRGPUTIM DS    CL8          TIME/DATE AUTH. ID WAS LAST UPDATED
000065        SRGPCAID DS    CL18         CREATOR OF THE RESOURCE GROUP
000077        SRGPUAID DS    CL18         LAST UPDATOR OF THE RESOURCE GROUP
000089                 DS    XL19         * UNUSED
     0009C  SRGPDSLN EQU   *-SRGP       * LENGTH OF DSECT
         **********************************************************************
```

#SROPDS

```
      COPY  #SROPDS
******************************************************************
***                                       ***
***    SROP DSECT                                  ***
***                                       ***
***    RESOURCE OPTION                                  ***
***                                       ***
***     COPYRIGHT (C) 2007 CA technologies. ALL RIGHTS RESERVED.          ***
***                                       ***
***                                       ***
***                                       ***
******************************************************************
```

Offset Value

```
000000      SROP     DSECT                         10/29/90
000000      SROPRTYP DS    CL4       RESOURCE TYPE
000004      SROPRNAM DS    CL60       RESOURCE NAME
000040      SROPSMTH DS    CL1       SECURITY METHOD
         *                * 'I' FOR IDMS
         *                * 'E' FOR EXTERNAL
         *                * 'O' FOR OFF
000041            DS    XL1        * unused
000042      SROPMULT DS    CL1       MULTIPLE SIGNON FLAG
         *                * '0' FOR MULTIPLE SIGNON NOT ALLOWED
         *                * '1' FOR MULTIPLE SIGNON ALLOWED
000043      SROPCTIM DS    CL8       TIME/DATE RESOURCE OPTION WAS CREATED
00004B      SROPUTIM DS    CL8       TIME/DATE RES. OPT. WAS LAST UPDATED
000053      SROPCAID DS    CL18      CREATOR OF THE RESOURCE OPTION
000065      SROPUAID DS    CL18      LAST UPDATOR OF THE RESOURCE OPTION
000077            DS    XL17       * UNUSED
     00088  SROPDSLN EQU   *-SROP     * LENGTH OF DSECT
         *****************************************************************
```

#SRSADS

```
          COPY  #SRSADS
**********************************************************************
***                                               ***
***    SRSA DSECT                                            ***
***                                               ***
***    RESOURCE AUTHORITY                                              ***
***                                               ***
***     COPYRIGHT (C) 2007 CA technologies. ALL RIGHTS RESERVED.              ***
***                                               ***
***                                               ***
***                                               ***
**********************************************************************
```

Offset Value

```
000000       SRSA     DSECT                                     10/29/90
000000       SRSAAID  DS    CL18        AUTH. ID THE AUTHORITY IS GRANTED TO
000012       SRSARTYP DS    CL4         RESOURCE TYPE THE AUTHORITY IS GRANTED ON
000016       SRSARNAM DS    CL60        RESOURCE NAME THE AUTHORITY IS GRANTED ON
000052       SRSARA   DS    H           RUNTIME AUTHORITIES
           *                   *  1 FOR SELECT
           *                   *  1 FOR EXECUTE/RUN
           *                   *  2 FOR INSERT
           *                   *  4 FOR UPDATE
           *                   *  8 FOR DELETE
           *                   *128 FOR OWNER
000054       SRSARAWG DS    H           RUNTIME AUTHORITIES (WITH GRANT OPTION)
           *                   *  1 FOR SELECT WITH GRANT
           *                   *  1 FOR EXECUTE WITH GRANT
           *                   *  2 FOR INSERT WITH GRANT
           *                   *  4 FOR UPDATE WITH GRANT
           *                   *  8 FOR DELETE WITH GRANT
           *                   *128 FOR OWNER WITH GRANT
000056       SRSADA   DS    H           DEFINITION AUTHORITIES
           *                   *  1 FOR CREATE
           *                   *  2 FOR ALTER
           *                   *  4 FOR DROP
           *                   *  8 FOR DISPLAY
           *                   * 16 FOR USE
           *                   * 31 FOR DEFINE
           *                   * 32 FOR REFERENCE
           *                   *128 FOR OWNER
000058       SRSADAWG DS    H           DEFINITION AUTHORITIES (WITH GRANT OPTION)
           *                   *  1 FOR CREATE WITH GRANT
           *                   *  2 FOR ALTER WITH GRANT
           *                   *  4 FOR DROP WITH GRANT
           *                   *  8 FOR DISPLAY WITH GRANT
```

```
      *            * 16 FOR USE WITH GRANT
      *            * 31 FOR DEFINE WITH GRANT
      *            * 32 FOR REFERENCE WITH GRANT
      *            *128 FOR OWNER WITH GRANT
00005A    SRSAOA  DS   H      OTHER AUTHORITIES
      *            *  1 FOR SYSADMIN
      *            *  2 FOR DDADMIN
      *            *  4 FOR DBADMIN
      *            *  8 FOR DCADMIN
      *            * 16 FOR SIGNON
      *            * 32 FOR DBAREAD
      *            * 64 FOR DBAWRITE
00005C    SRSACTIM DS   CL8     TIME/DATE RESOURCE AUTH. WAS CREATED
000064    SRSAUTIM DS   CL8     TIME/DATE RESOURCE AUTH. LAST UPDATED
00006C    SRSACAID DS   CL18    CREATOR OF THE RESOURCE AUTHORITY
00007E    SRSAUAID DS   CL18    LAST UPDATOR OF THE RESOURCE AUTHORITY
```

Offset Value

```
000090           DS   XL16     * UNUSED
   000A0  SRSADSLN EQU   *-SRSA     * LENGTH OF DSECT
      ***********************************************************************
