CICS/ESA
CICS-RACF Security Guide
Version 4 Release 1

Document Number SC33-1185-02

Note!

Before using this information and the product it supports, be sure to read the general information under “Notices” on page ix.

First edition (October 1994)

This edition applies to Version 4 Release 1 of the IBM licensed program Customer Information Control System/Enterprise Systems Architecture (CICS/ESA), program number 5655-018, and to all subsequent versions, releases, and modifications until otherwise indicated in new editions. Consult the latest edition of the applicable IBM system bibliography for current information on this product.

This book is based on the CICS-RACF Security Guide for CICS/ESA 3.3, SC33-1185-00. Changes from that edition are marked by vertical lines to the left of the changes.

The CICS/ESA 3.3 edition remains applicable and current for users of CICS/ESA 3.3.

Order publications through your IBM representative or the IBM branch office serving your locality. Publications are not stocked at the address given below.

At the back of this publication is a page entitled “Sending your comments to IBM”. If you want to make comments, but the methods described are not available to you, please address them to:

IBM United Kingdom Laboratories Limited, Information Development,
Mail Point 095, Hursley Park, Winchester, Hampshire, England, SO21 2JN.

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

Contents

Notices

The following paragraph does not apply in any country where such provisions are inconsistent with local law:
INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION “AS IS” WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore this statement may not apply to you.

References in this publication to IBM products, programs, or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any of the intellectual property rights of IBM may be used instead of the IBM product, program, or service. The evaluation and verification of operation in conjunction with other products, except those expressly designated by IBM, are the responsibility of the user.

IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to the IBM Director of Licensing, IBM Corporation, 500 Columbus Avenue, Thornwood, New York 10594, U.S.A..

Programming Interface information

This book is intended to help you use the IBM Resource Access Control Facility to provide security for CICS.

This book also documents General-use Programming Interface and Associated Guidance Information and Product-sensitive Programming Interface and Associated Guidance Information provided by CICS.

General-use programming interfaces allow the customer to write programs that obtain the services of CICS.

General-use Programming Interface and Associated Guidance Information is identified where it occurs, by an introductory statement to a chapter or section.

Product-sensitive programming interfaces allow the customer installation to perform tasks such as diagnosing, modifying, monitoring, repairing, tailoring, or tuning of CICS. Use of such interfaces creates dependencies on the detailed design or implementation of the IBM software product. Product-sensitive programming interfaces should be used only for these specialized purposes. Because of their dependencies on detailed design and implementation, it is to be expected that programs written to such interfaces may need to be changed in order to run with new product releases or versions, or as a result of service.

Product-sensitive Programming Interface and Associated Guidance Information is identified where it occurs, by an introductory statement to a chapter or section.

Trademarks and service marks

The following terms, used in this publication, are trademarks or service marks of IBM Corporation in the United States or other countries:

ACF/VTAM		CICS
CICS/ESA	CICS/MVS	CICS OS/2
CICS/VM	ESA/370	IBM
MVS/ESA	MVS/SP	MVS/XA
Operating System/2	OS/2	Personal System/2
PS/2	VTAM	RACF

Preface

What this book is about

This book is about using the IBM Resource Access Control Facility (RACF) to provide security for CICS.

If you need to know where programming interface information is described, or about the definitions of the different types of information in the CICS library, you should read the *CICS Family: Library Guide*.

Who this book is for

This book is intended for security administrators responsible for controlling access to resources used by CICS. These resources are used by CICS terminals, users, or transactions in CICS regions, and by CICS application programs running in those regions. The book will also be of interest for CICS system programmers who may need to communicate their requirements to the security administrator for their installation.

What you need to know to understand this book

It is assumed that you have a good working knowledge of RACF facilities. It is also assumed that you know something about the types of resource owned and controlled by CICS, at least at the level discussed in the *CICS/ESA 3.3 Facilities and Planning Guide*.

Although many RACF command examples are shown in this book, it is assumed that you have access to the *RACF Security Administrator's Guide* and that you know how to issue TSO commands (or use ISPF panels to perform equivalent functions).

How to use this book

The parts and chapters of this book are self-contained. Use an individual part or chapter where it contains information about the particular task you are engaged in. For example, see Part 4, "Customization" on page 211 if your task is to customize your CICS security processing.

Notes on terminology

In general, this book refers to the Customer Information Control System as "CICS." However, when it is necessary to distinguish between particular CICS versions, we use the following abbreviations:

- "CICS/OS/VS" is used for Customer Information Control System/Operating System/Virtual Storage.
- "CICS/MVS" is used for Customer Information Control System/Multiple Virtual Storage.
- "CICS/ESA" is used for Customer Information Control System/Enterprise Systems Architecture.
- Other abbreviations for CICS releases used in this book are as follows:
  - For CICS/OS/VS Version 1 Release 7 – CICS/OS/VS 1.7
  - For CICS/MVS Version 2 Release 1 and subsequent modification levels – CICS/MVS 2.1
  - For CICS/MVS Version 3 Release 1.1 – CICS/MVS 3.1.1
  - For CICS/ESA Version 3 Release 2.1 – CICS/ESA 3.2.1
  - For CICS/ESA Version 3 Release 3 – CICS/ESA 3.3
  - For CICS/ESA Version 4 Release 1 – CICS/ESA 4.1

"RACF" is used for Resource Access Control Facility.

- For RACF Version 1 Release 9 – RACF 1.9.2
- For RACF Version 2 Release 1 – RACF 2.1

RACF refers to any supported release of Version 1.9.2 or higher

RACF 1.9.2 or RACF 2.1 refer to function specific to that release or later.

The MVS/Enterprise Systems Architecture (MVS/ESA) operating system, which is a prerequisite for CICS/ESA Version 4, is generally referred to as "MVS."

For definitions of security-related CICS and RACF terms used in this book, see "Glossary" on page 295.

Bibliography

CICS/ESA 4.1 library

Evaluation and planning	
Release Guide	GC33-1161
Migration Guide	GC33-1162
General	
CICS Family: Library Guide	GC33-1226
Master Index	SC33-1187
User's Handbook	SX33-1188
Glossary (softcopy only)	GC33-1189
Administration	
Installation Guide	SC33-1163
System Definition Guide	SC33-1164
Customization Guide	SC33-1165
Resource Definition Guide	SC33-1166
Operations and Utilities Guide	SC33-1167
CICS-Supplied Transactions	SC33-1168
Program Directory	GC33-1200
Programming	
Application Programming Guide	SC33-1169
Application Programming Reference	SC33-1170
System Programming Reference	SC33-1171
Sample Applications Guide	SC33-1173
Distributed Transaction Programming Guide	SC33-1174
Front End Programming Interface User's Guide	SC33-1175
Diagnosis	
Problem Determination Guide	SC33-1176
Messages and Codes	SC33-1177
Diagnosis Handbook	LX33-6093
Diagnosis Reference	LY33-6082
Data Areas	LY33-6083
Supplementary Data Areas	LY33-6081
Communication	
Intercommunication Guide	SC33-1181
CICS Family: Inter-product Communication	SC33-0824
CICS Family: Communicating from CICS/ESA and CICS/VSE	SC33-0825
Special topics	
Recovery and Restart Guide	SC33-1182
Performance Guide	SC33-1183
CICS-IMS Database Control Guide	SC33-1184
CICS-RACF Security Guide	SC33-1185
Shared Data Tables Guide	SC33-1186
External CICS Interface	SC33-1390

Other CICS books

- *CICS Application Programming Primer (VS COBOL II)*, SC33-0674
- *CICS Application Migration Aid Guide*, SC33-0768
- *Transaction Processing: Concepts and Products*, GC33-0754
- *CICS Family: API Structure*, SC33-1007
- *CICS/ESA XRF Guide* for CICS/ESA Version 3 Release 3, SC33-0661
- *CICS Family: General Information*, GC33-0155
- *CICS/ESA Facilities and Planning Guide* for CICS/ESA Version 3 Release 3, SC33-0654
- *IBM CICS Transaction Affinities Utility MVS/ESA*, SC33-1159

Books from related libraries

MVS/ESA

MVS/ESA Message Library: System Codes, GC28-1815

MVS/ESA Planning: Security, GC28-1604

MVS/ESA System Programming Library: Initialization and Tuning, GC28-1828.

External Security Interface (RACROUTE) Macro Reference for MVS and VM, SC28-1366

Systems Network Architecture (SNA)

SNA Reference: Peer Protocols, SC31-6808

RACF 1.9.2

RACF Auditor's Guide, SC28-1342

RACF Command Language Reference, SC28-0733

RACF Diagnosis Guide, LY28-1016

RACF General User's Guide, SC28-1341

RACF Macros and Interfaces, SC28-1345

RACF Messages and Codes, SC38-1014

RACF Security Administrator's Guide, SC28-1340

RACF System Programming Library, SC28-1343

RACF Data Areas, LY28-1830

RACF Secured Sign-on SPE Information Package Version 1 Release 9.2, SC23-3765

RACF 2.1

RACF Auditor's Guide, SC23-3727

RACF Command Language Reference, SC23-3731

RACF Diagnosis Guide, LY27-2635

RACF General User's Guide, SC23-3728

RACF Macros and Interfaces, SC23-3732

RACF Messages and Codes, SC23-3730

RACF Security Administrator's Guide, SC23-3726

RACF System Programmer's Guide, SC23-3725

RACF Data Areas, LY27-2636

Summary of changes

Changes for this edition

The main changes made for CICS/ESA 4.1 are as follows:

- Chapter 1, “Security facilities in CICS” on page 3 contains additional introductory information on the following:
  - Non-terminal security
  - Surrogate user security
  - MRO security
  - CICS/ESA Front End Programming Interface security
  - Generating and using RACF PassTickets
- Chapter 2, “RACF facilities” on page 9 has been amended as follows:
  - “Console profiles” on page 25 discusses security control that can be implemented on the console user
  - The signon table (SNT) has been removed. This change affects “CICS segment” on page 14, “Defining XRFSOFF” on page 16, and “CICS default user” on page 17.
  - Information about TIMEOUT data is included in the section on “CICS segment” on page 14.
  - “Generating and using RACF PassTickets” on page 8 has been added.
  - The ability to define each user as belonging to several groups is mentioned in “Security classification of data and users” on page 23.
  - Situations in which it is necessary to use the PERFORM SECURITY REBUILD command are indicated in “Refreshing resource profiles in main storage” on page 29.
  - Specific information about refreshing resource profiles has been added on several pages.
- Chapter 3, “CICS data set and system security” on page 39 has been amended as follows:
- CICS/ESA 4.1 does not work with RACF versions before 1.9. Any mention of earlier versions of RACF has been removed from “CICS-supplied RACF dynamic parse validation routines” on page 39.
- “SEC” on page 54 has been updated to reflect the fact the SEC system initialization parameter no longer supports MIGRATE.
- The XUSER resource class for surrogate user checking has been included in Table 5 on page 57.
- Chapter 4, “Verifying CICS users” on page 63 has been changed in the following ways:
  - Mention of TCAM terminals has been removed from “Controlling access to CICS from specific ports of entry” on page 67.
  - Surrogate user checking is mentioned in “Surrogate job submission in a CICS environment” on page 52.

- The way CICS obtains information about users is reflected in "Obtaining CICS-related data for a user" on page 72.
- Information about national languages, and non-terminal transactions appears in "National language and non-terminal transactions" on page 74.
- Details of CSGM and CESN have been moved to *CICS/ESA CICS-Supplied Transactions*.

- Chapter 6, "Resource security" on page 83 now includes information about transactions not attached to terminals (see "Transactions started without terminals" on page 93).
- Chapter 7, "Surrogate user security" on page 103 is a new chapter about Surrogate user security.
- In Chapter 8, "CICS command security" on page 109, several new resources and their related CICS commands have been added to Table 12 on page 110.
- SEC=MIGRATE is no longer supported, so references to this option have been removed from Chapter 10, "Security for CICS-supplied transactions" on page 125.
- In Chapter 12, "Implementing LU6.2 security" on page 141, the following changes have been made:
  - A new section describes attach-time security processing and addition of SNA profile support. See "SNA profiles and attach-time security" on page 152.
  - Mention of internal bind-time security has been removed. (See "Defining bind-time security" on page 143).
  - The use of ATTACHSEC(VERIFY) in addition to ATTACHSEC(IDENTIFY) in checking the user identifier has been included in "Specifying user security in link definitions" on page 148.
- Information about CICS-APPC password expiration management, which previously appeared in a separate manual, is included in Chapter 13, "APPC password expiration management" on page 161 and Appendix A, "APPC PEM requester example program" on page 265.
- In Chapter 15, "Implementing MRO security" on page 189, the following changes have been made:
  - "Bind-time security with MRO" on page 189 has been reorganized and expanded because of the introduction of an external security manager and the cross-system coupling facility (XCF).
  - Information on the external call interface has been included in "Distributed program link security with MRO" on page 200.
- Chapter 16, "Security for shared data tables" on page 205 has been added and provides information on the security checks available when using SDT.
- In Chapter 17, "Customizing security processing" on page 213, the operation of the external security manager is described in the section, "Determining the userid of the CICS region" on page 218.
- Chapter 20, "Problem determination in a CICS-RACF security environment" on page 249 has been updated to reflect the levels of RACF for which PERFORM SECURITY REBUILD is still necessary. Mention of RACF releases earlier than 1.9 has also been removed.

- Changes have been made in various chapters to reflect the replacement of several message numbers (for example, DFHXS0100 by DFHXS1111).
- The new Appendix B, “National Language” on page 287, provides information on language codes that can be defined to a user in the LANGUAGE segment of RACF.
- A new Appendix C, “API command and resource check cross reference” on page 289, has also been added.

Changes are indicated by vertical lines to the left of the changes.

Part 1. Introduction

This part introduces you to the subject of CICS security, using RACF as the CICS external security manager. It provides an overview of the CICS security requirements, and the facilities RACF provides to satisfy those requirements. Part 1 contains:

- Chapter 1, “Security facilities in CICS” on page 3 introduces you to the various aspects of CICS transaction and resource security.
- Chapter 2, “RACF facilities” on page 9 describes the basic facilities that RACF provides, and that CICS relies upon for its security administration.

Chapter 1. Security facilities in CICS

This chapter describes, from the CICS viewpoint, the following aspects of CICS transaction and resource security:

Why CICS needs security

Today, an unprecedented number of computer system users are completely dependent on their systems, and on the data managed by those systems. There are now terminals in many different locations in most organizations, and their use is commonplace. At the same time, easy-to-use, high-level inquiry languages are available, and there is much greater familiarity with data processing methods. This means that more and more people can use computers to retrieve or modify data stored within a computer system.

The speed, flexibility, and size of modern systems make large quantities of data accessible to many terminal users. As the systems become easier to use, there is also more scope for terminal users to gain access to confidential or valuable data.

Without a corresponding growth in awareness of good data security practices, these advances can result in accidental (or deliberate) data exposure. This means that your data can be subject to:

- Unauthorized access
- Disclosure
- Modification
- Destruction

As an online transaction-processing system (often supporting many thousands of terminals), CICS clearly needs the protection of a security system to ensure that the resources to which it manages access are protected, and secure from unauthorized access.

To provide the necessary security for your CICS regions, CICS uses the MVS system authorization facility (SAF) to route authorization requests to an external security manager (ESM), such as RACF, at appropriate points within CICS transaction processing.

What CICS security protects

Let us take a brief look at the types of assets that CICS manages, and the potential exposures of those assets. The assets are the application programs, the application data, and the application output. To prevent disclosure, destruction, or corruption of these assets, you must first safeguard the CICS system components themselves.

There are two distinct areas from which exposures to the CICS system can arise. The first of these is from sources external to CICS. You can use RACF data set protection as the primary means of preventing unauthorized access, from either TSO users or batch jobs, to the assets CICS manages.

The other potential area of exposure arises from CICS users. CICS provides a variety of security and control mechanisms. These can limit the activities of CICS terminal users to only those functions that any particular individual user is authorized to use.

What CICS security does not protect

CICS itself does *not* provide facilities to protect its own assets from external access. You should restrict access to the program libraries, to the CICS regions, and to the programmers responsible for incorporating approved application and system changes. Similarly, the data sets and databases used by CICS and by CICS applications must be accessible only by approved batch processing and operations procedures.

CICS does not protect your system from application programs that use undocumented or unsupported interfaces to bypass CICS security. You are responsible for ensuring that such programs are not installed on your system.

CICS does not protect your application source libraries. You should ensure that procedures are established and followed that prevent the introduction of unauthorized or untested application programs into your “production” application base. You should also protect the integrity of your system by exercising control over libraries that are admitted to the system, and changes to those libraries.

Terminal user security

To secure resources from unauthorized access, CICS needs some means of uniquely identifying individual users of the system. For this purpose, you must first define the users to RACF by creating an entry in the RACF database, referred to as a **user profile**. To identify themselves to CICS, users sign on by specifying their RACF user identification (userid), and the associated password, or operator identification card (OIDCARD) in the CICS-supplied signon transaction, CESN. Alternatively, an equivalent transaction developed by your own installation can be used by issuing the EXEC CICS SIGNON command provided for this purpose.

When users enter the CESN transaction, CICS verifies userids and passwords by a call to RACF. If the terminal user signon is valid, the CICS user domain keeps track of the signed-on user. Thereafter, CICS uses the information about the user when calling RACF to make authorization checks.

See “Terminal profiles” on page 23 for information about the terminal security facilities provided by RACF. See Chapter 4, “Verifying CICS users” on page 63 for information about using terminal user security in CICS.

Preset terminal security

For some selected terminals, and MVS consoles when used as CICS terminals, you should consider using CICS preset terminal based security as an alternative to terminal user security. A terminal becomes a preset security terminal when you specify the USERID operand on the terminal definition.

CICS preset terminal security allows you to permanently associate a userid with a terminal that is defined to CICS. This means that CICS implicitly “signs on” the terminal when it is being installed, instead of the terminal being signed on subsequently. Preset security is often defined for devices without keyboards, such as printers, at which users cannot sign on.

You can also use this form of security on ordinary display terminals as an alternative to terminal user security. This permits anyone with physical access to a terminal with preset security to enter the transactions that are authorized for that terminal, without the need to sign on to CICS. The terminal remains signed on as long as it is installed, and no explicit signoff can be performed against it. If the userid associated with a display terminal with preset security has been authorized to use any sensitive transactions, you should ensure that the terminal is in a secure location to which access is restricted. The terminals physically located within a CICS network control center would be ones for which preset security might be appropriate.

You can use preset security to assign a userid with *lower* authority than the default, for terminals in unrestricted areas.

For example, to define a terminal with preset security, use RACF and CICS (CEDA) commands as follows:

```
ADDUSER userid NAME(preset_terminal_user_name) OWNER(owner_userid or group_id)
               DFLTGRP(group_name)
CEDA DEFINE TERMINAL(cics_termid) NETNAME(vtam_termid) USERID(userid)
            TYPETERM(cics_typeterm)
```

For further information on preset security terminals in the transaction routing environment refer to “Preset security terminals and transaction routing” on page 155 (LU6.2 security) and “Preset security terminals and transaction routing” on page 198 (MRO security).

Non-terminal security

Security can also be specified for transactions that are not associated with terminals. These are:

- Started non-terminal transactions
- Transient data trigger-level transactions
- Program List Table (PLT) programs that run during CICS initialization.

For more information about non-terminal security see “Transactions not associated with a terminal” on page 81.

Transaction security

The CICS facilities for transaction security ensures that CICS calls RACF each time a transaction is initiated, to verify that the userids associated with that transaction are permitted access.

See “General resource profiles” on page 27 for information about the resource classes that RACF supports for CICS transaction security. See Chapter 5, “Transaction security” on page 77 for information about using transaction security.

CICS resource security

You can control access to CICS resources that a transaction uses. You do this by specifying YES on the resource security parameter, RESSEC, in the CICS TRANSACTION resource definition. These CICS resources can be:

- Application programs
- DL/I program specification blocks (PSBs)
- Files — VSAM and BDAM
- Journals
- Temporary storage queues
- Transient data queues
- Transactions initiated by a CICS START command.

See Chapter 6, “Resource security” on page 83 for information about using CICS resource security.

CICS command security

You can control security for a system programming subset of the CICS application programming interface (SPI) commands. You do this by specifying YES in the command security parameter, CMDSEC, on the CICS TRANSACTION resource definition. This is known as CICS command security, and operates on all the commands that require the special CICS translator option, SP. (These can be seen in Table 11 on page 109). Command security operates in addition to any transaction or resource security you define for a transaction. For example, if a user is permitted to use a transaction called FILA, which issues an EXEC CICS INQUIRE FILE command that the user is *not* permitted to use, CICS issues a “not authorized” (NOTAUTH) condition in response to the command, and the command fails.

See “General resource profiles” on page 27 for information about the resource classes that RACF supports for CICS command security. See Chapter 8, “CICS command security” on page 109 for information about using CICS command security.

Surrogate user security

CICS performs surrogate user security checking in a number of instances to ensure that the surrogate user is authorized to act for the other user. For more information see Chapter 7, “Surrogate user security” on page 103.

Surrogate user checking can be enforced for:

- CICS default user
- Started transactions
- Preset terminal security
- PLT security
- Installation of transient data queues
- Installation of default user.

QUERY SECURITY command

In addition to using CICS security checking for CICS-controlled resources (or as an alternative to it), you can use the EXEC CICS QUERY SECURITY command to control security access within the CICS application. This method also allows you to define security profiles to RACF for resources other than CICS resource profiles, and enables a more detailed level of security checking than is available through the default resource classes.

See “General resource profiles” on page 27 for information about the resource classes that RACF supports for resource security checking within transactions. For more information about resource security checking, see Chapter 6, “Resource security” on page 83.

APPC (LU6.2) session security

So far, all the discussion has been about the security CICS performs for transactions running within a single CICS region, with its own resources and terminal network. A number of CICS regions can also be connected by means of **intercommunication**; for example, **intersystem communication (ISC)** using an SNA access method, such as ACF/VTAM, to provide the necessary communication protocols. This method is normally used for communication between CICS regions residing in different host computers, but it can also connect CICS regions in the same host computer. (See the *CICS/ESA Intercommunication Guide* for more information about CICS intercommunication facilities.)

One of the ISC protocols that CICS uses is for advanced program-to-program communication (APPC), which is the CICS implementation of the LU6.2 part of the SNA architecture.

For interconnected systems, the same basic security principles apply, but the resource definition is more complex, and you have additional security requirements. CICS treats APPC sessions, connections, and partners as resources, all of which have security requirements. In addition to the transaction, resource, and command security introduced earlier, CICS provides the following security mechanisms for the APPC environment:

- Bind-time (or session) security, to prevent an unauthorized connection between CICS regions

- Link security defines the authority of the remote system to access transactions or resources to which the connection itself is not authorized
- User security checks that a user is authorized both to attach a transaction and to access all the resources and SP-type commands that the transaction is programmed to use.

See Chapter 12, “Implementing LU6.2 security” on page 141 for more information.

Multiregion operation (MRO) security

Another means of using intercommunication is **multiregion operation** (MRO). This is available for links between CICS regions in a single sysplex, independent of the systems network architecture (SNA) access method. See Chapter 15, “Implementing MRO security” on page 189 for information about MRO security.

CICS/ESA Front End Programming Interface security

The security options provided for the CICS/ESA Front End Programming Interface are equivalent to those provided for CICS command security (see page 6). Front End Programming Interface security is not discussed in this book, but in the *CICS/ESA Front End Programming Interface User's Guide*.

Generating and using RACF PassTickets

A PassTicket is a character string generated by a program that can be used in place of a password, with the following constraints:

- A specific PassTicket may be submitted for validation only *once*.
- The PassTicket must be used within 10 minutes of being generated.
- To ease the problem of system time differences, a specific PassTicket can be used up to 10 minutes earlier or later in a target system, compared to the generating system.

FEPI can generate a PassTicket for use on a target system. RACF 2.1 must be installed on your system in order to generate a RACF PassTicket. Wherever a password can be presented in CICS/ESA 4.1, a PassTicket can be used as a substitute.

Note: The PassTicket generation and validation algorithm means that the system that creates the PassTicket and the system that validates it must both use the same level of this function. That is, if the creating system has the function applied, and the validating system does not, the PassTicket is invalid.

For more information about the system time differences, and the use of the PassTicket within the 10 minute interval, see the *RACF Security Administrator's Guide*.

Chapter 2. RACF facilities

For its security management capability, CICS relies upon a number of facilities provided by RACF. Although RACF provides the basic security access and authorization facilities, it does not by itself perform any security checking.

This chapter covers:

Overview

RACF provides the following facilities:

- The necessary functions to record information identifying individual users of system resources, and information identifying the resources that require protection. The information you define to RACF about users and resources is stored in user and resource **profiles**.
- The facilities to define which users, or groups of users, are either permitted access, or excluded from access, to the resources for which profiles have been defined. The information recording the users, or groups of users, permitted to access any particular resource is held in an **access list** within the profile that protects a resource.
- A method to process requests, issued by subsystems or jobs running in an MVS system, to authenticate the identity of users defined to RACF, and to check their access authorization to resources. The requesting subsystem or job uses the responses to these requests to determine the course of subsequent processing.
- The facilities for logging security-related events, such as users signing on and signing off, the issuing of RACF commands, and attempts to access protected resources. Successful attempts to access protected resources may be recorded by the MVS System Management Facility (SMF). If you want to record all attempts to access protected resources, whether successful or not, use RACF auditing, as described in the *RACF Auditor's Guide*.) RACF records security-relevant events in the SMF data set. The RACF auditor can run the RACF report writer to generate reports based on the SMF records.

  For information on using RACF to perform **auditing** functions (specifying auditing operands on RACF commands, and using the RACF report writer to

generate reports of audited security-related activity), see the *RACF Auditor's Guide*.

RACF administration

As the security administrator for one or more CICS regions, and for the users of the CICS applications, it is your job to ensure that your installation's data is properly protected. Using RACF, you are responsible for protecting all system resources, and, in the context of this manual, CICS resources in particular.

A key feature of RACF is its hierarchical management structure. The RACF security administrator is defined at the top of the hierarchy, with authority to control security for the whole system. If you are not yourself the RACF security administrator, you must ask that person to delegate to you sufficient authority to work with RACF profiles and system-wide settings. You must also work with the RACF auditor, who can produce reports of security-relevant activity based on auditing records generated by RACF.

RACF security administrators have either the system-SPECIAL attribute, the group-SPECIAL attribute, or a combination of other authorities.

- If you have the system-SPECIAL attribute, you can issue any RACF command, and you can change any RACF profile (except for some auditing-related operands).
- If you have the group-SPECIAL attribute, your authority is limited to the scope of the RACF group for which you have the SPECIAL attribute.
- The other authorities include:
  - The CLAUTH (class authority) attribute, which allows you to define RACF profiles in specific RACF classes
  - Having the authority that goes with being the OWNER of existing RACF profiles, which allows you to list profiles change the access, and delete them
  - Having a group authority such as CONNECT or JOIN in a RACF group.

For complete information about the authorities required to issue RACF commands, and for information on delegating authority and the scope of a RACF group, see the *RACF Security Administrator's Guide*.

For information on the RACF requirements for issuing RACF commands, see the descriptions of the commands in the *RACF Command Language Reference* manual.

You can find out whether you have the system-SPECIAL or group-SPECIAL attribute by issuing the LISTUSER command from a TSO/E session. If you have the system-SPECIAL attribute, SPECIAL appears after the USER ATTRIBUTES phrase in the first part of the output. If you have the group-SPECIAL attribute, SPECIAL appears after the USER ATTRIBUTES phrase in the offset section that describes your connection to a RACF group. For a complete description, with an example of LISTUSER output, see the *RACF General User's Guide*.

Delegation of RACF administrative responsibility

As CICS security administrator, you need to do the following tasks (if you do not have the system-SPECIAL attribute, you need to be given the necessary authority):

- *Define and maintain profiles in CICS-related general resource classes*. In general, authority to do this is granted by assigning a user the CLAUTH (class authority) attribute in the specified classes. For example, the RACF security administrator could issue the following command:

  ```
  ALTUSER  your_userid  CLAUTH(TCICSTRN)
  ```

  The above command gives access to all classes of the same POSIT number. The POSIT number is an operand of the ICHERCDE macro of the class descriptor table (CDT).

  Many of the general resource classes mentioned in this book (such as APPL, APPCLU, FACILITY, OPERCMDS, SURROGAT, TERMINAL, and VTAMAPPL) affect the operation of products other than CICS. If you are not the RACF security administrator, you may need to ask that person to define profiles at your request.

- *Add RACF user profiles to the system*. In general, this authority is granted by assigning the CLAUTH (class authority) attribute in the user's profile. For example, the RACF security administrator could issue the following command:

  ```
  ALTUSER  your_userid  CLAUTH(USER)
  ```

  Whenever you add a user to the system, you must assign that user a default connect group. Assigning that user a default connect group changes the membership of the group (by adding the user as a member of the group). Therefore, if you have JOIN group authority in a group, the group-SPECIAL attribute in a group, or are OWNER of a group, CLAUTH(USER) lets you add users to the system and connect them to groups that are within the scope of the group.

- *List RACF system-wide settings and work with all profiles related to CICS*. Authority to do this can be given by setting up a RACF group, ensuring that certain CICS-related RACF profiles are in the scope of that group, and connecting a user to the group with the group-SPECIAL attribute. For example, the RACF security administrator could issue the following command:

  ```
  CONNECT  your_userid  GROUP(applicable-RACF_groupid)  SPECIAL
  ```

| With the SETROPTS GENERICOWNER command in effect and prefixing active, administrators can be assigned. This is done by creating a generic profile in each class using the prefix as a high-level qualifier. For example:

```
RDEFINE  TCICSTRN   cics_region_id.**  UACC(NONE)
         OWNER(cics_region_administrator_userid)
```

| The SETROPTS GENERIC command must be used before defining generic profiles, as described in "Brief summary of RACF commands" on page 21.

| For more information on delegation of RACF administration, see the *RACF Security Administrator's Guide*.

RACF user profiles

RACF holds user data in the form of user profiles in the RACF database. These user profiles can consist of one or more segments — a RACF segment, and others that are optional. For CICS users, the important segments are:

- The RACF segment, which holds the basic information for a RACF user profile
- The CICS segment, which holds data for each CICS user
- The LANGUAGE segment, which specifies the user's national language preference.

These segments are explained briefly in the following sections.

Table 1 on page 13 summarizes where the RACF userids for different types of CICS users are obtained. Note that users who issue operator commands from operator consoles must be authorized to issue the MODIFY command, as described in "OPERCMDS resource class" on page 33, as well as having authority to issue the CICS transaction, as described in Chapter 5, "Transaction security" on page 77.

Table 1. Types of CICS users and their userids

User type	Userid obtained from
Region user	The userid under which the CICS region executes. It is specified in the RACF ICHRIN03 started-procedures table or the USER parameter of the CICS startup JOB statement, or in the STARTED class.
CICS default user	The userid specified on DFLTUSER in the system initialization parameters or at startup. It is used for terminal users who have not signed on. (See "CICS default user" on page 17.)
PLTPI user	The userid for PLTPI programs. It is specified on the PLTPIUSR system initialization parameter. Default is the Region Id.
CICS terminal users who sign-on	The userid specified by a terminal user during explicit sign-on. (See "Identifying CICS terminal users" on page 63.)
Preset terminal user	The userid specified on the terminal definition. (See "Preset terminal security" on page 5.)
ATI user	The userid specified in DFHDCT TYPE=INTRA,USERID parameter, or EXEC CICS SET TDQUEUE ATIUSERID option.
Started transaction user	The userid for a started non-terminal transaction.
Link user	The userid used during MRO or ISC communication. (See "Link security" on page 136.)
Remote user	The userid for a transaction attached by the userid on a remote system, for example, by using transaction routing.
Surrogate user	The userid specified for a user who has the authority to start work on behalf of another user and is authorized to act for that user. See Chapter 7, "Surrogate user security" on page 103.
Surrogate job user	The userid used for batch jobs submitted by CICS, but not using the region userid. (See "Coding the USER parameter on the CICS JOB statement" on page 43.)

RACF segment

You identify a RACF user by an alphanumeric userid, which RACF associates with the user profile for that user. The "user" that you define to RACF need not be a person, such as a CICS terminal user. For example, in the CICS environment, a RACF userid can be associated with the procedure you use to start CICS as a started task; and a userid can be associated with a CICS terminal (for the purpose of preset security).

The following list shows some of the basic segment information that RACF holds for a user:

Keyword	Description
USERID	The userid of the user
NAME	The user's name

OWNER The owner of the user's profile—the RACF administrator or other user authorized by the administrator, or a RACF group

DFLTGRP The default group that the user belongs to

AUTHORITY The user's authority in the default group

PASSWORD The user's password.

You define the RACF segment of a user profile using the ADDUSER command, or the RACF ISPF panels. When planning RACF segments of user profiles for CICS users, you need to identify the groups that you want them to be in. Start by identifying RACF administrative units for the users. For example, you could consider all users who have the same manager, or all users within an order entry function, as administrative units. RACF handles these units as groups of individual users who have similar requirements for access to CICS system resources.

For an overview of the steps required to add users to the system, see the *RACF Security Administrator's Guide*.

CICS segment

The CICS segment of the RACF user profile contains data for CICS users. For information on the order in which CICS searches for the operator information, see "Obtaining CICS-related data for a user" on page 72.

CICS user data

The information you can specify in the CICS segment is as follows:

OPCLASS

The operator classes are for use by CICS when routing basic mapping support (BMS) messages initiated within a CICS transaction. The operator classes are numeric values in the range 1–24.

You need to specify operator classes for users who use CICS transactions that issue EXEC CICS ROUTE commands with the (optional) OPCLASS parameter. You specify the corresponding value as an operator class in the CICS segment of the user profile for automatic routing to occur.

See the *CICS/ESA Application Programming Guide* for information about BMS and the use of the OPCLASS parameter for routing messages.

The default value for OPCLASS is 1. (See "When the defaults are effective" on page 15.)

OPIDENT

The 1–3 character operator identification code that you assign to each operator.

CICS stores the code in the operator's terminal entry in the CICS terminal control table (TCTTE) when the operator signs on. This operator ID is displayed in certain CICS messages and can also be used in the EXEC CICS ROUTE command for routing BMS messages. (For more information about BMS, see the *CICS/ESA Application Programming Guide*). The operator ID forms part of the "Unit of Work" identifier which is used in recovery. It is also used when using the CEDA LOCK function, as described in the *CICS/ESA Resource Definition Guide*.

The default value for OPIDENT is blank. (See "When the defaults are effective" on page 15).

OPPRTY
The operator priority value—a decimal number that you want CICS to use when determining the task priority for CICS transactions that the operator invokes at a CICS terminal. The priority value can be in the range 0 through 255, where 255 is the highest priority.

CICS uses the sum of operator priority, terminal priority, and transaction priority to determine the dispatching priority of a transaction.

The default value for OPPRTY is 0. (See "When the defaults are effective.")

TIMEOUT
The time that must elapse since the user last used the terminal before CICS "times-out" the terminal.

For RACF Version 2.1 the time must be a decimal integer in the range 0 through 9959 (the rightmost two digits represent a number of minutes, and must be 00 through 59. Any digits to the left of these represent hours).

For RACF Version 1.9 the time must be a decimal integer in the range is 0 through 60.

To specify one hour and eight minutes (RACF 2.1 only) you would code a value here of 0108. For example:

```
ALTUSER userid CICS(TIMEOUT(0108))
```

The value of 0 (the default), means that the terminal is *not* timed out (see "When the defaults are effective"). For a discussion of the coexistence issues, see "Extending timeout values" on page 244.

XRFSOFF
The CICS extended recovery facility (XRF) sign-off option. You specify this to indicate whether or not you want CICS to sign off the operator following an XRF takeover.

FORCE
Specify FORCE if you want CICS to sign off the operator automatically in the event of an XRF takeover.

NOFORCE
Specify NOFORCE if you want CICS to leave an operator signed on in the event of an XRF takeover.

The default value for XRFSOFF is NOFORCE. (See "When the defaults are effective").

When the defaults are effective

The defaults listed are effective only when a CICS segment has been defined for that userid. You can make the CICS segment default by defining it as follows:

```
ADDUSER userid DFLTGRP(group_name) NAME(user_name)
        OWNER(group_id | userid)
        PASSWORD(password)
        CICS
```

For example, you may want to define a CICS segment in this way if you want to enforce the *system* defaults, rather than defaulting to the default user attributes, or

if you are setting up a test system and have not yet decided on the values you want to use.

If you omit the CICS segment completely, defaults are obtained as described in "Obtaining CICS-related data for a user" on page 72.

If you specify some of the CICS segment options, but omit others, the defaults described above apply to the omitted options,

You can remove the CICS segment as follows:

```
ALTUSER userid NOCICS
```

Defining XRFSOFF

The XRFSOFF function is also available at the TYPETERM definition level, as described in the *CICS/ESA Resource Definition Guide*, and at the CICS system level in the form of a system initialization parameter, as described in the *CICS/ESA System Definition Guide*. (As for the CICS segment, the default value for XRFSOFF in the system initialization parameters and in the TYPETERM definition is NOFORCE.)

Note that the FORCE option in the system initialization table or the TYPETERM definition overrides NOFORCE in the CICS segment.

Table 2 shows how specifying FORCE or NOFORCE in the system initialization parameters, on the TYPETERM definition (or the terminal control table (TCT)), and in the CICS segment together determine whether a terminal remains signed on after an XRF takeover.

As Table 2 shows, for a terminal to remain signed-on after an XRF takeover, NOFORCE must be specified in all three locations.

Table 2. Effects of FORCE and NOFORCE options

TYPETERM definition	CICS segment	System initialization parameter	
		FORCE	NOFORCE
FORCE	FORCE	Signed-off	Signed-off
	NOFORCE	Signed-off	Signed-off
NOFORCE	FORCE	Signed-off	Signed-off
	NOFORCE	Signed-off	**Signed-on** (see note)

Note: If takeover has exceeded the time specified by the XRFSTME system initialization parameter, users at terminals that have a nonzero TIMEOUT value do not remain signed-on after takeover. For example, suppose the following has been specified in a system that has XRFSOFF=NOFORCE:

```
RDEFINE USER1 CICS(XRFSOFF(NOFORCE) TIMEOUT(10))
RDEFINE USER2 CICS(XRFSOFF(NOFORCE) TIMEOUT(1))
```

If an XRF takeover occurs to a system in which XRFSTME=5 is specified in the system initialization parameters, and that takeover takes longer than five minutes, USER1 does not remain signed-on, but USER2 does.

CICS default user

When you are using CICS with external security, CICS assigns the security attributes of the CICS **default user** to all CICS terminal users that do not sign on. CICS also assigns the operator data from the CICS segment of the default user to signed-on users who do not have their own CICS segment data. To enable CICS to assign default security attributes and operator data, you must define a CICS default userid to RACF. You must then tell CICS which default user to use by specifying the DFLTUSER system initialization parameter. (See the *CICS/ESA System Definition Guide* for information about this parameter.) If you do not specify a default userid on the DFLTUSER parameter, CICS uses the name "CICSUSER."

Whether you use installation-defined operator data on your DFLTUSER parameter, or use the default, it is essential that the userid is defined to RACF and that the region userid has installed surrogate security to use the default user (see "Surrogate user security" on page 7).

CICS "signs on" the default user during system initialization. **If you specify SEC=YES as a system initialization parameter, and CICS cannot "sign on" the default userid, CICS initialization fails**.

CICS uses the security attributes of the default userid to perform all the security checks for terminal users that do not explicitly sign on. These security checks include **resource** and **command** security checking, in addition to **transaction-attach** security checking.

LANGUAGE segment

The language segment holds information about the national language in which the user receives messages. You can specify two languages, but CICS assigns only one language to a user. It assigns the primary language if it is specified and CICS supports that language. If the primary language is not specified or is not supported, CICS assigns the secondary language if it is specified and CICS supports it.

Specify the user's preferred national languages in the LANGUAGE segment of the RACF user profile, using the LANGUAGE parameter on the ADDUSER or ALTUSER command:

LANGUAGE
Use this parameter to specify primary and secondary languages for CICS users. CICS accepts and uses the languages you define in the segment, but ignores the RACF system-wide defaults. This is because CICS has its own system default for national languages, which you specify on the CICS system initialization parameter, NATLANG.

PRIMARY(primary_language)
This parameter identifies the user's primary language, overriding the system default. Depending on the national language feature you have installed, you can specify this as one of the 3-character codes in Appendix B, "National Language" on page 287.

SECONDARY(secondary_language)
This parameter identifies the user's secondary language, overriding the system default. You can specify this as one of the codes listed in Appendix B, "National Language" on page 287.

For more information about national language, see “National language and non-terminal transactions” on page 74.

Creating or updating segment data for a CICS user

To create or update CICS segment data for a CICS user, specify the CICS option on the RACF ADDUSER command for a new user, and on the ALTUSER command for an existing user. For example, the following command adds a new CICS user to the RACF database with associated CICS operator data:

```
ADDUSER userid  DFLTGRP(group_name) NAME(user_name) OWNER(group_id)
        PASSWORD(password)
        CICS(OPCLASS(1,2,...,n) OPIDENT(identifier) OPPRTY(priority)
             TIMEOUT(timeout_value) XRFSOFF(NOFORCE))
        LANGUAGE(PRIMARY(primary_language))
```

The following example of the ALTUSER command adds CICS operator data to an existing user in the RACF database:

```
ALTUSER userid
        CICS(OPCLASS(1,2,...,n) OPIDENT(identifier) OPPRTY(priority)
             TIMEOUT(timeout_value) XRFSOFF(NOFORCE))
        LANGUAGE(PRIMARY(primary_language))
```

Before issuing these commands to define CICS operator data, you must ensure that the CICS-supplied RACF dynamic parse validation routines are installed in an APF-authorized library in the linklist. See “CICS-supplied RACF dynamic parse validation routines” on page 39 for details of these exits.

If you do not have the system-SPECIAL attribute, ask your RACF security administrator for the authority to list or update the CICS and LANGUAGE segments in the user profiles. Listing or updating these segments is done by creating profiles in the RACF FIELD class, of the form shown in “FIELD resource class” on page 32.

If you want to change the opclass but you do not want to respecify the list, you can use the ADDOPCLASS and DELOPCLASS operands. For example:

```
ALTUSER userid
        CICS(ADDOPCLASS (1,2)
             DELOPCLASS (6,7))
```

RACF group profiles

In addition to defining individual user profiles in RACF, you can define **group profiles**.

A group profile defines a group of **users**. (This is not the same thing as a resource group profile, which defines a group of **resources** and is explained in “General resource profiles” on page 27.) A group profile can contain information about the group, such as who owns it; what subgroups it has; a list of connected users; and other information.

Users who are members of groups can share common access authorities to protected resources. For example, you might want to set up groups as follows:

- Users who work in the same department

- Users who work with the same sets of transactions, files, terminals, or other resources that you choose to protect with RACF
- Users who sign on to the same regions (if you have more than one CICS region).

In a CICS environment, group profiles offer a number of advantages:

- Easier control of access to resources
- The ability to assign authorities using the group-SPECIAL attribute or CONNECT group authority
- A reduction in the number of refreshes to in-storage profiles.

Aim to make your point of control the presence (or absence) of a userid within a group, not the access list of the resource profile. When someone leaves a department, simply removing the userid from the department's user group revokes all privileges. No other administration of profiles is required. Doing this keeps RACF administration to a minimum and avoids having an excessive number of resource profiles.

For details of how to define and use group profiles, see the *RACF Security Administrator's Guide*.

CICS maintains in-storage copies of resource profiles, so changes to those profiles do not take effect on the system until the in-storage profiles are refreshed.

To avoid the need to refresh the resource profiles, place *RACF groups* instead of userids on access lists. For more information see "Refreshing resource profiles in main storage" on page 29. Then, any user with CONNECT group authority in that group can change the membership of the group (using the CONNECT and REMOVE commands). This avoids the need to change the access list of the affected profiles (through the use of the PERMIT command). If you do not actually change a CICS general resource profile, you need not refresh its in-storage copy. However, users may need to sign on again, if their group membership has been changed.

For other benefits obtained from creating groups, see the *RACF Security Administrator's Guide*.

For example, the following command sequence creates a new group of users and moves a user from an existing group to the new group:

```
ADDGROUP  group_name2
REMOVE  user1  GROUP(group_name1)
CONNECT  user1  GROUP(group_name2)
```

| Note that in an ISC or MRO environment, the interval that elapses before a *remote* userid is deleted is determined by the CICS system initialization parameter USRDELAY, which specifies how long an unused userid can remain signed on. (This can be up to 7 days.) For information about specifying USRDELAY, see the *CICS/ESA System Definition Guide*.

Data set profiles

Using RACF facilities, you can protect data sets on direct access storage devices (DASD) and tapes. You do this by defining data set profiles for the data sets you want to protect. The rules for defining data set profiles to RACF are described in the *RACF Security Administrator's Guide* and the *RACF Command Language Reference* manual. For examples, see the *RACF General User's Guide*.

You define profiles to protect two RACF categories of data sets:

1. Profiles for **user data sets**, where the high-level qualifier is a RACF userid. All RACF-defined users can protect their own data sets.
2. Profiles for **group data sets**, where the high-level qualifier is a RACF group name (see "RACF group profiles" on page 18 for information about RACF groups). A RACF-defined user can RACF-protect group data sets provided the user has the necessary authority or attributes. (See the *RACF Security Administrator's Guide* for details.)

| **Note:** Data set profiles do not apply to CICS terminal users, only to the CICS
| region userid.