```

#SUGPDS

```
      COPY  #SUGPDS
**********************************************************************
***                                              ***
***    SUGP DSECT                                            ***
***                                              ***
***    USERGROUP                                              ***
***                                              ***
***     COPYRIGHT (C) 2007 CA technologies. ALL RIGHTS RESERVED.          ***
***                                              ***
***                                              ***
***                                              ***
**********************************************************************
```

Offset Value

```
000000       SUGP     DSECT                              05/16/89
000000       SUGPGPID DS    CL18       AUTH. ID OF THE PARENT GROUP
000012       SUGPUSID DS    CL18       AUTH. ID OF THE CHILD USER
000024       SUGPDEFG DS    CL1        DEFAULT GROUP INDICATOR
           *                    * ' ' FOR NOT THE DEFAULT GROUP
           *                    * 'D' FOR THE DEFAULT GROUP
000025       SUGPCTIM DS    CL8        TIME/DATE GROUP WAS CREATED
00002D       SUGPUTIM DS    CL8        TIME/DATE GROUP WAS LAST UPDATED
000035       SUGPCAID DS    CL18       CREATOR OF THE GROUP
000047       SUGPUAID DS    CL18       LAST UPDATOR OF THE GROUP
000059            DS    XL11       * UNUSED
     00064  SUGPDSLN EQU   *-SUGP      * LENGTH OF DSECT
           **********************************************************************
```

#SUSDDS

```
      COPY  #SUSDDS
************************************************************************
***                                            ***
***    SUSD DSECT                                         ***
***                                            ***
***    USER DATA                                         ***
***                                            ***
***     COPYRIGHT (C) 2007 CA technologies. ALL RIGHTS RESERVED.            ***
***                                            ***
***                                            ***
***                                            ***
************************************************************************
```

Offset Value

```
000000      SUSD     DSECT                                  10/29/90
000000      SUSDAUID DS    CL18       AUTHORIZATION ID
000012      SUSDCLTH DS    H          CATEGORY BIT MAP LENGTH
000014      SUSDTYPE DS    CL1        TYPE OF USER DATA
         *                    * 'C' - CATEGORY
         *                    * 'P' - PROFILE
         *                   PROFILE DATA
000015      SUSDSYST DS    CL8        SYSTEM NAME
00001D      SUSDPROF DS    CL18       PROFILE NAME
         *                   COMMON DATA
00002F      SUSDCTIM DS    CL8        TIME/DATE AUTH. CAT WAS CREATED
000037      SUSDUTIM DS    CL8        TIME/DATE AUTH. CAT WAS LAST UPDATED
00003F      SUSDCAID DS    CL18       CREATOR OF THE AUTHORIZATION CATEGORY
000051      SUSDUAID DS    CL18       LAST UPDATOR OF THE AUTHORIZATION CATEGORY
         *                   CATEGORY DATA
000063            DS    XL5        * UNUSED
000068      SUSDCATG DS    CL4096     CATEGORY BIT MAP
         *                     EACH BIT COORESPONDS TO THE CATEGORY
         *                     THE AUTH. ID HAS BEEN GRANTED ACCESS TO
     01068  SUSDDSLN EQU   *-SUSD     * LENGTH OF DSECT
         ***********************************************************************
```

#SUSRDS

```
          COPY  #SUSRDS
      ************************************************************************
      ***                                              ***
      ***    SUSR DSECT                                           ***
      ***                                              ***
      ***    USER                                              ***
      ***                                              ***
      ***     COPYRIGHT (C) 2007 CA technologies. ALL RIGHTS RESERVED.              ***
      ***                                              ***
      ***                                              ***
      ***                                              ***
      ************************************************************************
```

Offset Value

```
000000       SUSR     DSECT                                   10/29/90
000000       SUSRAUID DS    CL18        AUTHORIZATION ID
000012       SUSRTYPE DS    CL1         ID TYPE
      *                     * 'U' FOR USER
      *                     * 'G' FOR GROUP
000013       SUSRSTAT DS    CL1         CURRENT STATUS
      *                     * 'A' FOR ACTIVE
      *                     * 'I' FOR INACTIVE
      *                     * 'D' FOR LOGICALLY DELETED
000014       SUSRPSWD DS    CL8         PASSWORD
      *                     * X'00' FOR NULL
00001C       SUSRNAME DS    CL32        USER'S FULL NAME
      *                      (APPLICABLE ONLY WHEN SUSRTYPE='U')
00003C       SUSRDESC DS    CL40        DESCRIPTION
000064             DS    H          * unused
000066       SUSRPROF DS    CL18        SECURITY DOMAIN PROFILE
000078             DS    CL4         * unused
00007C       SUSRSTIM DS    CL8         TIME/DATE AUTH. ID LAST SIGNED ON
000084       SUSRGTIM DS    CL8         TIME/DATE USER LAST REMOVED FROM GROUP
00008C       SUSRPTIM DS    CL8         TIME/DATE PASSWORD LAST CHANGED
000094       SUSRCTIM DS    CL8         TIME/DATE AUTH. ID WAS CREATED
00009C       SUSRUTIM DS    CL8         TIME/DATE AUTH. ID WAS LAST UPDATED
0000A4       SUSRCAID DS    CL18        CREATOR OF THE AUTH. ID.