Generic data set profiles

By using generic profiles, you can reduce the number of profiles needed to protect data sets, and also reduce the size of the RACF database. In addition, generic profiles are not volume-specific (that is, data sets protected by a generic profile can reside on any volume).

Usually, you specify generic data set profile names by specifying a generic character; for example percent (%) or asterisk (*) in the profile name. For data set profiles, RACF distinguishes between asterisk (*) and double asterisk (**) if RACF's enhanced generic naming is in effect. See the *RACF Command Language Reference* manual for the rules governing the specification of generic profile names in the RACF DATASET class.

For example, if you have a group called CICS410, you can define a generic profile named 'CICS410.**', and any user in the access list of this profile can access, at the authorized level, data sets with the high-level qualifier CICS410. For example:

```
ADDSD 'CICS410.**' UACC(NONE) NOTIFY(admin_userid)
```

| You must use the SETROPTS GENERIC command before defining generic
| profiles, as described in "Brief summary of RACF commands" on page 21.

Note: Examples in this book show double asterisks (**), which require that enhanced generic naming be in effect. If enhanced generic naming is not in effect, use single asterisks (*) in place of the double asterisks. (Enhanced generic naming is put into effect by issuing the RACF SETROPTS EGN command. Note that SETROPTS EGN only affects data set names. Enhanced generic naming is always in effect for general resource profiles, such as TCICSTRN.)

Brief summary of RACF commands

Much of the RACF activity dealing with protected CICS resources involves creating, changing, and deleting *general resource profiles*.

Creating a general resource profile

To create a general resource profile, use the RDEFINE command. Generally, once you have created a profile, you then create an access list for the profile using the PERMIT command. For example:

```
RDEFINE  class_name  profile_name  UACC(NONE)
PERMIT  profile_name  CLASS(class_name)
        ID(user or group)  ACCESS(access_authority)
```

This book provides many examples of how to do this for specific CICS-related classes.

Removing user or group entries

To remove the entry for a user or group from an access list, issue the PERMIT command with the DELETE operand instead of the ACCESS operand:

```
PERMIT  profile_name  CLASS(class_name)
        ID(user or group)  DELETE
```

Changing a profile

If you want to change a profile (for example, changing UACC from NONE to READ) use the RALTER command:

```
RALTER  class_name  profile_name  UACC(READ)
```

Deleting a profile

To delete a resource profile, use the RDELETE command. For example:

```
RDELETE class_name  profile_name
```

Copying from profiles

You can copy an access list from one profile to another. To do so, specify the FROM operand on the PERMIT command:

```
PERMIT  profile_name  CLASS(class_name)
        FROM(existing_profile_name) FCLASS(class_name)
```

You can copy information from one profile to another. To do so, specify the FROM operand on the RDEFINE or RALTER command:

```
RDEFINE class_name profile_name
        FROM(existing-profile_name) FCLASS(class_name)
```

Note: Do not plan to do this if you are using resource group profiles. RACF does not copy the members (specified with the ADDMEM operand) when copying the profile. Also, there are other ways in which the new profile might not be an exact copy of the existing profile. For example, RACF places the userid of the resource profile owner in the access list with ALTER access authority. For complete information, see the description of the FROM operand on the appropriate commands in the *RACF Command Language Reference* manual.

Listing profiles

To list the names of profiles in a particular class, use the SEARCH command. The following command lists profiles in the TCICSTRN class:

```
SEARCH  CLASS(TCICSTRN)
```

The following command lists all profiles and their details in the GCICSTRN class:

```
SEARCH  CLASS(GCICSTRN)
RLIST GCICSTRN * ALL
```

For information on resource classes, see "General resource profiles" on page 27.

Group-SPECIAL users

If you are a group-SPECIAL user (not system-SPECIAL), the SEARCH command might not list all the profiles that exist in a class. To get a complete list of profiles in a class, you must have at least the authority to list each profile. For further information, see the description of RACF requirements for the SEARCH command in the *RACF Command Language Reference* manual, and "Which profile is used to protect the resource?" on page 251.

Activating protection

To begin protecting all the resources protected by profiles in a RACF class, you need to activate that class by issuing the SETROPTS command with CLASSACT specified:

```
SETROPTS CLASSACT(class_name)
```

Defining generic profiles

Before you can define a generic profile (that is, one that uses an asterisk (*), double asterisk (**), ampersand (&), or percentage (%) character), you must issue the command:

```
SETROPTS GENERIC(class_name)
```

Deactivating a class

Deactivating a class turns off protection without disturbing the profiles themselves. If a class is deactivated, RACF issues a "not protected" return code to CICS for *all resources* in that class. To deactivate a RACF class, issue the SETROPTS command with NOCLASSACT specified:

```
SETROPTS NOCLASSACT(class_name)
```

Notes:

1. For classes whose profiles are placed in storage by CICS, these commands do not take effect until the next time the PERFORM SECURITY REBUILD command is issued, or the CICS region is restarted. For information on the PERFORM SECURITY REBUILD command, see "Refreshing resource profiles in main storage" on page 29.
2. Do not issue SETROPTS NOCLASSACT for classes that CICS is currently using. If you do this, PERFORM SECURITY REBUILD will fail if CICS is currently running. If the class requested by the X**name** system initialization parameter is inactive during CICS initialization, CICS will fail.

Determining active classes

To determine which RACF classes are currently active, issue the SETROPTS command with LIST specified:

```
SETROPTS LIST
```

Security classification of data and users

RACF provides the means to classify some or all of the resources on your system. You can use security levels, security categories, or both, to protect all CICS-related resources.

Consider classifying resources if you want to control access to them without having to specify access lists in each resource profile. If you classify a resource, only users whose user profiles are appropriately classified will be able to access that resource. For information on using security levels and security categories, see the *RACF Security Administrator's Guide*. Because CICS uses the RACROUTE REQUEST=FASTAUTH function, some services such as security labels, and global access checking, are not available under CICS. See the *RACF Security Administrator's Guide* for information on what is available with FASTAUTH.

You can also put users with the same access or logging requirements into groups. A user can be defined as belonging to one or more groups, one of which is their default. The signon process allows the user to override the default RACF user group name if the list of groups checking is inactive.

Defining port of entry profiles

Port of entry is the generic term for the device at which the end user signs on. For CICS, the port of entry can be either a terminal or a console.

Terminal profiles

This section briefly describes some aspects of terminal profiles that are of interest to CICS users. For more detailed information about defining and protecting terminals on MVS systems, particularly on the following topics, see the *RACF Security Administrator's Guide*:

- Creating profiles in the TERMINAL and GTERMINL classes
- Preventing the use of undefined terminals
- Restricting specific groups of users to specific terminals
- Restricting the days or times when a terminal can be used
- Using security labels to control terminals.

You can control user access to terminals by defining them to RACF. (User access is determined at CICS sign-on time.) RACF supports two IBM-supplied resource class names for terminals:

TERMINAL For defining profiles of *individual* terminals.

GTERMINL For defining profiles of *groups* of terminals.

Note: For GTERMINL profiles, RACF always uses in-storage profiles, which must be manually refreshed. Every time you create, change, or delete a GTERMINL profile, you (or the RACF security

administrator) must issue a SETROPTS RACLIST(TERMINAL) REFRESH command for the change to take effect.

Defining profiles of individual terminals

To define terminals with NETNAMEs netid1, netid2, and netid3 in the TERMINAL resource class, use the command:

```
RDEFINE  TERMINAL (netid1, netid2, netid3) UACC(NONE)
                  NOTIFY(sys_admin_userid)
```

If the terminal IDs start with the same characters, you can create a generic profile to cover a group of terminals with the same initial characters. You must use the SETROPTS GENERIC command before defining generic profiles, as described in "Brief summary of RACF commands" on page 21. This reduces the amount of effort needed to create the access list. For example:

```
RDEFINE  TERMINAL  netid*  UACC(NONE)
         NOTIFY(sys_admin_userid)
PERMIT netid* CLASS(TERMINAL)
       ID(group1, group2,.., groupn) ACCESS(READ)
```

Defining profiles of groups of profiles

Alternatively, you could define the same terminals in the resource group class, by including them as members of a suitable terminal group. For example:

```
RDEFINE  GTERMINL term_groupid
         ADDMEM(netid1, netid2, netid3) UACC(NONE)
         NOTIFY(sys_admin_userid)
```

To remove a terminal from a resource group profile, specify the DELMEM operand on the RALTER command. For example:

```
RALTER  GTERMINL term_groupid
        DELMEM(netid3)
```

To allow a group of users in a particular department to have access to these terminals, use the PERMIT command as follows:

```
PERMIT term_groupid CLASS(GTERMINL) ID(dept_groupid) ACCESS(READ)
```

Profiles in the TERMINAL and GTERMINAL classes

For CICS terminals, the terminal profiles to define to RACF in the TERMINAL or GTERMINL class are:

- **For VTAM terminals**, the value of the NETNAME that is specified in the RDO terminal definition.
- **For all other terminals**, the 4-character CICS terminal ID, taken from the DFHTCT macro.

When CICS is signing on a default userid, a link userid, or an attach-time userid from a remote system (see Table 1 on page 13), no terminal userid checking is done for LU6.2. Terminal userid checking is done for MRO. No check is made as to whether that userid is authorized to use that terminal. For more information, see "Link security with LU6.2" on page 145, "Link security with LU6.1" on page 183, or "Link security with MRO" on page 192, according to the environment you are using.

Universal access authority for undefined terminals

RACF supports a universal access facility for undefined terminals, which you can control by means of the SETROPTS TERMINAL command (provided you have the necessary authorization). When SETROPTS TERMINAL(NONE|READ) is in effect, it affects *all* MVS terminal subsystems.

If SETROPTS TERMINAL(READ) is in effect, RACF allows any user to log on at any undefined terminal (that is, a terminal not defined in the TERMINAL or GTERMINL resource classes). If SETROPTS TERMINAL(NONE) is in effect, RACF does not allow anyone to log on at any undefined terminals.

Note: Before issuing the SETROPTS TERMINAL(NONE) command, you must define some TERMINAL or GTERMINL class profiles, with enough authorizations to ensure that at least some of the terminals can be used, otherwise no one will be able to access any terminal.

Overriding the SETROPTS TERMINAL command

You can override the SETROPTS TERMINAL command at the group level by specifying the TERMUACC or NOTERMUACC options on the ADDGROUP or ALTGROUP command. The effect of the TERMUACC parameter is to enforce the universal access option. For example, if SETROPTS TERMINAL(READ) is active, the TERMUACC option means that all users in the group can access any undefined terminal. On the other hand, if you specify NOTERMUACC for the group, the SETROPTS TERMINAL command has no effect for that group, and all the users in the group need explicit authorization to use a terminal. To enable groups with the NOTERMUACC option to access terminals, you must add their group userids to the access list of the appropriate TERMINAL or GTERMINL profiles.

Console profiles

If the CONSOLE class has been activated you can control whether:

- A user is allowed to sign on to a console
- CICS is allowed to sign on a userid for a console defined with preset security.

Console protection is implemented in a similar method to that for protecting terminals, with the exception of the following, which were discussed in "Universal access authority for undefined terminals":

1. The SETROPTS TERMINAL command does not apply to consoles
2. The TERMUACC group attribute does not apply to consoles

Before activating the CONSOLE class, check the *MVS/ESA Planning: Operations* manual for the effects of console protection on MVS consoles.

Define the profile in the console class using either the console name or number, according to your standards for defining consoles. For example, using a console name:

```
RDEFINE CONSOLE CICSCONS UACC(NONE)
```

APAR PN91900

MJO 30/1/97

Note that the name given to the console is equal to the 3-digit number ID if it is not defined to MVS. Therefore the name always exists. When using preset security with consoles the user doing the install should have READ access to the console name. For MVS 4.1 and later the user must have READ access to the console-name to signon at a console. (Users with a release earlier than MVS 4.1 must have READ access to the console ID once the console class has been activated). The following example shows how user CICSOPR would sign on the console named CONCICS1:

```
RDEFINE CONSOLE CONCICS1 UACC(NONE)
PERMIT CONCICS1 CLASS(CONSOLE) ID(CICSOPR) ACCESS(READ)
```

Note that unlike the case with TERMINAL protection, a signon attempt will fail if made at a console that has not been defined in the activated CONSOLE class. The access authority to undefined consoles is NONE.

Conditional access processing

RACF offers capabilities of increasing a user's authority to resources if that user is signed on at a particular terminal or console. This is called ***conditional access*** processing.

You grant conditional access to a resource by adding

```
WHEN(TERMINAL(netname))
```

or

```
WHEN(CONSOLE(console name))
```

to the PERMIT command.

The following example allows members of the PAYROLL group to read the SALARY file wherever they are signed on. They would be able to update it only from the terminal with netname PAY001, by issuing the following commands:

```
RDEFINE FCICSFCT SALARY UACC(NONE)
PERMIT SALARY CLASS(FCICSFCT) ID(PAYROLL) ACCESS(READ)
PERMIT SALARY CLASS(FCICSFCT) ID(PAYROLL)
       (WHEN(TERMINAL(PAY001)) ACCESS(UPDATE)
```

To allow members of the operations group OPS to be able to use the CEMT transaction only from the console names MVS1MAST, the following command should be issued:

```
RDEFINE TCICSTRN CEMT UACC(NONE)
PERMIT CEMT CLASS(TCICSTRN) ID(OPS) WHEN(CONSOLE(MVS1MAST)) AC(READ)
```

Notes:

1. The CONSOLE class must be active before CONSOLE conditional access lists can be used.
2. Conditional access lists may only increase authority and not decrease it.

   For other considerations on conditional access lists see the *RACF Security Administrator's Guide*.

General resource profiles

RACF and CICS have default names for each matching class of resource. These defaults match for the corresponding releases of CICS and RACF. These classes are described in Table 3 on page 28.

RACF resource class names

For each resource class that is unique to CICS, there are two resource class names defined to RACF. The first of these is the name of the *member* class in which you define profiles whose names match the names of the resources, such as CICS transactions, programs, or DL/I PSBs. For profiles in this class, you define an access list for each individual resource name. In the following example, the RDEFINE commands define three profiles named CEMT, CEDA, and CEDB in the TCICSTRN resource class. The PERMIT commands allow one or more users or groups of users to access the CEMT transaction:

```
RDEFINE  TCICSTRN CEMT UACC(NONE)
         NOTIFY(sys_admin_userid)
RDEFINE  TCICSTRN CEDA UACC(NONE)
         NOTIFY(sys_admin_userid)
RDEFINE  TCICSTRN CEDB UACC(NONE)
         NOTIFY(sys_admin_userid)
PERMIT   CEMT CLASS(TCICSTRN) ID(group1, group2) ACCESS(READ)
PERMIT   CEDA CLASS(TCICSTRN) ID(group1, group2) ACCESS(READ)
PERMIT   CEDB CLASS(TCICSTRN) ID(group1, group2) ACCESS(READ)
```

The second class name is the RACF *resource group* class. To define a profile in a resource group class, use the RDEFINE command with the ADDMEM operand to add resources as members of the group. For example, you could define a profile named CICSTRANS in the GCICSTRN resource class, adding the CICS-supplied transactions (CEMT, CEDA, CEDB, CEDF, and so on) as members of the group. You then only need to specify an access list for the group, and not for each individual transaction, as in the following example for the CICSTRANS group profile:

```
RDEFINE  GCICSTRN CICSTRANS UACC(NONE)
         ADDMEM(CEMT, CEDA, CEDB)
         NOTIFY(sys_admin_userid)

PERMIT   CICSTRANS  CLASS(GCICSTRN) ID(group1, group2) ACCESS(READ)
```

By using the resource group profiles, you can reduce the number of profiles you need to maintain in the resource classes. Further, provided you avoid defining duplicate member names, using this method reduces the storage requirements for the RACF in-storage profiles that CICS builds during initialization.

RACF provides an in-storage checking service to avoid the I/O operations that would otherwise be needed in RACF. (It does this by means of the RACROUTE REQUEST=FASTAUTH macro.) For this purpose, CICS requests RACF to bring its resource profiles into main storage during CICS initialization.

To make administration easier, avoid defining duplicate profiles. If duplicates are encountered, RACF merges the profiles according to the ICHRLX02 selection exit. However, if no selection exit is installed, RACF follows the default merging rules as indicated in the RLX2P data area. For more information about this, see the *RACF Control Facility Data Areas* manual.

IBM-supplied resource class names for CICS

The IBM-supplied set of default resource names for use by CICS is held in the RACF class descriptor table (CDT). You can use resource classes defined by your installation. For more information, see “Defining your own resource class names for CICS resource classes” on page 34.)

Table 3. RACF default resource class names for CICS

Default class name	Description	Class
TCICSTRN	CICS transactions, normal attach security	Member
GCICSTRN	CICS transaction groups	Group
PCICSPSB	CICS PSBs	Member
QCICSPSB	CICS PSB groups	Group
ACICSPCT	CICS started transactions and the following EXEC CICS commands: COLLECT STATISTICS TRANSACTION, DISCARD TRANSACTION, and INQUIRE|SET TRANSACTION	Member
BCICSPCT	Groups for the above	Group
DCICSDCT	CICS transient data queues	Member
ECICSDCT	Groups for the above	Group
FCICSFCT	CICS files	Member
HCICSFCT	CICS file groups	Group
JCICSJCT	CICS journals	Member
KCICSJCT	CICS journal groups	Group
MCICSPPT	CICS programs	Member
NCICSPPT	CICS program groups	Group
SCICSTST	CICS temporary storage queue	Member
UCICSTST	CICS temporary storage queue groups	Group
CCICSCMD	EXEC CICS system commands, EXEC CICS FEPI system commands	Member
VCICSCMD	EXEC CICS system command groups EXEC CICS FEPI system command groups	Group

Note: Each default class name has been allocated a group or class category according to its initial character.

Refreshing resource profiles in main storage

With RACF Version 1.9.2

If you are using RACF Version 1.9.2 (or earlier), and you add, change, or delete a RACF profile in a CICS-related resource class, the change does not take effect until you issue a PERFORM SECURITY REBUILD command, or until you restart CICS. Either of these two events causes the CICS region to request RACF to build in-storage profiles using the RACROUTE REQUEST=LIST macro. Keeping the profiles in main storage saves I/O operations. While CICS is running, there are two methods you can use to issue the command to refresh the in-storage profiles:

- Using the CEMT master terminal command:

  ```
  CEMT PERFORM SECURITY REBUILD
  ```

- Executing a transaction that issues the CICS command:

  ```
  EXEC CICS PERFORM SECURITY REBUILD
  ```

Note: You must issue the PERFORM SECURITY REBUILD in each CICS region for which you want the change to take effect. Only the classes that the region uses are refreshed (as defined in the system initialization parameter X*name*).

The PERFORM SECURITY REBUILD command is complete when NORMAL is displayed on the CEMT user's terminal, or when the EXEC command returns control with no errors. If RACF is unable to process the PERFORM SECURITY REBUILD command (for example, due to a failure), CICS may terminate, according to the circumstances of the RACF failure. In this case, message DFHXS1106 is issued because an abend has occurred in RACF.

You should bear in mind that only one PERFORM SECURITY REBUILD command can be processed at a time within a system and that, while it is being processed, all transactions doing a security check must wait until it has completed. You should therefore not use PERFORM SECURITY REBUILD excessively in a production system. If a generic profile is changed you must also issue the SETROPTS GENERIC(class-name) REFRESH command, after PERFORM SECURITY REBUILD.

With RACF Version 2.1

If you are using RACF Version 2.1, you must refresh the classes defined in RACLIST by using the TSO command:

```
SETROPTS RACLIST(xxxxxxxx) REFRESH
```

where (xxxxxxxx) is the RACF class to be refreshed, for example TCICSTRN. A CEMT PERFORM SECURITY REBUILD command gives a response of "NOT REQUIRED"

Other IBM-supplied RACF resource class names affecting CICS

The following other IBM-supplied RACF resource class names affect CICS:

APPCLU The resource class in which you define profiles for verifying the identity of APPC partner logical units (LU6.2) during VTAM session establishment.

APPL The resource class in which you define profiles for controlling terminal users' access to VTAM applications, such as CICS.

CONSOLE The resource class used to define profiles for consoles.

FACILITY The resource class that includes profile definitions for controlling of library lookaside (LLA) libraries, MRO bindtime security, and shared data tables security.

FIELD The resource class that includes profile definitions for listing or updating the CICS and language segments in the user profiles, and the session segments in APPCLU profiles.

OPERCMDS The resource class that controls which operator commands CICS is authorized to issue.

PROPCNTL The resource class that controls userid propagation.

PTKDATA The resource class that includes PassTicket encryption keys.

RACFVARS The resource class that controls RACF variables.

RACGLIST The resource class for controlling the optimization RACLISTed classes.

STARTED The resource class that provides the userids for MVS started jobs.

SURROGAT The resource class that includes profiles for the following userids:

- preset
- default
- non-terminal
- PLTPI

Also used for transactions started without a terminal, and for controlling job submission.

TERMINAL The resource class used to define profiles for terminals.

VTAMAPPL The resource class in which you define profiles for controlling the userids that can open VTAM ACBs from non-APF authorized programs.

Unlike the IBM-supplied RACF resource classes provided for CICS, you cannot change the class names of these general resource classes. Two of them have CICS system initialization parameters — XAPPC for APPCLU and XUSER for SURROGAT profiles.

APPCLU resource class

Before you can use RACF to control which APPC (LU6.2) logical units can establish connections with each other, you need to know the NETID and the LU identifiers of each session partner. With this information, you can use the RDEFINE command to create two profiles in the APPCLU resource class for each LU6.2 pair, defining one profile on each MVS system. For example, on the local system use the command:

```
RDEFINE  APPCLU  netid1.luid1.luid2 UACC(NONE)
         SESSION(SESSKEY(password))
```

On the remote system use the command:

```
RDEFINE  APPCLU  netid2.luid2.luid1 UACC(NONE)
         SESSION(SESSKEY(password))
```

In these examples:

netid is the network id, as specified on the NETID parameter in the VTAM startup member (ATCSTRxx) of SYS1.VTAMLST. If the VTAM in the local system is different from that in the remote system, netid1 and netid2 are different.

luid1 and luid2 are the LU names of the partners. In each case, the first LU name is the local LU name and the second is the remote LU name.

Note: CICS does not use the CONVSEC parameter information of the RDEFINE command, although this can be specified in the session segment. The equivalent information is kept in the ATTACHSEC operand of the CONNECTION definition.

You have the following options when you specify the SESSION keyword:

- Specifying the session key (using the SESSKEY suboperand on the SESSION operand).
- Specifying the interval for which the session key will be in effect for LU-LU pairs controlled by the profile (using the INTERVAL suboperand on the SESSION operand).
- Specifying locking or unlocking the LU-LU pairs controlled by this profile using the LOCK and UNLOCK suboperands on the SESSION operand.

  You can use LOCK to prevent users using a link. If LOCK is in force, the relevant profile cannot be used, the session does not bind, and CICS issues message DFHZC4941.

Defining the session key in the profiles is optional for RACF, but the key must be supplied for CICS to make use of the profiles. The session key must be the same in both systems.

You can specify either an 8-character alphanumeric session key, or a 16-digit hexadecimal session key, which corresponds with the format of bind passwords in CICS. If the session keys at both ends of the link do not match, the link cannot be established.

You can also specify an interval after which the password expires, but be aware of the impact this may have on the users at the remote end of the link. If either password expires, the link cannot be established. Depending upon the auditing of the profile records, ICH415I messages may or may not not be written out. See “Defining bind-time security” on page 143. (CICS issues message DFHZC4942 to the CSNE destination when the password has expired.) You should therefore ensure that you are aware when a password interval is about to expire so that a links do not fail for this reason.

For a more detailed example of RDEFINE APPCLU, see the section on controlling VTAM LU6.2 binds in the *RACF Security Administrator's Guide*. See also “Example of defining an APPCLU profile” on page 142.

See Chapter 12, “Implementing LU6.2 security” on page 141 for information about implementing LU6.2 session security.

APPL resource class

RACF provides the APPL resource class for defining profiles of applications such as CICS. CICS always passes the generic APPLID of the originating region in MRO (for example, the TOR) with the RACROUTE REQUEST=VERIFY ENVIR=CREATE macro. There is no APPLID propagation between non-MRO regions. For information about VTAM generic resource, see “Controlling access to the CICS region” on page 50. By restricting the access lists for the APPL profiles you define, you can control which terminal users (RACF user IDs) can sign on in the various CICS regions. For example:

```
RDEFINE APPL applida UACC(NONE) NOTIFY(sys_admin_userid)
PERMIT  applida CLASS(APPL) ID(group1,..,groupn)  ACCESS(READ)
```

Note: An APPLID represents a CICS region. See “Controlling access to the CICS region” on page 50.

See the *RACF Security Administrator's Guide* for more information about controlling access to applications.

FACILITY resource class

If you are using the library lookaside (LLA) facility of MVS, you can control a program's ability to use the LLACOPY macro. You need to authorize CICS jobs to use this macro. You do this by giving each CICS job UPDATE authority to the CSVLLA data set resource in the FACILITY class for each LLA-controlled data set used by that job. For example:

```
RDEFINE FACILITY CSVLLA.lldataset UACC(NONE)  NOTIFY
PERMIT  CSVLLA.lladataset CLASS(FACILITY) ID(....)  ACCESS(UPDATE)
```

The FACILITY class is also used for MRO bind-time security. For more information about this, see “Bind-time security with MRO” on page 189, and “Bind security” on page 207, which discusses MRO bind-time security in connection with shared data tables.

FIELD resource class

Resources in the FIELD class control access to certain fields in the RACF database. By creating profiles in the RACF FIELD class, of the following form, you can permit listing or updating of the CICS or LANGUAGE segments in the user profiles, and of appropriate fields in partner-LU profiles.

```
USER.CICS.OPIDENT
USER.CICS.OPCLASSN
USER.CICS.OPPRTY
USER.CICS.TIMEOUT
USER.CICS.XRFSOFF
USER.LANGUAGE.USERNL1
USER.LANGUAGE.USERNL2
APPCLU.SESSION.SESSKEY
APPCLU.SESSION.KEYINTVL
APPCLU.SESSION.SLSFLAGS
```

Alternatively, you can set up a generic profile `USER.CICS.**`, to control access to all fields in the CICS segment. Before defining generic profiles you must use the SETROPTS GENERIC command, as described in “Brief summary of RACF commands” on page 21.

You need READ access to list these profiles, and UPDATE access to change them. For further guidance, see the section on field level access checking in the *RACF Security Administrator's Guide*.

OPERCMDS resource class

You use the OPERCMDS resource class to specify which operator commands CICS is authorized to issue; for example, commands in the command list table (CLT), and MODIFY network commands. This resource class also controls which console users are allowed to issue MODIFY commands directed to particular CICS regions. For more information, see the *RACF Security Administrator's Guide*.

PROPCNTL resource class

The PROPCNTL resource class is described in "Controlling userid propagation" on page 52.

PTKTDATA resource class

The PTKTDATA resource class holds the encryption key used for generating and validating PassTickets.

A profile is added for each APPLID that receives sign-ons with PassTickets. The format of the command to add profiles is:

```
RDEFINE PTKTDATA applid
        SSIGNON(KEYMASKED(password-key))
        KEYENCRYPTED(password-key))
```

This class is included in RACF 2.1 and can be added to RACF 1.9.2 by applying PTFs OY65281 and OY65283. For more information see *RACF Secured Sign-on SPE Information Package Version 1 Release 9.2*, SC23-3765.

RACFVARS resource class

RACF variables. In this class, profile names which start with an ampersand (&), act as RACF variables that can be specified in profile names in other RACF general resource classes.

RACGLIST resource class

Contains the resolved copies of globally RACLISTed profiles.

SURROGAT resource class

The SURROGAT resource class is used for CICS use of surrogate user validation and JES job submission. See "Surrogate job submission in a CICS environment" on page 52, "Using the SURROGAT resource class" on page 69, and Chapter 7, "Surrogate user security" on page 103.

STARTED resource class

Profiles can be defined in this class for each job, or group of jobs, that need to run under a unique userid.

TERMINALS resource class

The TERMINAL resource class is used to authorize the ability to signon at terminals.

VTAMAPPL resource class

The VTAMAPPL resource class controls which userids running non-APF-authorized programs can OPEN the VTAM ACB associated with the CICS address space (which runs as a VTAM application). You can use this resource class to prevent any user from impersonating a CICS region by opening a VTAM ACB with the APPLID of a CICS region.

For specific information, see “Controlling the opening of a CICS region’s VTAM ACB” on page 51.

Defining your own resource class names for CICS resource classes

You can define your own resource classes so that you have unique resource class names for each CICS region.

Benefits

Defining your own resource class names can have the following benefits.

Controlling access from other regions: You can prevent users running in one CICS region from accessing the resources of other CICS regions that have different class names specified. (You can also do this by using prefixing; see the description of the SECPRFX parameter in “Defining security-related system initialization parameters” on page 54).

Reducing your virtual storage requirements: If more than one CICS region uses the same installation-defined classes, then each of those regions will load the same profiles in those classes. The use of prefixing does not reduce the number of profiles that are loaded. If your installation is short of virtual storage, the use of installation-defined classes can help relieve the storage constraint. (Other ways to relieve storage constraints are to reduce the number of resource profiles by using resource group profiles, to use generic profiles (for example, C*), and to permit groups to have only profiles instead of userids.)

Note: If you are using RACF 2.1, prefixing reduces your virtual storage requirements. If you are using RACF 1.9.2 or earlier, you should use different classes.

Group administrator for each region: For each CICS region that has installation-defined classes, you can authorize a different group administrator to create profiles to be used by that region.

To get this benefit, you must define the installation-defined classes with a POSIT number other than 5 (the POSIT number of the IBM-supplied CICS classes). Then give the group administrator the CLAUTH (class authority) for at least one of those classes.

If you use unique resource classes for each CICS region, it is not necessary to use prefixing.

You must use the SETROPTS GENERIC command before defining generic profiles, as described in “Brief summary of RACF commands” on page 21.

With prefixing active, you can also assign different administrators without fear of conflict. To do this, create a generic profile in each class using the prefix as a high-level qualifier. For example:

```
RDEFINE  TCICSTRN   cics_region_id.**  UACC(NONE)
         OWNER(cics_region_administrator_userid)
```

The administrator specified as the OWNER of each such profile can create and maintain more specific profiles. The other administrators cannot do so.

Note: If you are running CICS with XRF, you should regard the active CICS and its alternate as one CICS system as far as RACF is concerned, and define the same resource class names to both the active and alternate CICS region.

Setting up installation-defined classes

To set up installation-defined classes, work with your RACF system programmer to add new class descriptors to the installation-defined part (module ICHRRCDE) of the RACF class descriptor table (CDT). For an example of how to add installation-defined classes to the CDT, see Chapter 17, “Customizing security processing” on page 213.

All installation-defined classes that are defined in the CDT must also be defined in the MVS router table. This is because the MVS router checks any class used in a router request to determine if it actually exists. If it does not, no request is sent to RACF. To define classes to the MVS router, add them to ICHRFR01, the user-modifiable portion of the MVS router table, as described in the *RACROUTE Macro Reference* manual. Also see “Specifying user-defined resources to RACF” on page 218.

When setting up installation defined classes, we recommend that you copy the IBM-supplied defaults from the CDT, an example of which is in the *RACF Macros and Interfaces* manual. You will then need to change the name, group or member name, POSIT number, and ID. See the description of the ICHERCDE macro in the *RACF Macros and Interfaces* manual for details of valid values for these operands. See the *RACF Macros and Interfaces* manual for information on creating installation-defined resource classes. For an example of how to add resource classes, see the IBM-supplied sample, DFH$RACF, which is in CICS410.SDFHSAMP.

For CICS resources, the first character of the resource class name is predefined by CICS, consistent with the default resource class names. You can define the second through eighth characters of the resource class name, but for ease of administration, we recommend that you specify the same characters for both the member and group class. The 7 characters specified for the member class are the part of the resource class name you define to CICS in the various X*name* parameters, except for the XAPPC and XUSER parameters that have no “name” option. You should avoid using the letters “CICS” in the second through fifth characters in any class names you define. RACF requires that at least one of the characters in the classname should be a national or numeric character.

Part 2. Implementing RACF protection for a single-region CICS

This part discusses the tasks necessary to implement security on a single-region CICS, regardless of where the task needs to be performed — either in the CICS environment or in the RACF environment. Where necessary, it refers you to other manuals in the CICS and RACF libraries for more detailed information about resource and security-related definitions.

- **Chapter 3, "CICS data set and system security" on page 39** deals with the protection of the MVS data sets that CICS requires — the program load libraries, and the CICS system data sets (such as the local and global catalogs, journal, auxiliary temporary storage, and transient data intrapartition data sets).
- **Chapter 4, "Verifying CICS users" on page 63** deals with all aspects of signon security, including the part played by the CICS segment.
- **Chapter 5, "Transaction security" on page 77** describes the security checks that CICS performs to verify that a user entering a transaction at a CICS terminal is authorized to use the transaction. This is known as transaction-attach security. It also explains the part played by the CICS **default userid**.
- **Chapter 6, "Resource security" on page 83** describes the purpose of the RESSEC and CMDSEC attributes on resource definitions, with reference to the appropriate CICS manuals. The purposes of the RACF user, group, profile, and resource class definitions are explained, together with examples illustrating how CICS and RACF together control access to resources.
- **Chapter 7, "Surrogate user security" on page 103** describes the surrogate user checking activity that CICS can perform using the RACF surrogate user facility. The application of surrogate user security checking where appropriate, and the necessary RACF definitions, are discussed. Some examples are also given.
- **Chapter 8, "CICS command security" on page 109** describes CICS command security for the system programming commands. These commands are available either through the CEMT master terminal transaction, or the CICS API. The purpose of the CMDSEC attribute on resource definitions is described.
- **Chapter 9, "Security checking using the QUERY SECURITY command" on page 117** describes security checking by the user application using the EXEC CICS QUERY SECURITY command, which enables application programs to request from RACF the level of access a user has to a particular resource. The application program can determine what action to take based on the CICS-value data area (CVDA) values that CICS returns.
- **Chapter 10, "Security for CICS-supplied transactions" on page 125** describes how you should protect the CICS-supplied transactions, both those that are for CICS internal use only (and cannot be invoked directly from a CICS terminal), and those provided explicitly for users at CICS terminals.

Chapter 3. CICS data set and system security

This chapter describes how you should protect the MVS data sets that CICS requires. It discusses the following:

CICS installation requirements for RACF

You can control access to the resources used by your CICS region (or regions) by using RACF facilities. The CICS libraries supplied on the distribution volume include the CICS modules you need to support external security management.

CICS-supplied RACF dynamic parse validation routines

To define CICS terminal operator data, you need to use the CICS-supplied RACF dynamic parse validation routines. You must install these routines in SYS1.CICS410.SDFHLINK, which should be made an APF-authorized library in your MVS linklist. (For more information, see the *CICS/ESA Installation Guide*.)

The routines, which must be installed on the MVS system that you use for RACF administration, are as follows:

DFHSNNFY CICS segment update notification routine

DFHSNPTO CICS segment TIMEOUT print formatting routine (RACF 2.1 only)

DFHSNVCL CICS segment OPCLASS keyword validation routine

DFHSNVID CICS segment OPIDENT keyword validation routine

DFHSNVPR CICS segment OPPRTY keyword validation routine

DFHSNVTO CICS segment TIMEOUT keyword validation routine.

Using RACF support in a multi-MVS environment

If you are operating a multi-MVS environment with shared DASD, which is probably the case if you are running CICS with XRF, you are likely to want the active and alternate CICS systems to have access to the same terminal user characteristics. You can enable this by having the active and alternate CICS systems share the same RACF database.

MVS program properties table considerations

For performance reasons you should consider making your CICS regions nonswappable, by specifying the NOSWAP option in the PPT statement of the SCHEDxx member of the SYS1.PARMLIB library. If your installation has an entry for the DFHSIP program in the MVS program properties table (PPT), you should ensure that the NOPASS option is not set for DFHSIP in the PPT statement of the SCHEDxx member of the SYS1.PARMLIB library, because if it is, this bypasses password and RACF authorization checking on data sets accessed by the CICS region. For more information about specifying CICS MVS PPT options, see the *CICS/ESA Installation Guide*.

Protecting CICS load libraries

Although, in general, CICS runs in unauthorized state, the CICS initialization program, DFHSIP, needs to run in authorized state for part of its execution. For this reason, the version of the DFHSIP module supplied on the distribution tape is link-edited with the "authorized" attribute (using the linkage-editor SETCODE AC(1) control statement), and is installed in CICS410.SDFHAUTH.

APAR 71213

6/10/95 MJO

This library must be defined to the operating system as APF-authorized.

APAR PN71213

6/10/95 MJO

To prevent unauthorized or accidental modification of CICS510.SDFHAUTH, this library should be RACF-protected. Without such protection, the integrity and security of your MVS system are at risk. To control the unauthorized start-up of a CICS system using DFHSIP, you should also consider implementing the following:

- If DFHSIP is in a library that has been placed in the MVS link list, protect DFHSIP with a profile in the PROGRAM resource class. Give READ access to this profile only to those users who are allowed to execute CICS.
- If DFHSIP has been placed in the link pack area (LPA), it cannot be protected by the PROGRAM resource class. You should, therefore, control the start-up of CICS by controlling the loading of any suffixed DFHSIT load module. Ensure that no DFHSIT load module is included in the LPA, then control the loading of DFHSIT by creating a generic DFHSIT profile in the program resource class. Give read access to this profile only to those users who are allowed to execute CICS.

You also need to RACF-protect SYS1.CICS410.SDFHLINK and SYS1.CICS410.SDFHLPA; and the other libraries (including CICS410.SDFHLOAD) that make up the STEPLIB and DFHRPL library concatenations.

See "Authorizing CICS procedures to run under RACF" on page 41 for more information about protecting CICS data sets and creating suitable data set security profiles.

Note: The source language of your application programs is sensitive; you should also consider having RACF protect the data sets containing it.

Protecting CICS system data sets

When you start a CICS region (either as a job or a started task) in an MVS environment that has RACF installed, the job or task is associated with a userid, (referred to as the **CICS region userid**). The authority associated with this userid determines what RACF-protected resources the CICS region can access.

Each CICS region, for either production or test use, should be subject to normal RACF data set protection based on the region userid under which the CICS region executes. You specify the region userid under which CICS executes in one of three ways:

As a started task:

- In the RACF started procedures table, ICHRIN03, when you start CICS as a started task using the MVS START command. (See "Authorizing CICS procedures to run under RACF.") However, do not assign the "trusted" or "privileged" attributes to CICS entries in the started procedures table. For more information, see the description of associating MVS started procedures with userids in the *System Programming Library: RACF* manual.

As a started job:

- In a STARTED general resource class profile, on the user parameter of the STDATA segment.

As a job:

- On the USER parameter of the JOB statement when you start CICS as a JOB.

To ensure proper differentiation between the authorizations for different CICS regions, you should run each with a unique region userid. For example, the userid under which you run the production CICS regions to process payroll and personnel applications should be the only CICS userid authorized to access production payroll and personnel data sets.

If you are using intercommunication, it is particularly important to use unique userids, unless you want to bypass link security checking by using equivalent systems. For more information, see "Link security with LU6.2" on page 145, "Link security with LU6.1" on page 183, or "Link security with MRO" on page 192, depending on the environment you are using.

Authorizing CICS procedures to run under RACF

You can invoke your CICS startup procedure to start CICS as a started task or a started job. RACF provides the ICHRIN03 procedure table for started tasks, and the STARTED general resource class for started jobs. These are discussed in the following topics:

Using the ICHRIN03 table for started tasks

If you run CICS as a started task, you must associate the cataloged procedure name with a suitably authorized RACF user through the RACF table, ICHRIN03. RACF supplies a default ICHRIN03 table, which you can modify. See the *System Programming Library: RACF* manual for more information about this table, and how you can add the default entry.

If your ICHRIN03 table contains the default entry, you need not update the table, but you must define a RACF user with the same name as the cataloged procedure.

If your ICHRIN03 does not contain the default entry (or you choose not to set the default entry), you must update the table with an entry that contains the cataloged procedure name and its associated RACF user. This RACF user need not have the same name as the cataloged procedure.

Whether your ICHRIN03 table contains the default entry or a specific entry you have defined, the RACF user identified through ICHRIN03 must have the necessary access authority to the data sets in the cataloged procedure.

For example, if you associate a cataloged procedure called DFHCICS with the RACF userid CICSR, the userid CICSR needs to have access to the CICS resources accessed by the task started by DFHCICS.

Using STARTED profiles for started Jobs

Using a single procedure to start all your CICS regions as started tasks limits you to a single CICS region userid, as defined in the RACF started task table, ICHRIN03.

Using the started job security support provided by RACF 2.1 removes this constraint, and allows you to use separate userids for each started job, even though they are all started from the same procedure. Alternatively, you can use generic profiles for groups of CICS regions that are to share the same userid—for example, for all regions of the same type, such as terminal-owning regions.

The support for started jobs is provided by the RACF STARTED general resource class, and its associated STDATA segment. You define profiles in this class for each job, or group of jobs, that needs to run under a unique userid.

You must ensure that the userids specified in STDATA segments are defined to RACF. You must also ensure that the userids are properly authorized to the data set profiles of the CICS regions that run under them.

Example of a generic profile for a multiple AORs: The following example shows how to define a generic profile for jobs that are to be started using a procedure called CICSTASK. In this example the job names begin with the letters CICSDA for a group of CICS application-owning regions:

```
RDEFINE  STARTED  (CICSTASK.CICSDA*)  STDATA( USER=(CICSDA##) )
```

When you issue the START command to start CICSTASK with a job name of, say, CICSDAA1, MVS passes the procedure name (CICSTASK), and the job name (CICSDAA1) in order to obtain the userid under which this CICS application-owning region is to run. In the example shown above, the CICS region userid is defined as CICSDA##, for all regions started under the generic profile CICSTASK.CICSDA*.

Example of a unique profile for a each region: The following example shows how to define a unique profile for jobs that are to be started using a procedure called CICSTASK, and each started job is to run under a unique CICS region userid:

```
RDEFINE  STARTED  (CICSTASK.CICSDAA2)  STDATA( USER=(CICSDAA2) )
```

When you issue the START command to start CICSTASK with the job name CICSDAA2, MVS passes the procedure name (CICSTASK), and the job name (CICSDAA1) in order to obtain the userid under which this CICS application-owning region is to run. In the example shown above, the CICS region userid is defined as CICSDAA2, the same as the APPLID.

Defining user profiles for CICS region userids

Before attempting to bring up a CICS region, you must ensure that the required userids are defined. These are the CICS region userid and the CICS default userid. If you are suitably authorized, you can define a RACF user profile for a CICS region by means of the ADDUSER command. For example, to define CICSR as a userid for a CICS region, you can enter the following RACF command from TSO:

```
ADDUSER CICSR  NAME(user-name) DFLTGRP(cics_region_group)
```

In this example, DFLTGRP has been specified, so the initial password is the DFLTGRP name. If DFLTGRP is not specified, the password is set by default to the name of the group to which the person issuing the ADDUSER command belongs. Alternatively, you can specify a password explicitly on the PASSWORD parameter of the ADDUSER command. See “Coding the USER parameter on the CICS JOB statement” for details about changing new userid passwords.

Do not assign the OPERATIONS attribute to CICS region userids. Doing so would allow the CICS region to access RACF-protected data sets for which no specific authorization has been performed. CICS region userids do not need the OPERATIONS attribute if the appropriate CONNECT or PERMIT commands have been issued. These commands authorize the CICS region userid for each CICS region to access only the specific data sets required by that region.

Coding the USER parameter on the CICS JOB statement

If you start CICS from a job, include the parameters USER= and PASSWORD= on the JOB statement. For example:

```
//CICSA   JOB ... ,USER=CICSR,PASSWORD=password
```

Note that when you define a new user to RACF, the password is automatically flagged as expired. For this reason, the first time you start CICS under a new userid you must change the PASSWORD parameter on the JOB statement. For example:

```
PASSWORD=(oldpassword,newpassword)
```

If you want to avoid specifying the password on the JOB statement, you can allow a surrogate user to submit the CICS job. A surrogate user is a RACF-defined user who is authorized to submit jobs on behalf of another user (the original user), without having to specify the original user’s password. Jobs submitted by a surrogate user execute with the identity of the original user. See “Surrogate job submission in a CICS environment” on page 52 for more information. The region

| userid must also have SURROGAT authority to use the default user, see
| Chapter 7, “Surrogate user security” on page 103.

Authorities required for CICS region userids

The CICS control program runs under the CICS region userid. Therefore, this userid needs access to all the resources that CICS itself, rather than application programs, needs to use. There are two types of these resources:

1. Resources external to CICS, such as disk files, the spool system, and the VTAM network.
2. Resources internal to CICS, such as system transactions and auxiliary userids.

Authorizing external resources: Like any other batch job, each CICS region must be able to access many external resources. The authority for CICS to access these resources is obtained from the CICS region userid. It doesn't matter which signed-on user requests CICS to perform the actions that access these resources. The external services are aware only that CICS is requesting them, under the region userid's authority.

The resources that you must give access to are:

- External disk data sets used by CICS

  CICS needs authority to open all the disk data sets that it uses. See “Authorizing access to CICS data sets” on page 47.