0000B6       SUSRUAID DS    CL18        LAST UPDATOR OF THE AUTH. ID
0000C8             DS    XL16        * UNUSED
   000D8  SUSRDSLN EQU   *-SUSR      * LENGTH OF DSECT
      ***********************************************************************
```

Appendix C: Privileges Required for Statements

This section contains the following topics:

DDL Statements

DISPLAY Statements

To execute a DISPLAY statement on a physical database entity requires one of these privileges if the dictionary and the resource type of the entity has been secured:

- DISPLAY privilege on the entity
- DBADMIN privilege on the dictionary where the entity is defined

This table shows what privilege is required to execute a DISPLAY statement when the indicated resource type has been secured:

Statement	Privilege required	Resource type	Resource name
DISPLAY SEGMENT	DISPLAY	DB	*segment-name*
DISPLAY FILE	DISPLAY	DB	*segment-name*
DISPLAY AREA	DISPLAY	DB	*segment-name*
DISPLAY DMCL	DISPLAY	DMCL	*dmcl-name*
DISPLAY[JOURNAL] BUFFER	DISPLAY	DMCL	*dmcl-name*
DISPLAY ARCHIVE/DISK/TAPE JOURNAL	DISPLAY	DMCL	*dmcl-name*
DISPLAY DBTABLE	DISPLAY	DBTB	*dbtable-name*
DISPLAY DBNAME	DISPLAY	DBTB	*dbtable-name*

Definition Statements

To execute the physical database definition statements in the following table requires:

- DBADMIN privilege on the dictionary where the physical database definition is stored, if that dictionary has been secured.
- The privileges indicated in the following table, if the resource types have been secured.

The privileges must be granted in a session that is connected to the system dictionary.

Statement	**Privilege required**	**Resource type**	**Resource name**
CREATE SEGMENT	CREATE	DB	*segment-name*
ALTER SEGMENT	ALTER	DB	*segment-name*
DROP SEGMENT	DROP	DB	*segment-name*
CREATE FILE	ALTER	DB	*segment-name*
ALTER FILE	ALTER	DB	*segment-name*
DROP FILE	ALTER	DB	*segment-name*
CREATE AREA	ALTER	DB	*segment-name*
ALTER AREA	ALTER	DB	*segment-name*
DROP AREA	ALTER	DB	*segment-name*
CREATE DMCL	CREATE	DMCL	*dmcl-name*
ALTER DMCL... DBTABLE *name*	ALTER USE	DMCL DBTB	*dmcl-name* *dbtable-name*
DROP DMCL	DROP	DMCL	*dmcl-name*
GENERATE DMCL	ALTER	DMCL	*dmcl-name*
CREATE [JOURNAL] BUFFER	ALTER	DMCL	*dmcl-name*
ALTER [JOURNAL] BUFFER	ALTER	DMCL	*dmcl-name*
DROP [JOURNAL] BUFFER	ALTER	DMCL	*dmcl-name*
CREATE ARCHIVE/DISK/TAPE JOURNAL	ALTER	DMCL	*dmcl-name*
ALTER ARCHIVE/DISK/TAPE JOURNAL	ALTER	DMCL	*dmcl-name*
DROP ARCHIVE/DISK/TAPE JOURNAL	ALTER	DMCL	*dmcl-name*

Statement	Privilege required	Resource type	Resource name
CREATE DBTABLE	CREATE	DBTB	*dbtable-name*
ALTER DBTABLE	ALTER	DBTB	*dbtable-name*
DROP DBTABLE	DROP	DBTB	*dbtable-name*
GENERATE DBTABLE	ALTER	DBTB	*dbtable-name*
CREATE DBNAME	CREATE ALTER	DB DBTB	*database-name* *dbtable-name*
ALTER DBNAME	ALTER ALTER	DB DBTB	*database-name* *dbtable-name*
DROP DBNAME	DROP ALTER	DB DBTB	*database-name* *dbtable-name*

Utilities

To execute CA IDMS utilities, a user requires the privilege indicated in the following table if the resource type to which the privilege applies has been secured.

Note: The AREA NRU, and TABL resource types are secured when the DB resource type is secured.

Statement	Privilege Required	Resource Type	Resource Name
ARCHIVE JOURNAL	USE	DMCL	*dmcl-name*
ARCHIVE LOG	DBAWRITE	AREA	SYSTEM.DDLDCLOG
BACKUP AREA	DBAREAD	AREA	*segment-name.area-name*
BACKUP FILE	DBAREAD	AREA	*segment-name.area-name* (all areas in file)
BACKUP SEGMENT	DBAREAD	AREA	*segment-name.area-name* (all areas in segment)
BUILD	INSERT	TABL	*schema-name.table-identifier*
CLEANUP SEGMENT	DBAWRITE	AREA	*segment-name.area-name* (all areas in segment)
CLEANUP SEGMENT ... AREA	DBAWRITE	AREA	*segment-name.area-name*

Statement	Privilege Required	Resource Type	Resource Name
EXPAND PAGE	DBAWRITE	AREA	*segment-name.area-name* (all areas in file)
FASTLOAD	DBAWRITE	AREA	*segment-name.area-name* (all areas in load)
FIX ARCHIVE	USE	DMCL	*dmcl-name*
FIX PAGE	DBAWRITE	AREA	*segment-name.area-name*
FORMAT AREA	DBAWRITE	AREA	*segment-name.area-name*
FORMAT FILE	DBAWRITE	AREA	*segment-name.area-name* (all areas in file)
FORMAT JOURNAL	USE	DMCL	*dmcl-name*
FORMAT SEGMENT	DBAWRITE	AREA	*segment-name.area-name* (all areas in segment)
INSTALL STAMPS	DBAWRITE	AREA	*segment-name.area-name*
LOAD	INSERT	TABL	*schema-name.table-identifier*
MAINTAIN INDEX	DBAWRITE	AREA	*segment-name.area-name* (affected areas in segment)
PRINT INDEX	DBAREAD	AREA	*segment-name.area-name* (index and data areas)
PRINT JOURNAL	USE	DMCL	*dmcl-name*
PRINT LOG	DBAREAD	AREA	SYSTEM.DDLDCLOG
PRINT PAGE	DBAREAD	AREA	*segment-name.area-name*
PRINT SPACE FOR AREA	DBAREAD	AREA	*segment-name.area-name*
PRINT SPACE FOR FILE	DBAREAD	AREA	*segment name.area-name* (all areas in file)
PRINT SPACE FOR SEGMENT	DBAREAD	AREA	*segment-name.area-name* (all areas in segment)
PUNCH DBTABLE	USE	DBTB	*dbtable-name*
PUNCH DMCL	USE	DMCL	*dmcl-name*
RELOAD	DBAWRITE	AREA	*segment-name.area-name* (all areas in load)

Statement	Privilege Required	Resource Type	Resource Name
REORG	DBAREAD DBAWRITE	AREA	*segment-name.area-name* (all selected areas and areas with set connections to the records in the selected areas or indexes on the records in the selected areas)
RESTORE AREA	DBAWRITE	AREA	*segment-name.area-name*
RESTORE FILE	DBAWRITE	AREA	*segment-name.area-name* (all areas in file)
RESTORE SEGMENT	DBAWRITE	AREA	*segment-name.area-name* (all areas in segment)
RESTRUCTURE CONNECT	DBAWRITE	AREA	*segment-name.area-name* (all selected areas)
RESTRUCTURE	DBAWRITE	AREA	*segment-name.area-name* (all selected areas)
ROLLBACK AREA	DBAWRITE	AREA	*segment-name.area-name*
ROLLBACK DMCL	DBAWRITE	AREA	*segment-name.area-name* (affected areas in DMCL)
ROLLBACK FILE	DBAWRITE	AREA	*segment-name.area-name* (affected areas in file)
ROLLBACK SEGMENT	DBAWRITE	AREA	*segment-name.area-name* (affected areas in segment)
ROLLFORWARD AREA	DBAWRITE	AREA	*segment-name.area-name*
ROLLFORWARD DMCL	DBAWRITE	AREA	*segment-name.area-name* (affected areas in DMCL)
ROLLFORWARD FILE	DBAWRITE	AREA	*segment-name.area-name* (affected areas in file)
ROLLFORWARD SEGMENT	DBAWRITE	AREA	*segment-name.area-name* (affected areas in segment)
UNLOAD	DBAREAD	AREA	*segment-name.area-name* (all selected areas and areas with set connections to the records in the selected areas or indexes on the records in the selected areas)
UNLOCK AREA	DBAWRITE	AREA	*segment-name.area-name*
UNLOCK SEGMENT	DBAWRITE	AREA	*segment-name.area-name* (all areas in segment)

Statement	Privilege Required	Resource Type	Resource Name
UPDATE STATISTICS FOR AREA	DBAREAD	AREA	*segment-name*.*area-name*
UPDATE STATISTICS FOR SCHEMA	ALTER	SCHEMA	*schema-name*
UPDATE STATISTICS FOR TABLE	ALTER	TABL	*schema-name*.*table-identifier*
VALIDATE	SELECT	TABL	*schema-name*.*table-identifier* (all selected tables)
IDMSDBAN	DBAREAD	AREA	*segment-name*.*area-name*
IDMSRPTS	EXECUTE	NRU	*dictionary-name*. IDMSNWKG.IDMSRPTS Additional privileges may be required depending on the report requested.

Note: For more information, see GRANT Physical Database Definition Privileges (see page 305).

SQL Statements

Statement Categories

If the database resource has been secured, users require appropriate privileges to execute SQL statements in these three categories:

- SQL DDL
- SQL DML
- Access module management

SQL Session Authorization

A user does not have the authority to issue a CONNECT statement to a dictionary under the central version unless the user holds signon privilege for the system in which the dictionary is defined.