- External disk data sets used by application programs

  CICS needs authority to open all the disk data sets that your own application programs need. See “Authorizing access to user data sets” on page 49.

- VTAM applications

  You are advised to control the ability of *any* program to become a VTAM application. If you do this, CICS needs authority to open its VTAM ACB. See “Controlling the opening of a CICS region's VTAM ACB” on page 51.

- Jobs submitted to the internal reader

  If any of your applications submit JCL to the JES internal reader, you should prevent CICS from allowing them to be submitted without the USERID parameter. See “Controlling userid propagation” on page 52.

  However, you should not usually require your applications to provide a PASSWORD parameter on submitted jobs. So you *should* allow CICS to be a surrogate user of all the possible userids that may be submitted. See “Surrogate job submission in a CICS environment” on page 52.

- System spool data sets

  CICS needs authority to access data sets in the JES spool system. See “JES spool protection in a CICS environment” on page 53

CICS may need authority to use the library lookaside facility. See “Authorizing access with MVS library lookaside (LLA) facility” on page 49.

Authorizing internal resources: There are several internal functions in which CICS behaves like an application program, but is actually performing housekeeping functions that are not directly for any end user. The associated transactions execute under control of the CICS region userid, and because these transactions

access CICS internal resources, you must give the CICS region userid authority to access them. These are:

- CICS system transactions

  CICS needs authority to attach all the internal housekeeping transactions that it uses. See “Category 1 transactions” on page 126.

- Auxiliary userids

  If CICS surrogate user checking is specified with the XUSER system initialization parameter (the default) CICS needs authority to use certain additional userids. These are:

  – The default userid

    See “CICS default user” on page 103.

  – The userid used for transient data trigger transactions

    See “Transient data trigger-level transactions” on page 105.

  – The userid used for post-initialization processing (PLTPIUSR)

    See “Post-initialization processing” on page 103.

- Resources used by PLTPI programs

  If the PLTPIUSR system initialization parameter is omitted, the CICS region userid is used for all PLTPI programs. In this case, the CICS region userid must be given access to all the CICS resources that these programs use. See “PLT programs” on page 81.

Defining the default CICS userid to RACF

For each CICS region for which you specify SEC=YES, you must define a RACF user profile whose userid matches the value of the system initialization parameter, DFLTUSER. For example, if you specify DFLTUSER=NOTSIGND, you must define a RACF user profile named NOTSIGND.

If you do not specify a value for the DFLTUSER parameter, the CICS-supplied default userid is CICSUSER and you must define a RACF user profile named CICSUSER.

You must define a different default CICS userid for each CICS region if any of the following considerations apply:

- The default CICS userid requires different security attributes (such as membership in RACF groups).
- The default CICS userid requires different operator data (CICS segment of the RACF user profile).
- The default CICS userid requires a different default language (LANGUAGE segment of the RACF user profile).

To define a CICS default user with the system initialization parameter default name (CICSUSER), use the ADDUSER command with the CICS operand, as follows:

```
ADDUSER CICSUSER  DFLTGRP(group_id) NAME(user_name)
        OWNER(userid or group)
        PASSWORD(password)
        CICS(OPCLASS(1,2,...,n) OPIDENT(identifier) OPPRTY(priority)
             TIMEOUT(timeout_value)  XRFSOFF(xrf_sign-off_option))
```

We recommend that the security administrator always defines the password for default userids and started tasks, instead of allowing it to default.

We also recommend that each CICS region should use its own default user, which aids debugging. Set up a RACF default user group to keep the definitions similar.

If you have specified the system initialization parameter XUSER=YES (the default), you must authorize the CICS region userid to be a surrogate user of the default userid, for example:

```
PERMIT CICSUSER.DFHINSTL CLASS(SURROGAT) ID(cics_region_userid)
```

During startup, CICS "signs on" the default userid. If the default user signon fails (because, for example, the userid is not defined to RACF), CICS issues message DFHXS1104 and terminates CICS initialization.

When CICS successfully signs on a valid RACF userid as the default user, it establishes the terminal user data for the default user from one of the following sources:

- The CICS segment of the default user's RACF user profile
- Built-in CICS system default values.

See "Obtaining CICS-related data for a user" on page 72 for details of the sign-on process for obtaining CICS terminal operator data.

CICS assigns the security attributes of the default userid to all CICS terminals before any sign-on activity is initiated by the terminal user. The security attributes and terminal user data of the default user also apply to any terminals at which users do not sign on (using either the CICS-supplied CESN transaction or a user-written equivalent), unless the security has been explicitly preset by specifying a value for the USERID option in the terminal definition.

CICS also assigns the security attributes of the default userid to any "trigger level transactions" that are initiated for transient data queues without a USERID parameter. For specific recommendations on the set of transactions that the default userid should be authorized to use, see Chapter 10, "Security for CICS-supplied transactions" on page 125.

The default userid must give at least the minimum authorities that ought to be granted to any other terminal user. In particular:

- The default user must be given access to the region's APPLID. See "Controlling access to the CICS region" on page 50.
- The default user must be given access to the CICS-supplied transactions that are intended to be used by everybody. See the definitions in "Category 2 transactions" on page 127, especially those transactions that are recommended for inclusion in the ALLUSER example group of transactions.

Authorizing access to CICS data sets

When you have defined a region userid for your CICS job (or started task), you must permit that userid to access the CICS system data sets with the necessary authorization.

When authorizing access to CICS system data sets, you should allow for the following levels of access: READ, UPDATE, and CONTROL. You should also define data set profiles with UACC(NONE) to ensure that only CICS region userids can access those data sets. For information about the CICS region userid, see "Protecting CICS system data sets" on page 41.

For CICS load libraries and the partitioned data set (PDS) containing journal archiving JCL, you should only permit READ access. The following four data sets require CONTROL access.

- The temporary storage data set
- The transient data intrapartition data set
- The CAVM control data set (XRF)
- The CAVM message data set (XRF).

UPDATE access is required for all the remaining CICS data sets. Therefore, for CICS system data sets you need at least three generic profiles to restrict access to the appropriate level. This is summarized by the examples shown in Table 4.

Table 4. Summary of generic data set profiles

Required access level	**Type of CICS data sets protected**
READ	Load libraries and journal archive JCL
UPDATE	Auxiliary trace; transaction dump; journal; system definition; global catalog; local catalog; and restart
CONTROL	Temporary storage; intrapartition transient data; XRF message; and XRF control

If you use generic naming of the data set profiles, you can considerably reduce the number of profiles you need for your CICS regions. This policy is illustrated in the examples shown in Figure 1 on page 48 for a number of sample CICS regions.

You can issue the RACF commands shown in the examples from a TSO session, or execute the commands using the TSO terminal monitor program, IKJEFT01, in a batch job as illustrated in Figure 1. Alternatively, you can use the RACF-supplied ISPF panels. All of these methods enable you to create the necessary profiles and authorize each CICS region userid to access the data sets as appropriate for the corresponding CICS region.

```
//RACFDEF  JOB 'accounting information',
//             CLASS=A,MSGCLASS=A,MSGLEVEL=(1,1)
//DEFINE   EXEC PGM=IKJEFT01
//SYSPRINT DD SYSOUT=A
//SYSTSPRT DD SYSOUT=A
//SYSUDUMP DD SYSOUT=A
//SYSTSIN  DD *
ADDSD  'CICS410.SDFHLOAD'  NOTIFY(cics_sys_admin_id) UACC(NONE)        1
PERMIT 'CICS410.SDFHLOAD' ID(cics_id1,..., cics_group1,.,cics_groupn)
                           ACCESS(READ)
ADDSD  'CICS410.SDFHAUTH'  NOTIFY(cics_sys_admin_id) UACC(NONE)        1
PERMIT 'CICS410.SDFHAUTH' ID(cics_id1,.., cics_group1,.,cics_groupn)
                           ACCESS(READ)
ADDSD  'CICS410.DFHJPDS'  NOTIFY(cics_sys_admin_id) UACC(NONE)         1
PERMIT 'CICS410.DFHJPDS' ID(cics_id1,..., cics_group1,.,cics_groupn)
                          ACCESS(READ)
ADDSD  'CICS410.applid.**'  NOTIFY(cics_sys_admin_id) UACC(NONE)       2
PERMIT 'CICS410.applid.**'  ID(applid_userid) ACCESS(UPDATE)
ADDSD  'CICS410.applid.DFHXR*'  NOTIFY(cics_sys_admin_id) UACC(NONE)   3
PERMIT 'CICS410.applid.DFHXR*'  ID(applid_userid) ACCESS(CONTROL)
ADDSD  'CICS410.applid.DFHINTRA'  NOTIFY(cics_sys_admin_id) UACC(NONE) 3
PERMIT 'CICS410.applid.DFHINTRA'  ID(applid_userid) ACCESS(CONTROL)
ADDSD  'CICS410.applid.DFHTEMP'  NOTIFY(cics_sys_admin_id) UACC(NONE)  3
PERMIT 'CICS410.applid.DFHTEMP'  ID(applid_userid) ACCESS(CONTROL)
ADDSD  'CICS410.DFHCSD'           NOTIFY(cics_sys_admin_id) UACC(NONE) 4
PERMIT 'CICS410.DFHCSD' ID(cics_group1,..,cics_groupn) ACCESS(UPDATE)
/*
//
```

Figure 1. Example commands to authorize access to CICS data sets

Note: Data sets that need to be accessed in the same way by all CICS regions (for example, with READ or UPDATE access) should be protected by profiles that do *not* include an APPLID. For example, define the partitioned data sets that contain the CICS load modules and the journal archive skeletal JCL with profiles, so that all CICS region groups (or userids) have READ access.

You could also consider protecting all these data sets with one generic profile called 'CICS410.**'. However, you must strictly control who has read access to CICS410.SDFHAUTH, because it contains APF-authorized programs, and the profile protecting this data set *must* be defined with UACC(NONE). In these examples, all of the partitioned data sets are defined with UACC(NONE) and have an explicit access list.

| Although CICS modules exist in libraries SYS1.CICS410.SDFHLPA and
| SYS1.CICS410.SDFHLINK, no CICS region userid requires access to these
| libraries.

By establishing a naming convention for the data sets belonging to each region and one generic profile for each CICS region, with the CICS VTAM APPLID as one of the data set qualifiers, you can ensure that only one CICS region has access to the data sets. In the examples shown in Figure 1, all the names have a high-level qualifier of CICS410, but your installation will have its own naming conventions for you to follow.

CICS needs UPDATE access to all the data sets covered by these profiles. The CICS DD names for the data sets in this category are as follows:

DFHGCD Global catalog data set
DFHLCD Local catalog data set
DFHAUXT Auxiliary trace data set, A extent
DFHBUXT Auxiliary trace data set, B extent
DFHDMPA Transaction dump data set, A extent
DFHDMPB Transaction dump data set, B extent
DFHJnnA Journal data sets, A extents ('nn' is a number 01 through 99)
DFHJnnB Journal data sets, B extents ('nn' is a number 01 through 99)
DFHJ01X System log journal data set, emergency restart extent
DFHJACD Journal archive control data set
DFHRSD Restart data set

Note: The auxiliary trace data set, the transaction dump data set, and the MVS dump data set may contain sensitive information and should therefore be protected from unauthorized access.

CICS needs CONTROL access for the transient data intrapartition, temporary storage, and CICS availability manager (CAVM) data sets.

The CICS DD names for the data sets in this category are as follows:

DFHINTRA Transient data intrapartition data set
DFHTEMP Temporary storage data set
DFHXRCTL XRF control data set
DFHXRMSG XRF message data set

The CICS system definition data set (CSD) is protected by a discrete profile that all CICS groups have access to. This assumes that all the CICS regions are sharing a common CSD. If your CICS regions do not share a common CSD and each region has its own CSD, or if groups of regions share a CSD, you must define discrete or generic data set profiles as appropriate.

Authorizing access with MVS library lookaside (LLA) facility

If any of the load-module libraries in the DFHRPL concatenation is controlled by the MVS library lookaside (LLA) facility, the CICS region's userid must be authorized in *one* of the following ways:

- It must have UPDATE authority to the data set that contains the LLA module.
- It must have UPDATE authority in the FACILITY class to the resource CSVLLA. *datasetname*, where *datasetname* is the name of the library that contains the LLA module.

Authorizing access to user data sets

When you have defined the RACF userids for your CICS regions and given them access to the CICS system data sets, you must also permit the userids to access the CICS *application* data sets with the necessary authority. The following RACF commands permit the userid specified on the ID parameter to access some CICS user application data sets, with READ authority for the first two data sets, and UPDATE authority for the last two:

```
PERMIT 'CICS410.appl1.dataset1'  ID(user or group)  ACCESS(READ)
PERMIT 'CICS410.appl1.dataset2'  ID(user or group)  ACCESS(READ)
PERMIT 'CICS410.appl2.dataset3'  ID(user or group)  ACCESS(UPDATE)
PERMIT 'CICS410.appl2.dataset4'  ID(user or group)  ACCESS(UPDATE)
```

ACCESS(CONTROL) for VSAM entry-sequenced data sets (ESDS)

CICS file control uses control interval processing when opening a VSAM ESDS. This means that you must specify ACCESS(CONTROL) for all such data sets, otherwise the OPEN command fails with message DFHFC0966.

ACCESS(ALTER) for VSAM data sets when using BWO

In order to use backup while open (BWO) to back up VSAM data sets that are currently in use and are defined as BACKUPTYPE(DYNAMIC), the CICS region userid must have RACF ALTER authority to the data set or to the ICF catalog in which that data set is defined. If not, the OPEN command fails with message DFHFC5803. See the *CICS/ESA Recovery and Restart Guide* for guidance on using BWO.

Controlling access to the CICS region

You can restrict access by terminal users to specific CICS regions by defining CICS APPLID profiles in the RACF APPL class. For these purposes, the APPLID of a CICS region is the VTAM generic resource, if the GRNAME system initialization parameter is specified, or the XRF generic APPLID if XRF=YES is specified. Otherwise, it is the specific APPLID named in the APPLID system initialization parameter. If you define a profile in the APPL class for a CICS APPLID, or a generic profile that applies to one or more CICS APPLIDs with UACC(NONE), all terminal users trying to sign on to a CICS region must have explicit access to the profile that applies to that region's APPLID, either as an individual profile, or as a member of a group. For example:

```
RDEFINE APPL cics_region_applid UACC(NONE)  NOTIFY(sys_admin_userid)
```

For MRO only, the APPLID is propagated from the terminal-owning region to the other region that the user accesses. Therefore, you can force users to sign on through a TOR, by denying users access to any APPLID except that of the TOR.

Use the RACF PERMIT command to add authorized users to the access list of CICS APPL profiles. For example:

```
PERMIT cics_region_applid CLASS(APPL) ID(group1,..,groupn) ACCESS(READ)
```

permits all users defined in the listed groups to sign on to `cics_region_applid`.

The APPL class must be active for this protection to be in effect:

```
SETROPTS CLASSACT(APPL)
```

Also, for performance reasons, your installation should consider RACLISTing profiles in the APPL class:

```
SETROPTS RACLIST(APPL)
```

If the APPL class is already active, you must refresh the in-storage APPL profiles with the SETROPTS command:

```
SETROPTS RACLIST(APPL) REFRESH
```

Notes:

1. CICS always passes the APPLID to RACF when requesting RACF to perform user sign-on checks, and there is no mechanism within CICS to prevent this.
2. RACF treats undefined CICS APPLIDs as UACC(READ).
3. If the APPL class is active, and a profile exists for a CICS region in the APPL class, you must ensure that authorized remote CICS regions can sign on to a CICS region protected in this way.

Controlling the opening of a CICS region's VTAM ACB

You can control which users among those who are running non-APF-authorized programs can OPEN the VTAM ACB associated with a CICS address space (CICS region). This ensures that only authorized CICS regions can present themselves as VTAM applications providing services with this APPLID, thus preventing unauthorized users impersonating real CICS regions. (Note that the CICS region userid needs the OPEN access, not the issuer of the SET VTAM OPEN command.)

For each APPLID, create a VTAMAPPL profile, and give the CICS region userid READ access. For example:

```
RDEFINE VTAMAPPL applid UACC(NONE)  NOTIFY(userid)
PERMIT applid CLASS(VTAMAPPL) ID(cics_region_userid) ACCESS(READ)
```

The correct CICS APPLID to specify in the VTAMAPPL class is the specific APPLID, as specified in the CICS system initialization parameters. If you are using XRF (that is, if CICS is started with XRF=YES in effect), you must define two VTAMAPPL profiles — one each for both the active and alternate CICS region's **specific** APPLID (the second operand on the CICS APPLID startup option).

Note: If your alternate is on another MVS image, you must ensure that the RACF database is shared, or you must define the VTAMAPPL profiles in the other system's RACF database.

The VTAMAPPL class must be active and RACLISTed for this protection to be in effect:

```
SETROPTS CLASSACT(VTAMAPPL) RACLIST(VTAMAPPL)
```

If the VTAMAPPL class is already active, you must refresh the in-storage VTAMAPPL profiles with the SETROPTS command:

```
SETROPTS RACLIST(VTAMAPPL) REFRESH
```

Note: You (or the RACF security administrator) must issue the SETROPTS command to refresh these profiles. Issuing the CICS PERFORM SECURITY REBUILD command does not affect the VTAMAPPL class.

Controlling userid propagation

Jobs submitted from CICS to the JES internal reader without the USER operand specified on the JOB statement run under the CICS region's userid. These jobs have the access authorities of the CICS region itself, and so could potentially expose other data sets in the MVS system.

You (or the RACF security administrator) can prevent the CICS userid from being propagated to these batch jobs by defining a profile in the PROPCNTL class where the profile name is the CICS region's userid. For example, if the CICS region runs under userid CICS1, define a PROPCNTL profile named CICS1:

```
RDEFINE  PROPCNTL  CICS1
```

The PROPCNTL class must be active and RACLISTed for this protection to be in effect:

```
SETROPTS CLASSACT(PROPCNTL) RACLIST(PROPCNTL)
```

If the PROPCNTL class is already active, you must refresh the in-storage PROPCNTL profiles with the SETROPTS command:

```
SETROPTS RACLIST(PROPCNTL) REFRESH
```

You (or the RACF security administrator) must issue the SETROPTS command to refresh these profiles. Issuing the CICS PERFORM SECURITY REBUILD command does not affect the PROPCNTL class.

Note: If you use PROPCNTL for a CICS region and you are using CICS automatic journal archiving, you will to specify a userid in your archive JCL data set. If you are not using SURROGAT processing, you must also specify a password. (Archive JCL is in the data set pointed to by the DFHJPDS DD statement in your CICS JCL. The JCL member, in the DFHJPDS data set, for a particular journal is specified in the CICS DFHJCT table on the DFHJCT TYPE=ENTRY ARCHJCL=membername parameter.)

Surrogate job submission in a CICS environment

Batch jobs submitted by CICS can be allowed to run with a USER parameter other than the CICS region's userid, but without specifying the corresponding PASSWORD. This is called surrogate job submission. These jobs have the access authorities of the USER parameter actually specified on the JOB statement. If the PASSWORD parameter is specified on the JOB statement, surrogate processing does not occur.

You (or the RACF security administrator) can allow this by defining a profile in the SURROGAT class. For example, if the CICS region's userid is CICS1, and the job is to run for userid JOE, define a SURROGAT profile named JOE.SUBMIT:

```
RDEFINE  SURROGAT  JOE.SUBMIT  UACC(NONE)
         NOTIFY(JOE)
```

Further, you must permit the CICS region's userid to act as the surrogate to the profile just defined:

```
PERMIT JOE.SUBMIT CLASS(SURROGAT) ID(CICS1) ACCESS(READ)
```

The SURROGAT class must be active and RACLISTed for this protection to be in effect:

```
SETROPTS CLASSACT(SURROGAT) RACLIST(SURROGAT)
```

Warning

Any CICS user, whether signed on or not, is able to submit jobs that use the SURROGAT userid, if the CICS userid has authority for SURROGAT. If your installation is using transient data queues to submit jobs, you can control who is allowed to write to the transient data queue that goes to the internal reader. However, if your installation is using EXEC CICS SPOOLOPEN to submit jobs, you cannot control who can submit jobs (without writing an API global user exit program to screen the commands). CICS spool commands do no CICS resource or command checking.

An EXEC CICS ASSIGN USERID command can be used to find the userid of the user who triggered the application code. Application programmers can then provide code that edits a USER operand onto the JOB card destined for the internal reader.

For a complete description of surrogate job submission support, see the *RACF Security Administrator's Guide*.

Authorizing the CICS region userid as a surrogate user

When CICS performs surrogate user checking, the CICS region userid must be authorized as a surrogate. Authorization must be granted for the CICS region userid acting as a surrogate user for the following:

- The CICS default user
- The userid used for post-initialization processing (PLTPIUSR)
- The userid used for transient data trigger level transactions.

For more information about surrogate user checking, see Chapter 7, "Surrogate user security" on page 103.

JES spool protection in a CICS environment

Your installation can protect JES spool data sets with profiles in the JESSPOOL class. Spool files created by the SPOOLOPEN commands have the userid of the CICS region in their security tokens, not the userid of the person who issued the SPOOLOPEN command. Thus, the userid qualifier in the related JESSPOOL profiles is the CICS region's userid.

When using the SPOOLOPEN INPUT command, CICS checks that the first four characters of the APPLID correspond to the external writer name of the spool file. This checking is independent of any RACF checking that may also be done.

You should define ALTER access to the appropriate PROFILE in the JESSPOOL class when the JESSPOOL class is active on a SPOOLOPEN INPUT command, to prevent a NOTFND condition being returned.

Defining security-related system initialization parameters

There are several system initialization parameters that CICS provides for specifying your security requirements at the system level. These parameters are coded in the CICS system initialization table (SIT) or as system initialization overrides. For full details of system initialization parameters, see the *CICS/ESA System Definition Guide*.

SEC

You use the SEC system initialization parameter to specify the level of resource security management you want for your CICS region. There are two options:

YES
This means that the CICS external security interface will be initialized. Then control of CICS security is determined by the other security-related SIT options:

```
SECPRFX              XRFSOFF
DFLTUSER             XRFSTME
ESMEXITS             XCMD
SNSCOPE              XDCT
PSBCHK               XFCT
CMDSEC               XJCT
RESSEC               XPCT
PLTPIUSR             XPPT
PLTPISEC             XPSB
XAPPC                XTRAN
XUSER                XTST
```

NO
This means that there is no security checking of whether users are allowed to access CICS (and non-CICS) resources from this region, and signon cannot take place.

Note: With MRO bind-time security, even if you have specified SEC=NO the CICS region userid is still sent to the secondary system, and bind-time checking is carried out in the secondary system. See “Bind-time security with MRO” on page 189 for more information.

SECPRFX

This parameter is effective only if you also specify SEC=YES. You use the SECPRFX system initialization parameter to specify whether you want CICS to prefix the resource names that it passes to RACF for authorization. The prefix that CICS uses is the RACF userid under which the CICS region is running.