SQL DDL statements

To execute CA IDMS SQL DDL statements, a user requires the privilege indicated in the following table if the resource type to which the privilege applies has been secured.

Note: Resource types QSCH, TABL, and AREA are secured when the DB resource type is secured.

Statement	**Privilege required**	**Resource type**	**Resource name**
CREATE CALC	ALTER(2)	TABL	*table-identifier*
DROP CALC	ALTER(2)	TABL	*table-identifier*
CREATE CONSTRAINT			
	ALTER(2)	TABL	*referencing-table-identifier*
	REFERENCES	TABL	*referenced-table-identifier*
DROP CONSTRAINT	ALTER(2)	TABL	*referencing-table- identifier*
CREATE FUNCTION	CREATE(2)	TABL	*function-identifier*
ALTER FUNCTION	ALTER(2)	TABL	*function-identifier*
DROP FUNCTION	DROP(2)	TABL	*function-identifier*
CREATE INDEX			
	ALTER(2)	TABL	*table-identifier*
	USE	AREA	*segment-name.area-name*
ALTER INDEX	ALTER(2)	TABL	*table-identifier*
DROP INDEX	ALTER(2)	TABL	*table-identifier*
CREATE PROCEDURE	CREATE(2)	TABL	*procedure-identifier*
ALTER PROCEDURE	ALTER(2)	TABL	*procedure-identifier*
DROP PROCEDURE	DROP(2)	TABL	*procedure-identifier*
CREATE SCHEMA(1)	CREATE	QSCH	*schema-name*
ALTER SCHEMA(1)	ALTER	QSCH	*schema-name*
DROP SCHEMA [CASCADE]	DROP	QSCH	*schema-name*
CREATE TABLE			
	CREATE(2)	TABL	*table-identifier*
	USE	AREA	*segment-name.area-name*

Statement	Privilege required	Resource type	Resource name
ALTER TABLE	ALTER(2)	TABL	*table-identifier*
DROP TABLE [CASCADE]	DROP(2)	TABL	*table-identifier*
CREATE TABLE PROCEDURE	CREATE(2)	TABL	*table-procedure-identifier*
ALTER TABLE PROCEDURE	ALTER(2)	TABL	*table-procedure-identifier*
DROP TABLE PROCEDURE	DROP(2)	TABL	*drop-procedure-identifier*
CREATE VIEW	CREATE(2)	TABL	*view-identifier*
DROP VIEW [CASCADE]	DROP(2)	TABL	*view-identifier*

(1) If reference to a non-SQL-defined schema is made in the CREATE/ALTER SCHEMA statement, then the user must also hold either USE on NONSQL SCHEMA V*nnnn.schema-name* or DBADMIN on the dictionary where the non-SQL-defined schema is stored. If DBNAME is specified, the user must hold USE on the named database; if not specified, the user must hold DBADMIN on the system dictionary.

(2) The owner of the associated schema implicitly holds the privilege.

SQL DML Statements

Dynamic SQL Statements

To execute an SQL DML statement dynamically (for example, through the Command Facility), a user requires the access privilege indicated in the following table or must own schema associated with the table-like object, if the database has been secured.

If a DELETE, UPDATE, or INSERT statement contains a query expression or subquery, additional privileges are required for the table-like objects referred to by the query expression or subquery. If a SELECT statement explicitly names a view, additional privileges are required.

Note: For complete documentation of privileges required to execute SQL DML statements, see the *CA IDMS SQL Reference Guide.*

Embedded SQL Statements

If the SQL DML statement is embedded in an application program and the database accessed is secured externally, the executing user must hold the applicable privileges on all tables accessed by the statement, whether directly or indirectly through a view.

If the SQL DML statement is embedded in an application program and the database accessed is secured internally, the executing user requires only EXECUTE privilege on the access module; the user inherits the necessary access privileges from the grantor of EXECUTE privilege for the purpose of executing the access module.

Statement	Privilege required	Resource type	Resource name
DELETE	DELETE	TABL	*schema-name*.**table-name**
INSERT	INSERT	TABL	*schema-name*.**table-name**
SELECT	SELECT	TABL	*schema-name*.**table-name**
UPDATE	UPDATE	TABL	*schema-name*.**table-name**

Access Module Management Statements

To execute access module management statements a user requires the privilege indicated in the following table or must own the schema associated with the access module, if the database has been secured.

If the owner lacks one or more applicable table access privileges for SQL statements in the RCMs included in the access module when it is created or altered, a warning is issued. If privileges are lacking at runtime, an error occurs when the user attempts to execute the access module.

Statement	Privilege required	Resource type	Resource name
CREATE ACCESS MODULE	CREATE	DACC	*schema-name.access- module-name*
ALTER ACCESS MODULE	ALTER	DACC	*schema-name.access- module-name*
DROP ACCESS MODULE	DROP	DACC	*schema-name.access- module-name*
EXPLAIN ACCESS MODULE	DISPLAY	DACC	*schema-name.access- module-name*

Appendix D: User-Defined System Security Rules

This section contains the following topics:

User Exits

What You Can Do

You can establish site-specific security by using predefined user exits. A user exit allows you to extend CA IDMS software by calling a user-written Assembler routine each time a particular point in CA IDMS system execution is reached.

For example, you can use exit 28 (the security preprocessing exit) to capture and evaluate signon information whenever a user requests signon to the runtime system but before signon processing is initiated.

Exit 14, BIND RUN-UNIT and READY AREA

Description

Exit 14, the BIND RUN-UNIT and READY AREA exit, is invoked before a BIND RUN-UNIT or READY AREA is performed. This exit is invoked *after* exit 23, the pre-BIND RUN-UNIT exit, discussed as follows.

How to Use This Exit for Security Purposes

You can use exit 14 to do the following:

- Check a user's access to a given DBNAME.
- Check a user's access to a given area.
- Reject the run unit's signon to the CA IDMS system.

Exit 14 can be used to obtain information from the ERE SVC extension.

Exit 22, Report Security and Routing

Description

Exit 22, the report security and routing exit, is invoked after RHDCPRNT has assigned a unique report identifier to a new print request.

How to Use This Exit for Security Purposes

You can use exit 22 to specify security or routing information for a report. For example, you can obtain signon information about the user who is requesting a report. Then you can use the information to decide the report destination.

To use exit 22 for this purpose, do the following:

- Write a queue record that contains routing information.
- Modify the RHDCBANR routine so that it is included in the report header

Exit 23, Pre-BIND RUN-UNIT

Description

Exit 23, the pre-BIND RUN-UNIT exit, is invoked before a BIND RUN-UNIT is performed. This exit is invoked *before* exit 14.

How to Use This Exit for Security Purposes

You can use exit 23 to override the subschema name, database name, database node, dictionary name, and dictionary node specified in the program's BIND RUN-UNIT statement. If you replace one of these parameters with an invalid value, the BIND request will fail.

For example, you can choose to change the subschema name when a program specifies a subschema that it should not be accessing.

Exit 27, ERE Extension Examiner

Description

Exit 27, the ERE extension exit, is invoked after the DC/UCF system receives a new external request for services, but before the system performs any processing for the request.

Exit 27 can also be called on a DDS source node before a request for services is sent to a target node. In this case, the exit can be used to place information in the ERE extension.

How to Use This Exit for Security Purposes

You can use exit 27 to place information in the ERE extension from a DDS source node. For example, you can save site-specific security information (such as a user ID).

You can also use exit 27 to examine information in the ERE extension before CA IDMS processes a new external request for services. This information could have been placed in the ERE extension by another exit 27 routine on a DDS source node requesting the services.

Exit 28, Security Preprocessing Exit

Description

Exit 28, the security preprocessing exit, allows you to examine all security requests, including user signon and signoff, before they are processed by the security system.

Exit 28 is called after the security system has validated the function code but before it performs any other processing for the request. By setting a flag (SRBXFAB) in the Security Request Block (SRB), the exit can request that access be denied.

These parameters are passed:

- The address of the SRB
- The length of the SRB

How to Use This Exit for Security Purposes

You can use exit 28 to examine information in the SRB before CA IDMS central security processes the security request. For example, you might alter the required authorities based on site-specific security enforcement requirements.

Considerations

Exit 28 cannot force the security system to allow the requested access. If the exit does not abort the request by setting SRBXFAB, the security system processes the request normally.

You must write this exit routine to execute in SYSTEM MODE. The #DEFXIT macro, which adds the exit routine to the system, must perform the following:

- Specify MODE=SYSTEM
- Call the routine using either DC or IBM calling conventions
- Call the routine by entry point

Exit 29, Security Postprocessing Exit

Description

Exit 29, the security postprocessing exit, allows you to examine all security requests, including user signon and signoff, after they are processed by the security system. This exit can be used to perform the same tasks as exits 1 and 2 in previous releases.

Exit 29 is called after the security system has completed processing for a security request. By setting the flag SRBXFAB flag in the Security Request Block (SRB), the exit can request that access be denied.

These parameters are passed:

- The address of the SRB
- The length of the SRB

Exit 29 in Signon Processing

Exit 29 may be called twice in signon processing:

- After completion of external signon (an external signon is performed if any resource, including signon itself, is externally secured).

  If the SRB function is SIGNON and the SRBXSGN flag is on, exit 29 determines that it has been invoked following external signon.
- After internal signon

  If the SRB function is SIGNON and the SRBXSGN flag is off, exit 29 determines that it has been invoked following internal signon.

How to Use This Exit for Security Purposes

You can use exit 29 to log security violations or to implement site-specific security enforcement requirements.

Specifically, you can maintain multiple activity bit maps for the DEFAULT application and use exit 29 to move one of the bit patterns to the SONSECTY field of the SON, depending on signon information. If exit 29 has been used to move such a value to SONSECTY, central security, as part of internal signon processing, allocates storage for the DEFAULT activity bit map and moves the bit pattern in SONSECTY there.

After external security and prior to calling internal security, the SON can be accessed using the SRBSGNSON address. The SON is not accessible using the LTESONRC.