Prefixing is useful mainly when you have more than one CICS region. It enables you to prevent users on one CICS region from accessing the resources of a different CICS region that has a different prefix. For example, you might have one CICS region with the prefix CICSPROD and another with prefix CICSTEST. Users of the CICSTEST system would be able to use profiles that included the CICSTEST prefix, and users of the CICSPROD system would be able to use profiles that included the CICSPROD prefix. Users of both systems would be able to use resources protected by profiles that included CICS*.

There are two options on the SECPRFX parameter:

NO
CICS does not prefix the resource names in authorization requests that it passes to RACF from this CICS region.

YES
CICS prefixes the resource names with its RACF userid when passing authorization requests to RACF. The prefix corresponds to the CICS region userid.

To change these values you must employ an ICHRTX00 SAF preprocessing exit. For more information, see "Determining the userid of the CICS region" on page 218. For example, if a CICS job specifies USER=CICSREG on the JOB statement, and SECPRFX=YES is specified, you can define and allow access to the CICS master terminal transaction (CEMT) to RACF in the TCICSTRN resource class as follows:

```
RDEFINE  TCICSTRN  CICSREG.CEMT
         UACC(NONE)   NOTIFY(sys_admin_userid)
PERMIT  CICSREG.CEMT  CLASS(TCICSTRN)
        ID(groupid1,...,groupidn) ACCESS(READ)
```

You can also use a resource group profile in the GCICSTRN resource class. If you do, specify the prefix on the ADDMEM operand. The following shows CICSREG specified in a profile named CICSTRANS:

```
RDEFINE  GCICSTRN  CICSTRANS
         ADDMEM(CICSREG.CEMT)
         UACC(NONE)   NOTIFY(sys_admin_userid)
PERMIT  CICSTRANS  CLASS(GCICSTRN)
        ID(groupid1,...,groupidn) ACCESS(READ)
```

Note: If you protect a resource with a resource group profile, you should avoid protecting the same resource with another profile. If the profiles are different (for example, if they have different access lists), RACF merges the profiles for use during authorization checking. Not only can the merging have a performance impact, but it can be difficult to determine exactly which access authority applies to a particular user. (For more information, see the *RACF Security Administrator's Guide*.)

DFLTUSER

Specify a value for DFLTUSER to identify to CICS the name you have defined to RACF as the default userid. If you omit this parameter, the name defaults to CICSUSER. See "Defining the default CICS userid to RACF" on page 45.

ESMEXITS

Use ESMEXITS to specify whether you want CICS to pass installation data for use by the RACF installation exits. For more information on ESMEXITS, see Chapter 17, "Customizing security processing" on page 213.

SNSCOPE

SNSCOPE—the signon SCOPE—applies to all userids signing on by explicit signon request, for example the EXEC CICS SIGNON command or the CESN transaction. It is used to specify whether or not a userid can have more than one CICS session active at the same time.

The signon SCOPE is enforced with the MVS ENQ macro. The SNSCOPE values correspond to the STEP, SYSTEM and SYSTEMS levels of ENQ scoping. This means that only those CICS systems that specify exactly the same value for SNSCOPE can check the scope of each other.

SNSCOPE affects only users signing on at local terminals, or signing on after using CRTE transaction to connect to another system. For more information about using SNSCOPE, and the restrictions involved, see the *CICS/ESA System Definition Guide*.

PSBCHK

Code PSBCHK to specify that you want CICS to perform PSB authorization checks for remote terminal users who use transaction routing to initiate a transaction in this CICS region (to access an attached IMS system). The default PSBCHK=NO specifies that the remote link is checked, but no check is made against the remote user. The remote user is checked by specifying PSBCHK=YES.

CMDSEC

Code CMDSEC to specify whether or not you want CICS to honor the CMDSEC option specified on a transaction's resource definition. CMDSEC specified with the option ASIS means that CICS obeys the CMDSEC option. CMDSEC Specified with the option ALWAYS means that CICS ignores the CMDSEC option, and always performs the command check. For more information about these options, see *CICS/ESA System Definition Guide*.

PLTPIUSR

Code PLTPIUSR to specify the userid that CICS is to use for security checking for PLT programs that run during CICS initialization.

PLTPISEC

Code PLTPISEC to specify whether or not you want CICS to perform command security or resource security checking for PLT programs that run during CICS initialization.

RESSEC

Code this to specify whether or not you want CICS to honor the RESSEC option specified on a transaction's resource definition. RESSEC specified with the option ASIS means that CICS obeys the RESSEC option. Specified with the option ALWAYS means that CICS ignores the RESSEC option, and always performs the resource check. For more information about these options, see *CICS/ESA System Definition Guide*.

CICS resource class system initialization parameters

You specify at the system level (with the SEC=YES parameter) that you want CICS to use RACF to authorize access to CICS resources. You also specify at the system level which particular CICS resources you want CICS to check by means of the appropriate resource class parameters. These are the X*name* system initialization parameters. The full list of the CICS resource classes is shown in Table 5 on page 57, with the corresponding X*name* system initialization parameter.

Table 5. System initialization parameters for the CICS resource classes	
System initialization parameter	**Resource**
XAPPC={**NO**|YES}	APPC partner-LU verification
XCMD={**YES**|name|NO}	EXEC CICS system commands EXEC CICS FEPI system commands
XDCT={**YES**|name|NO}	Transient data destinations
XFCT={**YES**|name|NO}	Files
XJCT={**YES**|name|NO}	Journals
XPCT={**YES**|name|NO}	Started transactions and EXEC CICS commands: COLLECT STATISTICS TRANSACTION DISCARD TRANSACTION INQUIRE TRANSACTION and SET TRANSACTION
XPPT={**YES**|name|NO}	Programs
XPSB={**YES**|name|NO}	DL/I program specification blocks (PSBs)
XTRAN={**YES**|name|NO}	Attached transactions
XTST={**YES**|name|NO}	Temporary storage entries
XUSER={**YES**|NO}	Surrogate user checking

Notes:

1. The parameters are effective only with SEC=YES.
2. None of the parameters can be entered as console overrides.

If you specify YES for any X*name* system initialization parameters, CICS uses the default class names. (See "IBM-supplied resource class names for CICS" on page 28.)

As an example, the effect of specifying SEC=YES with three of the resource class parameters specified as X*name*=YES is illustrated in the following table:

Table 6. Specifying external security with default resource classes	
System initialization parameter	**Effect**
SEC=YES	Initializes CICS external security interface.
XTRAN=YES	CICS uses the TCICSTRN and GCICSTRN resource class profiles for transaction-attach security checking.
XFCT=YES	CICS uses the FCICSFCT and HCICSFCT resource class profiles for file access security checking.
XPSB=YES	CICS uses the PCICSPSB and QCICSPSB resource class profiles for PSB access security checking.

As an example, the effect of specifying SEC=YES with three associated resource class parameters specified as X*name*=*username* is shown in Table 7 on page 58.

Table 7. Specifying external security for user-defined resource classes	
System initialization parameter	**Effect**
SEC=YES	CICS uses full RACF security support.
XTRAN=$usrtrn	CICS uses the T$usrtrn and G$usrtrn user-defined resource class profiles for transaction-attach security checking.
XFCT=$usrfct	CICS uses the F$usrfct and H$usrfct user-defined resource class profiles for file access security checking.
XPSB=$usrpsb	CICS uses the P$usrpsb and Q$usrpsb user-defined resource class profiles for PSB access security checking.

When CICS is being initialized, it requests RACF to bring resource profiles into main storage to match all the resource classes that you specify on system initialization parameters. Note that (except for XAPPC) X*name*=YES is the default in the system initialization parameters, and will use the CICS default classnames, for example, GCICSTRN. You must supply RACF profiles for all those resources for which you do not specify X*name*=NO explicitly. If CICS requests RACF to load a general resource class that does not exist or is not correctly defined, CICS issues a message indicating that external security initialization has failed, and terminates CICS initialization.

For guidance on the syntax of external security system initialization parameters, see the *CICS/ESA System Definition Guide*.

The way you define the individual transaction definitions in the CSD determines whether you want to use RACF security for the resources and commands used with transactions. See Chapter 4, “Verifying CICS users” on page 63 and Chapter 5, “Transaction security” on page 77 for information about specifying resource and command security for transactions.

XAPPC and XUSER

The syntax of the XAPPC and XUSER system initialization parameters is slightly different from that of the other X*name* parameters. You can only specify YES or NO. XAPPC=YES indicates that you want session security for APPC sessions. CICS is not initialized if YES is specified and the APPCLU class is not defined to RACF. For more information on what happens in these circumstances, see “CICS/ESA initialization failures related to security” on page 257.

XAPPC enables RACF LU6.2 bind-time (also known as APPC) security. For more information, see “Bind-time security with LU6.2” on page 141.

For more information on the APPCLU class, see “APPCLU resource class” on page 30.

XUSER activates surrogate user security. For more information, see Chapter 7, “Surrogate user security” on page 103.

Using IBM-supplied classes without prefixing

To set up external security for transactions, files, and PSBs, using IBM-supplied resource classes and without prefixing, take the steps described in this section.

Before you define a profile, you must activate the relevant classes, using the SETROPTS CLASSACT and SETROPTS GENERIC commands, as described in "Brief summary of RACF commands" on page 21.

We recommend that you work in a test region first, to ensure the least interruption to actual business processes.

1. Plan and create RACF profiles in the relevant classes:

```
RDEFINE  TCICSTRN  transaction-name  UACC(NONE)  NOTIFY(userid)
RDEFINE  FCICSFCT  file-name         UACC(NONE)  NOTIFY(userid)
RDEFINE  PCICSPSB  PSB-name          UACC(NONE)  NOTIFY(userid)
```

2. Permit appropriate users or groups (preferably groups) to have access to the profiles:

```
PERMIT  transaction-name  CLASS(TCICSTRN)  ACCESS(READ)
        ID(userid or groupid)
PERMIT  file-name         CLASS(FCICSFCT)  ACCESS(READ)
        ID(userid or groupid)
PERMIT  PSB-name          CLASS(PCICSPSB)  ACCESS(READ)
        ID(userid or groupid)
```

3. Specify the following CICS system initialization parameters:

```
SEC=YES           XTRAN=YES        XCMD=NO
SECPRFX=NO        XFCT=YES         XDCT=NO
                  XPSB=YES         XJCT=NO
                                   XPCT=NO
                                   XPPT=NO
                                   XTST=NO
                                   XUSER=NO
                                   XAPPC=NO
```

4. Start the CICS region in which you will be using external security.
5. If you add, change, or delete RACF profiles in the related classes, refresh the in-storage profiles. (For more information, see "Refreshing resource profiles in main storage" on page 29.)

Using IBM-supplied classes with prefixing

To set up external security for transactions, files, and PSBs, using IBM-supplied resource classes with prefixing, take the steps described in this section.

Before you define a profile, you must activate the relevant classes, using the SETROPTS CLASSACT and SETROPTS GENERIC commands, as described in "Brief summary of RACF commands" on page 21.

We recommend that you work in a test region first, to ensure the least interruption to actual business processes.

Note: The following examples assume that the CICS region userid is CICS1.

1. Plan and create RACF profiles in the relevant classes:

```
RDEFINE  TCICSTRN  CICS1.transaction-name  UACC(NONE)  NOTIFY(userid)
RDEFINE  FCICSFCT  CICS1.file-name         UACC(NONE)  NOTIFY(userid)
RDEFINE  PCICSPSB  CICS1.PSB-name          UACC(NONE)  NOTIFY(userid)
```

2. Permit appropriate users or groups (preferably groups) to have access to the profiles:

```
PERMIT  CICS1.transaction-name  CLASS(TCICSTRN)  ACCESS(READ)
        ID(userid or groupid)
PERMIT  CICS1.file-name         CLASS(FCICSFCT)  ACCESS(READ)
        ID(userid or groupid)
PERMIT  CICS1.PSB-name          CLASS(PCICSPSB)  ACCESS(READ)
        ID(userid or groupid)
```

3. Specify the following system initialization parameters:

```
SEC=YES             XTRAN=YES        XCMD=NO
SECPRFX=YES         XFCT=YES         XDCT=NO
                    XPSB=YES         XJCT=NO
                                     XPCT=NO
                                     XPPT=NO
                                     XTST=NO
                                     XUSER=NO
                                     XAPPC=NO
```

4. Start the CICS region in which you will be using external security.
5. If you add, change, or delete RACF profiles in the related classes, refresh the in-storage profiles. (For more information, see “Refreshing resource profiles in main storage” on page 29.)

Using installation-defined classes without prefixing

To set up external security for transactions, files, and PSBs, without prefixing, take the steps described in this section. For an example of installation-defined classes (T$USRTRN and G$USRTRN) for the XTRAN parameter, see the IBM-supplied sample, DFH$RACF, in CICS410.SDFHSAMP.

Before you define a profile, you must activate the relevant classes, using the SETROPTS CLASSACT and SETROPTS GENERIC commands, as described in “Brief summary of RACF commands” on page 21.

We recommend that you work in a test region first, to ensure the least interruption to actual business processes.

1. Set up the following installation-defined classes:

   T$USRTRN like TCICSTRN, and G$USRTRN like GCICSTRN

   F$USRFCT like FCICSFCT, and H$USRFCT like HCICSFCT

   P$USRPSB like PCICSPSB, and Q$USRPSB like QCICSPSB

For specific information on setting up installation-defined classes, see the *System Programming Library: RACF* manual.

1. Plan and create RACF profiles in the relevant classes:

```
RDEFINE  T$USRTRN  transaction-name  UACC(NONE)  NOTIFY(userid)
RDEFINE  F$USRFCT  file-name         UACC(NONE)  NOTIFY(userid)
RDEFINE  P$USRPSB  PSB-name          UACC(NONE)  NOTIFY(userid)
```

2. Permit appropriate users or groups (preferably groups) to have access to the profiles:

```
PERMIT  transaction-name  CLASS(T$USRTRN)  ACCESS(READ)
        ID(userid or groupid)
PERMIT  file-name         CLASS(F$USRFCT)  ACCESS(READ)
        ID(userid or groupid)
PERMIT  PSB-name          CLASS(P$USRPSB)  ACCESS(READ)
        ID(userid or groupid)
```

3. Specify the following system initialization parameters:

```
SEC=YES           XTRAN=$USRTRN         XCMD=NO
SECPRFX=NO        XFCT=$USRFCT          XDCT=NO
                  XPSB=$USRPSB          XJCT=NO
                                        XPCT=NO
                                        XPPT=NO
                                        XTST=NO
                                        XUSER=NO
                                        XAPPC=NO
```

4. Start the CICS region in which you will be using external security.
5. If you add, change, or delete RACF profiles in the related classes, refresh the in-storage profiles. (For more information, see “Refreshing resource profiles in main storage” on page 29.)

Chapter 4. Verifying CICS users

This chapter covers all aspects of CICS signon security, including the use of the RACF CICS segment. It discusses the following:

Identifying CICS terminal users

If you are running CICS with RACF security checking, you control access to CICS resources, by CICS terminal users, through the levels of authorization you define in the appropriate RACF-managed resource profiles. You define these authorizations for specific users by adding individual RACF userids (or RACF group IDs) to the resource access lists; or, for unsigned-on users, by adding the default CICS userid to selected resource access lists.

All CICS terminal-user data must now be defined in the RACF CICS segment. See "Obtaining CICS-related data for a user" on page 72 for more information about CICS terminal-user data, and how CICS obtains it.

Users who do not sign on to CICS are restricted to using only those resources that the CICS default user is authorized to use, or those resources that are not protected (for example, resources with UACC(READ)). The default user should have access only to those resources for which security is not required. Therefore most users on a production CICS region are required to sign on to obtain authorization to access resources.

Signon process

When users logon to CICS through VTAM (or TCAM DCB), but do not sign on, they can use only those transactions that the CICS default user is permitted to use. As these are likely to be strictly limited, users must sign on to obtain authorization to run the transactions that they are permitted to use.

Explicit signon

Users can explicitly sign-on either by using the CICS-supplied transaction, CESN, which can be defined as the "good morning" transaction on the GMTRAN system initialization parameter; or by using an installation-provided signon transaction which uses the EXEC CICS SIGNON command. OIDCARD users can use CESN to signon if the card reader supports the DFHOPID identifier (AID). If it does not, you must use your own installation-provided signon transaction. For information about CESN see the *CICS/ESA CICS-Supplied Transactions* manual. For

programming information about EXEC CICS SIGNON, see the *CICS/ESA*
Application Programming Reference manual.

When a user signs on to CICS, the sign-on process involves the following **phases**:

Scoping
After the signon panel is completed and sent, CICS verifies that the entered userid does not match a userid already signed on within the scope of the SNSCOPE definition for the CICS system.

Identification
CICS calls RACF with the supplied userid to confirm that a profile has been defined for the user.

Verification
CICS passes information to RACF to verify that the user is genuine. For RACF this must be either a password or an OIDCARD or both. If the password entered has expired, CICS prompts the user for a new password. When the new password conforms to the RACF password formatting rules for an installation, the new password and the date-of-change are recorded in the RACF user profile.

Immediately following the request to RACF for userid and password verification, CICS clears the internal password field. This minimizes the possibility of the password being revealed in any dump that is taken for the CICS address space.

You may also voluntarily change your password by entering a new value.

```
                Sign-on for CICS/ESA Release 4.1.0     APPLID CICSA100

. . . . . . This is where the good morning message appears. . . . . .
. . . . . . It can be up to four lines in depth . . . . . .
. . . . . . to contain the maximum message length . . . . . .
. . . . . . of 246 characters . . . . . .



Type your userid and password, then press ENTER:

         Userid . . . . ________    Groupid . . . ________

         Password . . . ________

         Language . . . __

     New Password . . . ________




DFHCE3520 Please type your userid.
F3=Exit
```

Figure 2. The CICS sign-on panel

Authorization
RACF performs checks on the application name and the port of entry to verify that the user is allowed to use the CICS system. In the application name check, RACF determines whether the user is authorized to access the APPLID from the SIT. RACF does this by checking the access list of the CICS

application profile defined in the RACF APPL resource class. (See “Other IBM-supplied RACF resource class names affecting CICS” on page 29 for information about how to define profiles in the APPL resource class.)

With the port of entry check, RACF verifies that the user is authorized to sign-on using that port of entry.

The use of defined terminals can be restricted to certain times of the day, and to certain days of the week. See “Controlling access to CICS from specific ports of entry” on page 67.

These checks restrict the user to signing on only to those CICS regions for which they are authorized, and only from one of the terminals they are authorized to use.

Explicit signon, reached through CESN or EXEC CICS SIGNON, is performed by the user at the port of entry.

Table 8. Explicit and Implicit signons

Phase	Explicit	Implicit
Scoping	Yes	No
Identification	Yes	Yes
Verify	Yes	No (only with ATTACHSEC(IDENTIFY)
Authorize	Yes	Yes

User attributes

CICS obtains CICS user attributes from the CICS and LANGUAGE segment of the RACF database.

Signoff process

When the time-out period expires, CICS schedules the “goodnight transaction” specified in GNTRAN. If the GNTRAN transaction performs a signoff, the action CICS takes depends on the sign-off option specified in terminal’s TYPETERM resource definition. (The default GNTRAN is CESF, the basic CICS signoff without any options).

An exception in the TIMEOUT function is that the goodnight transaction is not used by a surrogate user in a CRTE session. A surrogate who times out is signed off with loss of the security capabilities the terminal previously had, with a message DFHSN1200 in the log to indicate what has happened. For more information about the use of system initialization parameter GNTRAN see “Goodnight transaction” on page 230. The possible signoff options and the associated actions are as follows:

SIGNOFF(YES)
CICS signs off the operator from CICS, but the terminal remains connected.

SIGNOFF(LOGOFF)
CICS signs off the operator from CICS *and* logs off the terminal from VTAM.

In addition, if the terminal is autoinstalled, the delay period specified by the AILDELAY operand in the system initialization parameters commences, and if the delay period expires before the terminal attempts to logon on again, CICS

deletes the terminal entry (TCTTE) from the TCT. For information about CICS autoinstall, see the *CICS/ESA Resource Definition Guide*.

SIGNOFF(NO)
CICS leaves the user signed on and the terminal remains logged on, effectively overriding the time-out period.

Explicit signoff

Explicit signoff removes the user's scoping. The user must be explicitly signed on before signing off with CESF or EXEC CICS SIGNOFF. The user is returned to the default level of security.

Note: CESN will not sign the user off until a valid attempt has been made to use
| the panel, even if the signon attempt subsequently fails. It is not recommended that CESN be used for the "goodnight" transaction.

Implicit signon and Implicit signoff

Implicit signon means that all other userids added to the system by CICS are considered to be implicitly signed on without a password.

APAR PQ08499

MJO 11/11/97

A user is implicitly signed off if the transaction suffers a TERMERR condition while
attempting to send data to its principal facility. However, the user is not subject to
USRDELAY and is signed off immediately. If SNSCOPE is in use, the scope will
be released at the time of sign off. If the transaction handles the ABEND, it
continues running as a non-terminal task with the authority of the starting user.

APAR PN87147

MJO 5/11/96

A user is also implicitly signed off if the terminal at which he is signed on suffers a
TERMERR condition during a transaction. However, he will not be signed off until
the transaction has completed execution. If SNSCOPE is in use, the user will not
be able to sign on at any terminal until the transaction has completed. For this
reason, the transactions should be terminated as quickly as possible after
encountering a TERMERR.

Auditing signon and signoff activity

RACF can log all signon and signoff activity to SMF, including any invalid or unsuccessful signon attempts. You can only properly interpret the logging of unsuccessful signon attempts by also recording successful sign-ons. For example, if a user makes one or two unsuccessful attempts followed immediately by a successful signon, the unsuccessful signons can be interpreted as being caused by keying errors at the terminal. However, several unsuccessful attempts for a variety of userids occurring within a short space of time, and without any subsequent successful signon activity being recorded, may well be cause for a security concern that warrants investigation.

Recording the successful signon and signoff activities also establishes an audit trail of the access to particular systems by the terminal user population. This may also be useful for systems capacity planning, and generally constitutes a very modest portion of the information recorded to SMF.

CICS uses its CSCS transient data destination for security messages. Messages of interest to the security administrator for the CICS region are directed to this destination. In some instances, when security-related messages are directed to terminal users, corresponding messages are written to the CSCS transient data destination. In the case of the DFHCE3544 and DFHCE3545 messages that are sent to terminal users, for example, the corresponding messages DFHSN0118 and DFHSN0119 are sent to CSCS. The DFHSNxxxx messages include reason codes that indicate the precise nature of the “invalid signon attempt.”

XSNON and XSNOFF

XSNON and XSOFF are global user exits that can be used for monitoring signon and signoff activity to CICS. For information about these global user exits, see the *CICS/ESA Customization Guide*.

Controlling access to CICS from specific ports of entry

During signon processing, CICS iissues a request to RACF to verify the user’s password, and to check whether the user is allowed to access that terminal. This check is also performed for the userid specified for preset security terminal definitions. If the terminal is not defined to RACF, RACF responds to CICS according to the system-wide RACF option specified by the SETROPTS command. The options are as follows:

TERMINAL(READ) With this option in force, terminal users can sign on at any terminal covered by a profile to which they have been permitted occurs or any terminal not defined as protected by RACF.

TERMINAL(NONE) With this option in force, terminal users can sign on at only those terminals with specific terminal profiles defined to RACF, and which they are authorized to use.

You can override the system-wide terminal options at the RACF group level by means of the group terminal options, TERMUACC or NOTERMUACC.

See “Universal access authority for undefined terminals” on page 25 for more information about the SETROPTS command for terminals, and about the TERMUACC|NOTERMUACC option on groups.

Preset terminal security

For some selected terminals, and MVS consoles when used as CICS terminals, you should consider using CICS preset terminal security as an alternative to terminal user security. A terminal becomes a preset security terminal when you specify the userid operand on the terminal definition.

Note: If you are using preset security on a console, access is controlled by the CONSOLE general resource class. For more information about preset security on a console, see “Console profiles” on page 25.

CICS preset terminal security allows you to permanently associate a userid with a terminal that is defined to CICS. This means that CICS implicitly “signs on” the terminal when it is being installed, instead of a subsequent signon of that terminal. Typically, you define preset security for devices without keyboards, such as printers, at which users cannot sign on.

You can also use this form of security on ordinary display terminals as an alternative to terminal user security. This permits anyone with physical access to a terminal with preset security to enter the transactions that are authorized for that terminal, without the need to sign on to CICS. The terminal remains signed on as long as it is installed, and no explicit signoff can be performed against it. If the userid associated with a display terminal with preset security has been authorized to use any sensitive transactions, you should ensure that the terminal is in a secure location to which access is restricted. Preset security might be appropriate, for example, for the terminals physically located within a CICS network control center.

You can use preset security to assign a userid with *lower* authority than the default, for terminals in unrestricted areas.

For example, to define a terminal with preset security, use RACF and CICS (CEDA) commands as follows:

```
ADDUSER userid NAME(preset_terminal_user_name) OWNER(owner_userid or group_id)
               DFLTGRP(group_name)
CEDA DEFINE TERMINAL(cics_termid) NETNAME(vtam_termid) USERID(userid)
            TYPETERM(cics_typeterm)
```

For further information on preset security terminals in the transaction routing environment, refer to “Preset security terminals and transaction routing” on page 155 (LU6.2 security) and “Preset security terminals and transaction routing” on page 198 (MRO security).

Controlling the use of preset security

When a preset-security terminal is installed, the specified userid is implicitly signed on at the terminal. Ensure that only a trusted person is allowed to define and install terminals with preset security, because the userid specified on the terminal may have access to CICS resources not available to the installer.

Surrogate user checking ensures that a user is authorized to act for another user. Surrogate user checking can be enforced when a user installs a terminal that is preset for a different userid. Surrogate user checking is specified by the RACF SURROGAT resource class. The CICS *userid.*DFHINSTL resource can be defined in the SURROGAT resource class for authority to install terminals that are preset for that specific userid.

When a terminal is installed with a preset userid, the surrogate user is the userid doing the install operation. See Chapter 7, “Surrogate user security” on page 103 for more information.

The CEDA command checks the authority of the user to install preset terminals. It should therefore be considered whether to restrict the following functions with a view to controlling who can define and install terminals with preset security:

- The CEDA transaction
- The XUSER system initialization parameter
- Batch access to the CSD using the DFHCSDUP utility
- The LOCK command for locking CSD definitions.

Note: When CICS installs a GRPLIST that contains preset terminal definitions no checking is done at initialization time

Although CICS does not check at initialization time you can still ensure that you control who can define and install terminals and sessions with preset security by using the CEDA LOCK command to control the contents of GRPLIST groups.

Restricting use of the CEDA transaction

If the CEDA transaction is enabled on your production CICS regions, its use to authorized users must be restricted. This ensures that control is exercised over who can define resources, such as terminals, to CICS. See Chapter 10, “Security for CICS-supplied transactions” on page 125 for information about protecting CICS-supplied transactions.

You should also ensure that you restrict who can install terminals with preset security, so that even when such terminals are defined in the CSD, only authorized users can install them on CICS. (This authority is additional to the authority needed to run CEDA.) The user must already have authority to run the CEDA transaction.

Using the SURROGAT resource class

To define a surrogate profile and authorize a user to install a terminal definition with preset security, use the following commands:

```
RDEFINE userid1.DFHINSTL SURROGAT UACC(NONE)
PERMIT userid1.DFHINSTL CLASS(SURROGAT) ID(userid2) ACCESS(READ)
```

This permits userid2 to install a terminal preset with userid1.

Defining the XUSER system initialization parameter

To ensure that CICS can perform surrogate user security checks on the use of the CEDA INSTALL command for terminals with preset security, define the XUSER system initialization parameter. See “CICS resource class system initialization parameters” on page 56 for information about defining the XUSER parameter.

Restricting batch access to the CSD

You can also use the CSD batch utility program, DFHCSDUP, to define resources in the CSD. So that only authorized users are allowed to update the production CSDs, you should restrict the access list on the CSD data set profile to the CICS region userids and other authorized users only. The INSTALL command is not available in DFHCSDUP.

Using the LOCK command

CICS also installs resource definitions in the CSD during a cold start, from the list of groups defined on the GRPLIST system initialization parameter. To control the addition of resource groups to the CICS startup group list, you should use the CEDA or DFHCSDUP LOCK command to lock the list. This protects the group list from unauthorized additions. You should also lock all the groups that are specified in this list.

Note: The OPIDENT of the signed-on user is used as the key for the LOCK and UNLOCK commands. For information about LOCK and UNLOCK, see the *CICS/ESA Resource Definition Guide*.

Other preset security considerations

If you intend to use preset security, you need to consider these additional topics:

- Using autoinstall models
- Sessions with preset security
- Terminals defined in the TCT.

Autoinstall models

If you are using autoinstall models with preset security, the same authorization check is made as that for ordinary terminals when the model is installed. No authorization check is made when the autoinstall model is used to perform the automatic installation. If an autoinstall model with preset security becomes invalid, (for example, if the userid is revoked), any attempts to install a terminal with this model fail.

Sessions

A session becomes governed by preset security if you specify the userid operand on the session definition. The same checking is performed if you install preset security sessions.

Terminals defined in the terminal control table

For terminals defined in the terminal control table (TCT) (for example, TCAM DCB terminals), the userid is also defined in the TCT and, when CICS initializes, it signs on these terminals. If the signon fails (for example, the userid is revoked), the terminal is put out of service. If the userid later becomes valid (for example, it is resumed), setting the terminal in service results in a successful signon. No surrogate user check is performed for these terminals.

Using an MVS system console as a CICS terminal

If it is intended to use an MVS system console as a CICS terminal, authorization to use the MVS MODIFY command may be needed. This is done using the OPERCMDS resource class, and is described in “OPERCMDS resource class” on page 33.

We also recommend that preset security is specified on the console’s CICS terminal definition. Otherwise you should explicitly signon to get more authority than the default user. The password will usually be seen on the console and in the system log. However, if CICS has been defined as an MVS subsystem in a JES2 system, you can use the HIDEPASSWORD=YES option of the DFHSSIxx member in SYS1.PARMLIB, which enables CICS to intercept the command and overwrite

the password with asterisks. For details about defining CICS as an MVS subsystem, see the *CICS/ESA Installation Guide*.

The format of the CESN command, when entered from a console, is as follows:

```
MODIFY jobname,CESN [USERID=userid][,PS=password]
       [,NEWPS=newpassword][,GROUPID=groupid]
       [,LANGUAGE=language-code]
```

If any of the data entered on the CESN command is invalid, or if the password is missing or expired, CICS prompts the user to enter the missing or invalid data by issuing a system message that requires a response (a WTOR message). The response must be provided by issuing the REPLY command. When CICS prompts for a password, it uses a security routing code to ensure that the response is not recorded on the console or in the system hardcopy log. To terminate the signon process, a REPLY command with a null operand should be entered. That is, enter

```
REPLY nn,
```

with nothing after the comma, where `nn` is the number of the message that the user is replying to. If MVS/ESA SP 4.1 is being used, TSO users can be authorized to use the TSO CONSOLE command. (For information on this command, see the *TSO/E System Programming Command Reference* manual, SC28-1878.) These users must be defined to CICS as consoles, using the CONSNAME option of the DEFINE TERMINAL command, as described in the *CICS/ESA Resource Definition Guide*.

APAR PQ10606

MJO 3/2/98

When the password parameter is omitted from the CESN command, RACF can produce a security violation message ICH408I. CESN cannot distinguish a user defined with OIDCARD, NOPASSWORD from a user defined with a PASSWORD who intentionally omits the password. To establish whether to prompt for a PASSWORD or to reject the signon (a user defined with OIDCARD cannot sign on at a console), the signon must be attempted. If the signon fails, message ICH408I is produced, and CICS interprets the return code from RACF to determine whether the PASSWORD or OIDCARD authenticator is required.

These users can sign on using CESN, or you may prefer to use preset security. When the TSO user uses the CONSOLE command, that user's userid becomes a console name. This console name can then be used as a CICS terminal if there is a corresponding TERMINAL definition with the CONSNAME option in CICS.

Furthermore, if the CONSOLE command is used to allow TSO operators to sign on to CICS with the CESN transaction, their passwords may be exposed on the TSO screen and in the MVS system log. These potential exposures can be removed by defining the terminal as having preset security.

Obtaining CICS-related data for a user

CICS obtains CICS-related data from one of the following sources:

- The CICS and LANGUAGE segments of the RACF profile
- Built-in CICS system default values.

How the data is obtained, for the default user and terminal users signing on, is explained in the following sections.

Obtaining CICS-related data for the default user

When implicitly signing on the CICS default user during initialization, CICS obtains the attributes for the default user in the following way:

1. CICS calls RACF to request user data for the CICS default user from the CICS segment and the LANGUAGE segment. If the CICS segment *or* the LANGUAGE segment data are present for the default userid, RACF returns this data to CICS. See "CICS segment" on page 14 for details of the information that you can define in the CICS segment. See "LANGUAGE segment" on page 17 for details of the LANGUAGE segment.
2. If RACF does not return the CICS segment or LANGUAGE segment data for the default userid, CICS assigns the following built-in system default values:

   National language Obtained from the first operand on the NATLANG system initialization parameter. This defaults to US English if not specified.

   Operator class One (OPCLASS=1)

   Operator identification Blank (OPIDENT=' ')

   Operator priority Zero (OPPRTY=0)

   Timeout Zero (TIMEOUT=0)

   XRF signoff Signoff not forced (XRFSOFF=NOFORCE)

Obtaining CICS-related data at signon

When handling an explicit signon a CICS terminal user, CICS obtains the CICS segment terminal user attributes in the following way:

1. CICS calls RACF to request data about the CICS terminal user from the CICS segment and the LANGUAGE segment. If the CICS segment *or* the LANGUAGE segment data are present for the terminal user, RACF returns this data to CICS. See "CICS segment" on page 14 for details of the information that you can define in the CICS segment. See "LANGUAGE segment" on page 17 for details of the LANGUAGE segment.
2. If RACF does not return the CICS segment or LANGUAGE segment data for the user, CICS uses the user attributes of the CICS default user, defined during system initialization. (See "Obtaining CICS-related data for the default user.")

CICS obtains the national language attribute in the following order:

1. The LANGUAGE option on the CICS-supplied CESN transaction, or the LANGUAGECODE or NATLANG option of the EXEC CICS SIGNON command, if supported by CICS. A *supported* national language is a *valid* national language which has been specified in the NATLANG system initialization

parameter and has the corresponding message definitions. See the *CICS/ESA System Definition Guide* for more information about defining this parameter.

2. The PRIMARY(primary-language) parameter in the LANGUAGE segment of the user's RACF profile, if supported by CICS.
3. The SECONDARY(secondary-language) parameter in the LANGUAGE segment of the user's RACF profile, if supported by CICS.
4. The NATLANG parameter in the CSD definition of the user's terminal.
5. The language established for the default user as described on page 72.

See Appendix B, "National Language" on page 287 for a list of valid national languages.

Note: CICS ignores the RACF default national language defined by the command:

```
SETROPTS LANGUAGE(PRIMARY(...) SECONDARY(...))
```

Defining terminal users and user groups to RACF

You should plan to define your CICS terminal users in groups. For this purpose, try to place the users of CICS systems in groups for ease of administration. For example, you might consider that all users who have the same manager, or all users within an order entry function, are administrative units. You can define such users to RACF as *groups* of individual users who have similar access requirements to CICS system resources. See the *RACF Security Administrator's Guide* for more information about:

- Access control and flexibility of operation for the system administrator
- Use of the group-SPECIAL attribute and its scope of control
- Reducing the need to refresh in-storage profiles.

When you define a group, and then define users as members of that group, all the users in the group can access the resources to which the group has been given access.

The group structure selected depends on your own installation's requirements. Use the RACF command ADDGROUP to create a new group:

```
ADDGROUP groupname OWNER(userid)
```

Use the ADDUSER command to add new users to the group, defining the group name as the user's default group:

```
ADDUSER userid NAME(username) DFLTGRP(group_id)
        CICS(OPCLASS(1,2,..,n) OPIDENT(abc) OPPRTY(255) TIMEOUT(minutes)
             XRFSOFF(NOFORCE) LANGUAGE(PRIMARY(language))
```

You can make a terminal user a member of more than one group by using the CONNECT command to add the user to a group other than that user's default group:

```
CONNECT userid GROUP(groupname)
```

Use the ALTUSER command to change a user's default, as follows:

```
ALTUSER userid DFLTGRP(groupname)
```

See the *RACF Command Language Reference* manual for the full syntax of these commands.

Use the ALTUSER command to add CICS data for an existing userid. See "CICS segment" on page 14 for details of the CICS optional data.

Example of defining terminal users and user groups to RACF

Assume there is a customer service department that:

- Takes orders
- Answers enquiries about those orders
- Establishes new customers.

Consider creating the following customer service group:

```
ADDGROUP custserv OWNER(grpmangr)
```

In this example, *grpmangr* is the RACF userid of the person in charge of the customer service department system.

The person represented by *grpmangr*, or the RACF security administrator, can then create additional groups within the group CUSTSERV, as follows:

```
ADDGROUP ORDERS OWNER(SUP1) SUPGROUP(CUSTSERV)
ADDGROUP ORDINQ OWNER(SUP2) SUPGROUP(CUSTSERV)
ADDGROUP NEWCUST OWNER(SUP3) SUPGROUP(CUSTSERV)
```

The group owners, the person represented by grpmangr or the RACF security administrator can then define users within the groups. For example, the person represented by SUP1 could define users of the group ORDERS, as follows:

```
ADDUSER AARCHER NAME('ANNE ARCHER') DFLTGRP(ORDERS)
ADDUSER JBRACER NAME('JOHN BRACER') DFLTGRP(ORDERS) PASSWORD(XPRDTD)
        CICS(OPCLASS(1) OPIDENT(JBR) OPPRTY(0) TIMEOUT(15) XRFSOFF(FORCE))
        LANGUAGE(PRIMARY(ENU))
```

Notes:

1. The password of the user Anne Archer defaults to ORDERS, but the password of the user John Bracer is initially set as XPRDTD.
2. The user John Bracer is defined with a CICS segment and with a LANGUAGE segment.

National language and non-terminal transactions

When a user specifies a national language during signon, the signon option overrides the language specified in the user's RACF CICS segment. The language thus specified is set for the duration that the user is signed on at the terminal. Any transaction invoked by the signed-on user runs with the national language specified on the signon.

However, if transactions use the EXEC CICS START command to start other transactions, the national language attribute for the started transactions is derived as follows:

1. If the USERID parameter is specified on the START command, the national language is taken from the RACF CICS segment of the specified userid.

2. If the user is signed on at a terminal with a preset national language specified on the terminal definition, this preset national language is assigned to the started transaction.
3. If there is no userid on the START command, and no preset national language on the terminal, the started transaction inherits the national language specified in the RACF CICS segment of the signed-on user (not the national language used in the signon).

If the national language of the original terminal is required, the terminal's national language can be inquired before the EXEC CICS START command is issued. The information can then be passed as data in the START command for the STARTed transaction to use.

Chapter 5. Transaction security

CICS can apply two levels of security to transactions. The first is security checking on the transaction itself, sometimes referred to as **attach-time**, or **transaction-attach security**. This chapter discusses transaction-attach security—the security checks that CICS performs to verify that a terminal user is authorized for the transaction to be run at the user's terminal.

Transaction-attach security applies to transactions that a user enters directly at a terminal, and also to transactions started from another CICS transaction. The other level of security you can use for CICS transactions applies to the resources used by the transactions: files, databases, PSBs, and CICS commands.

This chapter discusses transaction-attach security under the following main headings:

The % and & characters should be avoided when you define a CICS transaction
name because some characters have a special meaning to RACF. They can be
used, however, in RACF profiles that are used to protect CICS transactions.

CICS parameters controlling transaction-attach security

You control CICS transaction-attach security checking through CICS system initialization parameters. These are:

SEC Specify SEC=YES if you want to use RACF services to control access to any CICS resources — in particular, CICS transactions. (For more information, see "SEC" on page 54.)

SECPRFX Specify SECPRFX=YES if your transaction profiles are defined to RACF with a prefix that corresponds to the userid of the CICS region. (For more information, see "SECPRFX" on page 54.)

XTRAN Specify XTRAN=YES or XTRAN=resource_class_name if you want CICS to control who can initiate transactions. If you specify YES, CICS uses profiles defined in the RACF default resource classes TCICSTRN and GCICSTRN. (See "IBM-supplied resource class names for CICS" on page 28 for details of these resource classes.)

If you specify XTRAN=NO, CICS does not perform any authorization check on users initiating transactions.

Note that the default is YES. Therefore if you specify SEC=YES and omit the XTRAN parameter, transaction-attach security is in effect, using the default resource class names.

There are no CICS parameters that allow you to control transaction-attach security at the individual transaction level. When you specify SEC=YES and XTRAN=YES (or XTRAN=resource_class_name), CICS issues an authorization request for every
| transaction. It does this whether the transaction is started from a terminal, by using

an EXEC CICS START command, or triggered from the transient data queue, either with or without the termid operand. CICS performs this security check even if no user has signed on. Users who do not sign on can use only those transactions that are authorized to the default user.

Figure 3 illustrates the main elements of CICS transaction security.

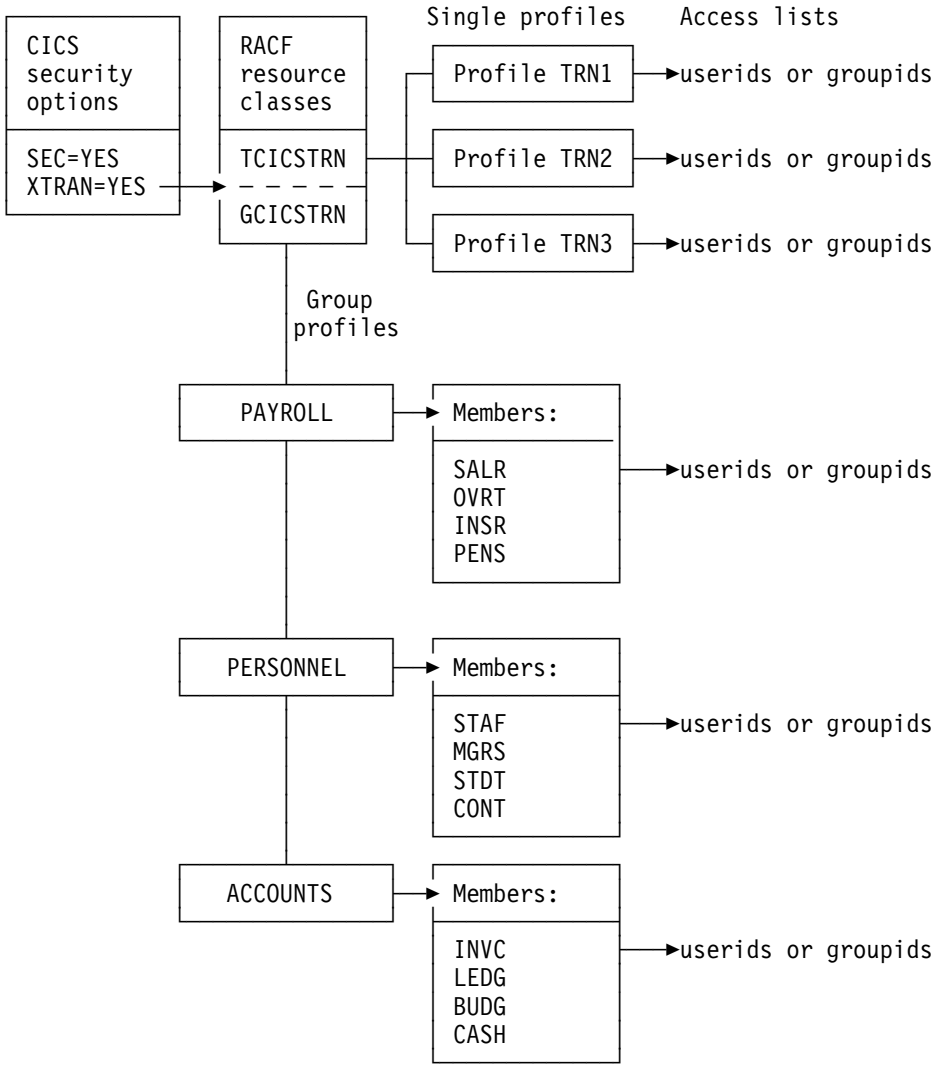

Figure 3. Illustration of the main elements of CICS transaction security

Transaction-attach processing when SEC=YES and XTRAN=YES

Every time a transaction is initiated at a CICS terminal, CICS issues an authorization request to determine whether the user associated with the terminal is authorized for that transaction. CICS and RACF process the authorization request using the currently active transaction profiles in the RACF class identified by the XTRAN SIT parameter. (For more information, see “Refreshing resource profiles in main storage” on page 29.)

Defining transaction profiles to RACF

For those CICS regions running with transaction security checking, you must define transaction profiles for all transactions that need to be protected from unauthorized access. You can either define these profiles in the default transaction resource classes, or in installation-defined classes that you have added to the RACF class descriptor table. (See "IBM-supplied resource class names for CICS" on page 28 for information about the transaction resource classes.)

Some recommendations

The following recommendations are intended to reduce the amount of work involved:

- Define transactions in the resource group class, GCICSTRN. This minimizes the amount of effort needed to define and maintain transaction profiles and their associated access lists, and also keeps down the size of in-storage profiles. However, note that using resource groups only reduces the amount of storage required if you avoid defining duplicate member names.
- Add users to the access list in groups rather than as individual users, and define access as READ.
- Use generic profiles or member names wherever possible.