Note: For more information about the SON (#SONDS DSECT), see the *CA IDMS DSECT Reference Guide*.

Considerations

Exit 29 cannot override a security violation.

You must write this exit routine to execute in SYSTEM MODE. The #DEFXIT macro, which adds the exit routine to the system, must perform the following:

- Specify MODE=SYSTEM
- Call the routine using either DC or IBM calling conventions
- Call the routine by entry point

Exit USRIDXIT

Description

The USRIDXIT exit can be used to provide a userid for batch jobs that need to run in a secured environment. This exit is mandatory at sites where no external security system is available. At the other sites, it can be used to override the userid that has been extracted from the batch address space by the external security system.

Considerations

A complete description on how to use this exit is given in the USRIDXIT source prototype that is delivered with the installation package.

Exit BTCIDXIT

Description

The BTCIDXIT exit is identical to the USRIDXIT exit, except for the batch jobs that use the 10.2-like batch interface (IDMSINTB module).

Considerations

A complete description on how to use this exit is given in the BTCIDXIT source prototype that is delivered with the installation package.

Using Installation Codes

What You Can Do

You can assign an installation code attribute value to the INSTCODE attribute keyword in a user or system profile. If you specify OVERRIDE=NO for INSTCODE, you can use the attribute for site-specific security checking. At runtime, an application program or exit 29 can retrieve the INSTCODE attribute value for the user session by linking to RHDCUF00 and issuing the DCUF SHOW INSTCODE command.

Note: For more information about linking to DCUF, see the *CA IDMS System Tasks and Operator Commands Guide*.

INSTCODE Considerations

In the logic to retrieve an INSTCODE attribute value, you should account for these possibilities:

- INSTCODE may not be an attribute of a given user session.
- An attribute value may be as many as 32 characters.

Using Terminal Autotasks

What is an Autotask

A terminal autotask is a task that you associate with a logical terminal. Through this association, you direct the CA IDMS system to initiate the preassigned task when either of these conditions occurs:

- The ENTER NEXT TASK CODE prompt would normally be displayed.
- A top-level program or dialog issues a DC RETURN request that specifies no next task code.

Suppose, for example, that a user who signs on to a CA IDMS system is assigned logical terminal LT12012. If this logical terminal is associated with an autotask, the system automatically initiates the autotask when the user connects to the terminal. After the user enters the name of the system, instead of displaying the ENTER NEXT TASK CODE prompt (which allows the user to choose what task to invoke), the terminal displays whatever screen is mapped out by the program associated with the preassigned autotask.

Securing a Terminal Autotask

Since terminal autotasks must be able to execute without a signed on user, they must be unsecured. If tasks are secured internally, you can unsecure terminal autotasks by assigning them to a category on which PUBLIC holds EXECUTE privilege. Alternatively, you can add an occurrence override to turn off security for each terminal autotask in the SRTT.

If tasks are secured externally, you must use an occurrence override in the SRTT to unsecure a terminal autotask.

When Should You Use an Autotask

Use an autotask for workers at one terminal who require access to *a limited number of data processing functions*.

For example, you can use an autotask for clerks who enter orders into a purchasing system. Or you can set up an autotask that executes a single manufacturing application. The application itself can perform a variety of functions, such as bill of material processing and master production scheduling.

In both of these cases, you create a menu for display when the autotask is invoked. This menu lists each option the user can choose, including a signon and signoff option if appropriate.

You can also use an autotask to invoke a **site-defined signon menu**. Set up this menu to appear when users connect to the terminal. After signing on to this menu, users invoke any task for which they are authorized.

How to Use an Autotask

To use an autotask, you must do the following:

- Associate an autotask with selected logical terminals.
- Optionally, establish signon and signoff functions as part of the autotask menu.
- Optionally, associate physical terminals with particular devices.
- Optionally, check a user's authority to use a particular terminal.

These topics along with the related design suggestions are discussed as follows.

Associating an Autotask with Selected Logical Terminals

Associating an Autotask with a Logical Terminal

You associate an autotask with a logical terminal using the AUTOTASK parameter of the system generation LTERM statement. This statement allows you to specify whether a task will be initiated automatically for the logical terminal:

- **AUTOTASK IS NULL**, the default, indicates that no task will be initiated automatically for this logical terminal. Logical terminals defined as printers *must* use the default value NULL.
- **AUTOTASK IS task-code** specifies the system will execute the specified task code automatically.

  The task code specified must be defined in the system dictionary with the system generation TASK statement. The task must be defined with the NOINPUT option because a terminal autotask should execute immediately.

Example

To associate LTERM LT12012 with the task code ORDERS, submit this information to the system generation compiler:

```
ADD LTERM LT12012 VERSION 1
  AUTOTASK IS ORDERS
  ENABLED
  PRIORITY IS 0
  PTERM IS PT12012.
```

Signon and Signoff Functions for an Autotask

Forcing Signon Through a Terminal Autotask

If you are using a site-specific signon menu to force a signon and you want users to see the ENTER NEXT TASK CODE prompt after they sign on to the CA IDMS system:

- Clear the autotask field (LTEAUTSK) in the logical terminal element (LTE) when the user signs on to the runtime system.
- Reset the autotask field when the user signs off from the runtime system.

Note: For more information about the LTE (#LTEDS DSECT), see the *CA IDMS DSECT Reference Guide*.

In a CA Technologies ADS Environment

If your autotask invokes a CA technologies ADS application, you can specify the following:

- Automatic signon for the application, by using a CA technologies ADS signon menu
- Automatic signoff for the application, by using the CA technologies ADS SIGNOFF function

Note: For more information about CA technologies ADS signon menus and the CA technologies ADS SIGNOFF function, see the *CA IDMS ADS Reference Guide*.

In an SQL or DML Environment

If your autotask invokes an SQL or DML application, design the application as follows:

1. When the user signs on through the autotask menu, link to the RHDCSNON program from the autotask program.

   Note: For more information, see the "SIGNON task" in the *CA IDMS System Tasks and Operator Commands Guide*.

   Alternatively, if no signon CLIST is invoked, an Assembler program can issue the #SECSGON macro to initiate user signon.

   Note: For more information, see #SECSGON (see page 218).

2. When the user signs off through the autotask menu, link to either RHDCSNOF or RHDCBYE:

   - RHDCSNOF is the program normally invoked by the CA IDMS system SIGNOFF task (RHDCSNOF leaves the ENTER NEXT TASK CODE prompt displayed on the terminal).
   - RHDCBYE is the program normally invoked by the CA IDMS system BYE task (RHDCBYE does not leave the ENTER NEXT TASK CODE prompt displayed on the terminal).

   Alternatively, if you do not wish to free resources as RHDCBYE and RHDCSON do, an Assembler program can issue the #SECSGOF macro to initiate user signoff.

   Note: For more information, see #SECSGOF (see page 215).

How to Clear and Reset the Autotask Field

To display the ENTER NEXT TASK CODE prompt after the signon menu, you clear the autotask field. You can reset the autotask field at signoff. You can do this by using exit 29 to move the appropriate value to the autotask field (LTEAUTSK) in the logical terminal element (LTE) after a successful signon or signoff. After signon, you move low values (binary zeros) to this field; after signoff, you move the appropriate task code.

Note: For more information about the LTE (#LTEDS DSECT), see the *CA IDMS DSECT Reference Guide*.

Associating Terminals with Devices

Overview

Once you associate an autotask with selected logical terminals, you need to ensure that these terminals correspond to particular devices. You do this by associating logical terminal/physical terminal pairs with the appropriate devices.

UCF, VTAM, and TCAM

For UCF, VTAM, and TCAM, you can choose whether to associate a logical/physical terminal pair with a particular device. If you do not explicitly associate a terminal pair with a device, the system takes the first available pair when a user signs on.

Other Access Methods

For all other access methods, you *must* associate terminal pairs with devices. In this case, users are always assigned a specific logical and physical terminal when they sign on to a particular device.

How to Associate Terminals with Devices

To associate UCF, VTAM, and TCAM terminals with particular devices, you use the NAME IS parameter of the system generation PTERM statement.

Suppose, for example, that the system definition includes logical terminal LT12012 and physical terminal PT12012. To associate this terminal pair with device FT068109, submit this information to the system generation compiler:

```
MODIFY PTERM PT12012
   ACQUIRE
   NAME IS FT068109.
```

Note: For more information about associating terminals with devices, see the *CA IDMS System Generation Guide*.

Checking Authority to Access a Particular Terminal

How to Check a User's Authority

When you implement autotasks, users can access only preassigned tasks at particular terminals. If you want to prevent users from accessing terminals that are *not* associated with a specific task, you must check their authority to use these terminals.

You can do this by:

1. Establishing installation codes that authorize users to access particular physical terminals.

   For example, you can specify a physical terminal ID, logical terminal ID, or VTAM node name as the attribute value for INSTCODE in the user or system profile.

2. Checking a user's INSTCODE attribute when the user signs on to a terminal.

   Note: For more information, see Using Installation Codes (see page 462).

Design Suggestions

Secure the DCMT VARY LTERM Command

Users can override the autotask assigned to a logical terminal with the DCMT VARY LTERM ONLINE command. If you are using autotasks in a secure environment, be sure to secure this DCMT command.

Note: For information about securing DCMT commands, see the chapter Securing System Resources (see page 95).

Associate Terminals with Devices Only when Necessary

When using autotasks, you should consider how many people will be using terminals that are restricted to particular tasks. You can use your terminal network more effectively if you associate terminals with devices only when necessary.

Terminal Association Criteria:

Need	Terminal configuration
A few people who will be using terminals restricted to an autotask	■ Associate the restricted terminals with particular devices. ■ Keep the other terminals generic (that is, not associated with particular devices). ■ Optionally, check a user's authorization to use a particular logical terminal at signon.
Many people who will be using terminals restricted to an autotask	■ Associate the non-restricted terminals with particular devices. ■ Keep the restricted terminals generic. ■ Optionally, check a user's authorization to use a particular logical terminal at signon.

Index