For example, the following RDEFINE and PERMIT commands illustrate some example payroll transactions, with access given to members of the payroll department:

```
RDEFINE  GCICSTRN  salarytrans
         NOTIFY(pay_manager)
         UACC(NONE)  ADDMEM(Pay1, Pay2, Pay3,.., Payn)
PERMIT  salarytrans CLASS(GCICSTRN)
        ID(paydept_group_userid) ACCESS(READ)
```

In this example, you could alternatively define the members generically, such as `P*` or `Pay*`.

| However, before you define a generic profile you must issue the command:

| `SETROPTS GENERIC(TCICSTRN)`

| You cannot specify the GCICSTRN class, because grouping classes cannot be
| used with the SETROPTS GENERIC command.

If you have transactions that anyone can use, you can avoid maintaining access lists for them by defining RACF transaction profiles for them with UACC(READ). For example:

```
RDEFINE TCICSTRN tranid UACC(READ)
```

| If you want to avoid defining all your transactions to RACF, you can specify
| universal access as follows:

| `RDEFINE TCICSTRN ** UACC(READ)`

| Then you need to define to RACF only those transactions that require more
| restrictive security.

Note: If you use a profile like that described above, you must define new profiles to RACF before installing the new CICS resources.

Using conditional access lists for transaction profiles

You can add another element of security by making the access list conditional upon the user being signed on at a particular terminal or console.

For example, if the above payroll examples are defined as generic transactions in the TCICSTRN class, you could define conditional access as follows:

```
RDEFINE  TCICSTRN  PAY*
         NOTIFY(pay_manager)  UACC(NONE)
PERMIT  pay*  CLASS(TCICSTRN)  ID(userid) ACCESS(READ)
              WHEN(TERMINAL(termid))
              WHEN(CONSOLE(*))
```

Notes:

1. The TERMINAL or CONSOLE class must be active for this support to take effect.
2. WHEN(TERMINAL(termid)) applies only to explicitly signed-on users, and only in the region where the user is explicitly signed on, and in regions connected to it by MRO links only.
3. CICS only uses the console and terminal ports of entry.

CEBT transaction

The CEBT transaction (the master terminal transaction used to control the alternate CICS system in an XRF environment) is not subject to transaction security checking. This means that any user is authorized to use CEBT. CEBT can only be issued from the operating system console, using the MODIFY command. You can use the OPERCMDS resource class to control who is allowed to use the MODIFY command. (For more information, see “OPERCMDS resource class” on page 33.)

Authorization failures and error messages

If terminal users try to initiate transactions that they are not authorized to use, CICS issues a security violation message (DFHAC2033) to the terminal. CICS then sends a corresponding message (DFHAC2003) to the CSMT transient data destination, and a DFHXS1111 message to CSCS. RACF issues an ICH408I message to the CICS region’s job log and to the security console (the console defined for routing code 9 messages). For a description of the ICH408I message, see the *RACF Messages and Codes* manual.

For more information on resolving authorization problems, see Chapter 20, “Problem determination in a CICS-RACF security environment” on page 249.

If auditing (such as that requested by the AUDIT operand) is requested for this access, RACF writes an SMF type 80 log record. Your RACF auditor can use the RACF report writer to generate reports based on these records. For more information, see the *RACF Auditor’s Guide*.

Transactions not associated with a terminal

For all resource security checking, CICS needs a userid in order to check the user's authority to access the resource. CICS can protect resources against unauthorized use if those resources are used in transactions that are not associated with a terminal. In addition to transactions started by an EXEC CICS START command without a terminal identifier specified, there are two other types:

- Transactions started without a terminal when the trigger level is reached for an intrapartition transient data queue
- Programs executed from the second phase of the program list table (PLT) during CICS startup.

Triggered transactions

The DFHDCT macro and the ATIUSERID option of the EXEC CICS SET command handle security for non-terminal transactions started by a transient data trigger level. The user issuing the SET command must have surrogate authority for the userid specified on the ATIUSERID option. The user to be associated with the triggered transaction is specified on the USERID operand on DFHDCT TYPE=INITIAL or TYPE=INTRA macros.

PLT programs

During CICS startup, a surrogate user security check is done for the region userid. See "Defining user profiles for CICS region userids" on page 43. This check determines whether the CICS job is authorized to be the surrogate of the userids specified on the ATIUSERID option and the PLTPIUSR parameter. The PLTPIUSR and PLTPISEC system initialization parameters specify security options for PLT programs that are run from the third stage of CICS startup (which is the second phase of the PLTP initialization.)

PLT programs that are run during shutdown are run under the authority of the userid for the transaction that requests the shutdown. The values of the RESSEC and CMDSEC options for that transaction are also applied to the PLT programs. If RESSEC(YES) and CMDSEC(YES) are specified on the definition of the transaction issuing the EXEC CICS PERFORM SHUTDOWN command, security checking is done at the first stage of shutdown.

The PLTPIUSR system initialization parameter specifies which userid is authorized to attach the PLT programs (and will be propagated to any transactions STARTed from PLTPI processing). By specifying the PLTPISEC parameter you can use the additional options RESSEC and CMDSEC once the transactions are attached.

Chapter 6. Resource security

This chapter describes:

- The facilities provided by CICS and RACF for controlling access to resources protected by RACF general resource security classes.

Chapter 5, “Transaction security” on page 77 described how to control access to CICS transactions, using CICS transaction-attach security. This chapter describes how you can implement a further level of security, by controlling access to the resources used by the CICS transactions. The implication of this is that although a user may be authorized to invoke a particular CICS transaction, the user may not be authorized to access files, PSBs, or other general resources used within the transaction. Unlike transaction-attach security, which you cannot “switch off” for individual transactions, you can control resource security checking at the individual transaction level.

Resources defined to CICS to support application programming languages are also subject to security checking if resource or command security checking is specified. For example, if a PL/I program abends, it may attempt to write diagnostic information to the CPLI transient data queue. If resource checking is active, and the user is not authorized to write to the CPLI transient data queue, the program will terminate with an APLI abend.

You control who can access the general resources used by CICS transactions by:

- Specifying SEC=YES as a system initialization parameter
- Specifying RESSEC=ALWAYS as a system initialization parameter
- Specifying RESSEC(YES) in the transaction resource definition
- Specifying the types of resource you want to protect by defining CICS system initialization parameters for the RACF general resource classes
- Defining the CICS resources to RACF in resource class profiles, with appropriate access lists.

General resource security checking by CICS and RACF

CICS uses RACF to protect the general resources that you can access through a CICS application program. These resources are described briefly in Table 9, with the associated CICS system initialization parameter that you use to specify the RACF class names.

Table 9 (Page 1 of 2). General resource checking by CICS

CICS parameter	General resource protected	Further information
XAPPC	Partner logical units (LU6.2). This resource is included in this list for completeness, but is not discussed in this chapter.	Chapter 12, "Implementing LU6.2 security" on page 141.
XCMD	The subset of CICS application programming commands that are subject to command security checking. This resource is included in this list for completeness, but is not discussed in this chapter. EXEC CICS FEPI system commands are also controlled by this parameter.	Chapter 8, "CICS command security" on page 109.
XDCT	CICS extrapartition and intrapartition transient data destinations, also known as queues. Define profiles in the destination class to control who is allowed to access CICS transient data queues.	"Transient data" on page 88.
XFCT	CICS file-control-managed VSAM and BDAM files. Define profiles in the file class to control who is allowed to access CICS VSAM and BDAM files.	"Files" on page 90.
XJCT	CICS system log and journals. Define profiles in the journal class to control who is allowed to access CICS journals.	"Journals" on page 91.
XPCT	CICS started transactions and EXEC CICS commands: COLLECT STATISTICS TRANSACTION, DISCARD TRANSACTION, INQUIRE TRANSACTION, INQUIRE REQID, SET TRANSACTION, and CANCEL. Define profiles in the started-transactions class to control who is allowed access to started CICS transactions.	"Started and XPCT-checked transactions" on page 92.

Table 9 (Page 2 of 2). General resource checking by CICS		
CICS parameter	**General resource protected**	**Further information**
XPPT	CICS application programs. Define profiles in the program class to control who is allowed to access CICS application programs that a CICS application invokes by means of a LINK, XCTL, or LOAD command.	“Application program security” on page 95.
XPSB	DL/I program specification blocks (PSBs). Define profiles in the program specification block class to control who is allowed to access the DL/I PSBs used in CICS application programs.	“Program specification blocks” on page 98.
XTRAN	CICS transactions. This resource is included in this list for completeness, but is not discussed in this chapter.	Chapter 5, “Transaction security” on page 77.
XTST	CICS temporary storage destinations. Define profiles in the temporary storage class to control who is allowed to access CICS temporary storage queues.	“Temporary storage” on page 96.
XUSER	Surrogate user security. This resource is included in this list for completeness, but is not discussed in this chapter	Chapter 7, “Surrogate user security” on page 103.

Note that no authorization processing is done for BMS commands.

RESSEC transaction resource security parameter

Specifying RESSEC(YES) in the definition of a transaction, together with the appropriate resource classes defined in the system initialization parameters, introduces another layer of security checking in addition to the transaction-attach security described in “Transaction-attach processing when SEC=YES and XTRAN=YES” on page 78.

For most simple (or single-function) transactions, this extra layer of security should not be necessary. For example, if the transaction is designed to enable the terminal user to update the personnel file and nothing else, it should be sufficient to authorize access to the transaction without controlling access to the file also. However, if you have complex or multiple-function transactions that offer users a choice of functions, or you are unsure about all the options available within a transaction, you may need to add the extra layer of security to restrict access to the data as well as to the transaction. Before implementing resource security checking, you should take into account the extra overhead that resource security checking involves, and only implement it if you believe the extra cost is worthwhile.

If you specify RESSEC(YES) on a transaction definition, CICS calls RACF for every CICS command that applies to a resource for which you have requested security,

using an X*name* resource class parameter. This is shown in Figure 4, in which the execution of transaction TRN1 results in seven RACF calls.

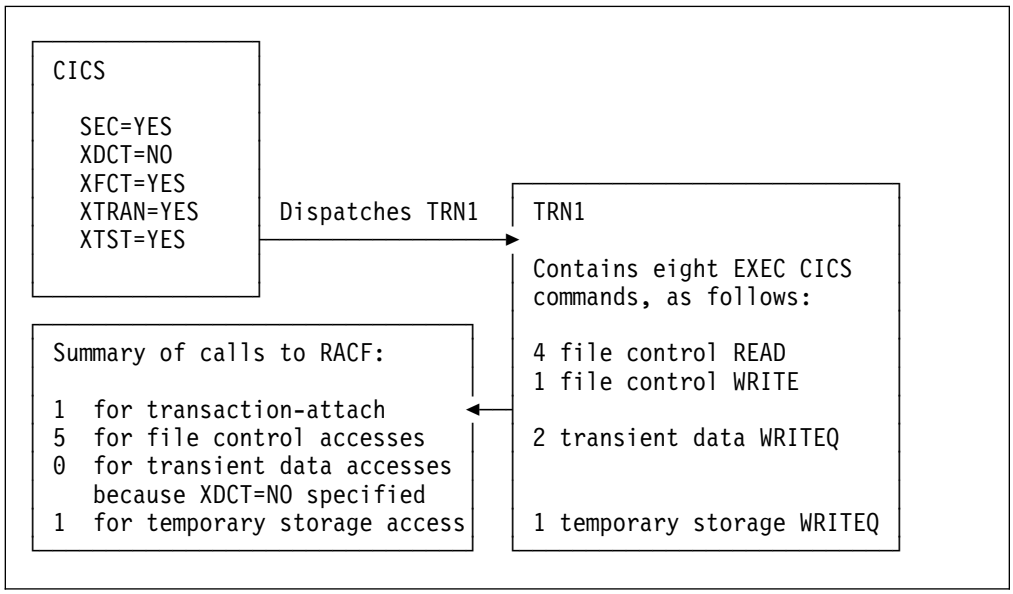

Figure 4. Multiple calls to RACF with resource security checking

The RESSEC system initialization parameter

You can force the effect of RESSEC=YES for all CICS transactions by specifying the RESSEC=ALWAYS system initialization parameter. In general, this is not recommended for the following reasons:

- For most simple transactions, just controlling access to the transaction is enough to control everything that the transaction can do.
- Invoking a resource check for every CICS resource consumes extra overhead that reduces the performance of all your transactions.
- Some CICS-supplied transactions may access resources of which you are unaware. It is your responsibility to ensure that users of these transactions are given enough authority to allow the transactions to continue to work.

Authorization failures

If a terminal user is not authorized to access the resource specified on a CICS command, CICS returns the NOTAUTH condition to the application program. CICS indicates this authorization failure by setting the EIBRESP field of the EXEC interface block (DFHEIBLK) to a value of 70 (and X'46' in byte 0 of the EIBRCODE field). Your CICS applications should be designed to handle security violations by passing control to an appropriate routine. They can do this in either of the following ways:

- Test the EIBRESP condition by adding the RESP option to each command that may receive a NOTAUTH condition. For example (in COBOL):

```
EXEC CICS FILE('FILEA')
   INTO(REC) RIDFLD(KEY)
        RESP(COMMAND-RESPONSE)
END-EXEC.


EVALUATE COMMAND-RESPONSE
   WHEN DFHRESP(NORMAL)
        CONTINUE
   WHEN DFHRESP(NOTAUTH)
    PERFORM SECURITY-ERROR
END-EVALUATE.
```

- Code an EXEC CICS HANDLE CONDITION NOTAUTH(label) command, where “label” is the name of the security violation routine.

If an application does not cater for security violations, CICS abends the transaction with an AEY7 abend code.

Logging RACF audit messages to SMF

With the exception of certain security commands (see Chapter 9, “Security checking using the QUERY SECURITY command” on page 117), CICS issues security authorization requests with the logging option. This means that RACF writes SMF type 80 log records to SMF. Which events are logged depends on the auditing in effect. For example, events requested by the AUDIT or GLOBALAUDIT operand in the resource profile, or by the SETROPTS AUDIT or SETROPTS LOGOPTIONS command, can be logged.

In addition to the SMF TYPE 80 log record, RACF issues an ICH408I message to consoles designated to receive messages for route code 9.

For more information on auditing, including how to use the RACF report writer to review SMF type 80 log records, see the *RACF Auditor’s Guide*.

Use of the WARNING option

The RACF WARNING option, if used on RACF profiles, is honored by CICS. The WARNING option allows users access to resources that otherwise would be denied. RACF logs to SMF occurrences of access that would have failed had WARNING not been in effect.

The selective use of WARNING can be particularly useful during the initial implementation of resource security for an application, as a means of checking for errors or omissions in the RACF security definitions. When WARNING results in an SMF type 80 record being recorded, you should verify whether the user should be added to the access list for the resource, and modify the RACF profiles accordingly. You should strictly limit the time during which resources are accessed with the warning option in force, and keep logging to a minimum during the warning period.

Note: For immediate notification of access authorization failures, specify the NOTIFY option.

Security for general resource types

Transient data

To implement security for transient data destinations (queues), do the following:

1. Specify RESSEC(YES) in the CSD resource definition of the appropriate transactions.
2. Define profiles to RACF in the DCICSDCT or ECICSDCT resource classes (or their equivalent if you have user-defined resource class names), with access lists as appropriate. Transient data queue names are a maximum of 4 characters in length, such as CSMT, CPLI, L86O, L86P, and so on.

   For example, use the following commands to define queues in the DCICSDCT class, and to authorize users to both read and write to these queues:

   ```
   RDEFINE  DCICSDCT  (qid1, qid2, ..., qidn) UACC(NONE)
                      NOTIFY(sys_admin_userid)
   PERMIT qid1 CLASS(DCICSDCT) ID(group1, group2) ACCESS(UPDATE)
   PERMIT qid2 CLASS(DCICSDCT) ID(group1, group2) ACCESS(UPDATE)
   ```

   To define transient data queues as members of a profile in the CICS transient data resource group class, with an appropriate access list, use the following commands:

   ```
   RDEFINE  ECICSDCT  (queue_groupname) UACC(NONE)
                      ADDMEM(qida, qidb, ..., qidz) NOTIFY(sys_admin_userid)
   PERMIT queue_groupname CLASS(ECICSDCT) ID(group_userid) ACCESS(UPDATE)
   ```

3. Specify SEC=YES as a CICS system initialization parameter (and SECPRFX=YES if you define profiles with the CICS region userid as a prefix).
4. Specify XDCT=YES for the default resource class names of DCICSDCT and ECICSDCT (or XDCT=class_name for user-defined resource class names).

Defining profiles for transient data queues

When you are defining profile names to RACF to control access to transient data queues, you should define profiles only for queues that are defined to CICS as follows:

TYPE=INTRA For an intrapartition transient data queue held on the CICS intrapartition (VSAM) data set, DFHINTRA. When DESTFAC=FILE, it is possible to specify a USERID. See "Considerations for triggered transactions" on page 89 for more information about intrapartition TD queues in this category, and "Transient data trigger-level transactions" on page 105 for more information about the USERID specification.

TYPE=EXTRA For an extrapartition transient data queue on a sequential data set.

TYPE=REMOTE For a transient data queue on another CICS region.

You define an indirect queue, however, with a destination control table entry as TYPE=INDIRECT, and CICS directs this to another destination, which can be extrapartition, intrapartition, or remote. The redirection can even be to another indirect destination. See the *CICS/ESA Resource Definition Guide* for more information about how to define CICS transient data queues in a destination control table (DCT).

If you are running CICS with security checking for transient data queues, CICS issues a call to RACF for each command that specifies a queue name. However, the resource name that CICS passes to RACF is the queue name of the final destination, which is not necessarily the name of the queue specified on the command.

For example, if an EXEC CICS command specifies queue QID2, which is defined as indirect to QID1, CICS calls RACF for an authorization check on QID1, not QID2. This is illustrated as follows:

```
DCT entries:  QID1  TYPE=EXTRA,
                    DESTID=QID1,
                    DSCNAME=CICSMSGS            (Final destination)
              QID2  TYPE=INDIRECT,
                    DESTID=QID2,
                    INDDEST=QID1                (Indirect to QID1)
CICS transaction:   EXEC CICS WRITEQ TD
                         QUEUE(QID2)
                         FROM(data_area)
                         LENGTH(length)
CICS calls RACF:    Does the terminal user of the CICS transaction
                    have UPDATE authorization for QID1?
```

Access authorization levels

You can read an item from a transient data queue only once, because whenever you read from a transient data queue, CICS deletes the entry (by performing a "destructive read"). Therefore, if you specify security with SEC=YES as a system initialization parameter, CICS requires a minimum authorization level of UPDATE for all TD commands (DELETEQ, WRITEQ, and READQ).

CICS-required destination control table entries

CICS itself uses a number of queues. These queues are defined in the sample destination control table, DFHDCT2$, which can be found in CICS410.SDFHSAMP. If you want to protect access to these from user application programs, define them to RACF with UACC(NONE) and without an access list. In the sample table, most of the queue names are indirect, pointing to the final destinations: CPLI, CSSL, or CCSO. Therefore, if you use the definitions as supplied, you need define only the queue names CPLI, CSSL, and CCSO to RACF, as follows:

```
RDEFINE  ECICSDCT  CICSQUEUES UACC(NONE)
                   ADDMEM(CPLI, CSSL, CCSO)
                   NOTIFY(sys_admin_userid)
```

Considerations for triggered transactions

For intrapartition TD queues with a trigger level greater than zero, the userid associated with the triggered transaction is derived from the following sources:

- The USERID parameter specified on the TYPE=INITIAL or TYPE=INTRA macro (for queues that specify DESTFAC=FILE).
- The userid associated with the terminal (for queues that specify DESTFAC=TERMINAL). This can be the CICS default userid if there is no user signed on at the terminal.
- The link userid on the connection definition (for queues that specify DESTFAC=SYSTEM).

Files

CICS application programs process *files*, which, to CICS, are logical views of physical VSAM or BDAM data sets. You identify a file to CICS by an 8-character *file name*, and you can define many files to CICS that refer to the same physical data set, which is separately identified by a 44-character data set name (DSNAME). For example, you can define file resource definitions called FILEA, FILEB, and FILEC, all of which refer to one physical VSAM data set, but with each file definition specifying different attributes.

CICS transactions access the data in physical data sets using the CICS file control name. Therefore, you control access to CICS-managed files by defining profiles in the RACF general resource classes for CICS files, not in the RACF data set class. You define the profiles using the CICS 8-character file name to identify the resource. (RACF data set authorization based on the 44-character data set name is used only during OPEN processing, to determine whether the CICS region userid is authorized to access the data set for which the OPEN has been requested. This does not depend on the userid running the transaction that caused the OPEN to be performed.)

To implement security for files managed by CICS file control, you must do the following:

1. Specify RESSEC(YES) in the CSD resource definition of the transactions that access the files.
2. Define profiles to RACF in the FCICSFCT or HCICSFCT resource classes (or their equivalent if you have user-defined resource class names), using the CICS file names to identify the profiles. For example, use the following commands to define files in the FCICSFCT class, and authorize users to read or write to the files:

   ```
   RDEFINE  FCICSFCT  (file1, file2, .., filen) UACC(NONE)
                     NOTIFY(sys_admin_userid)
   PERMIT file1 CLASS(FCICSFCT) ID(group1, group2) ACCESS(UPDATE)
   PERMIT file2 CLASS(FCICSFCT) ID(group1, group2) ACCESS(READ)
   ```

   To define files as members of a profile in the CICS file resource group class, with an appropriate access list, use the following commands:

   ```
   RDEFINE  HCICSFCT  (file_groupname) UACC(NONE)
                      ADDMEM(filea, fileb, .., filez) NOTIFY(sys_admin_userid)
   PERMIT file_groupname CLASS(HCICSFCT) ID(group_userid) ACCESS(UPDATE)
   ```

3. Specify SEC=YES as a CICS system initialization parameter (and SECPRFX=YES if you define profiles with the CICS region userid as a prefix).
4. Specify XFCT=YES for the default resource class names of FCICSFCT and HCICSFCT (or XFCT=class_name for user-defined resource class names).

Note that RDO transactions do not use file commands to access the CSD, and are not, therefore, subject to these mechanisms. For information on how you can protect RDO transactions, see “Controlling the use of preset security” on page 68.

Access authorization levels

If you specify security with SEC=YES as a system initialization parameter, CICS requires a level of authorization appropriate to the file access intended: a minimum of READ for read intent, and a minimum of UPDATE for update or delete intent.

Journals

You can define to your CICS regions up to 99 journals, using journal identifiers as part of the DD names in the range 01 through 99 (DFHJ01x – DFHJ99x, where 'x' is a suffix letter A or B).

In addition to the automatic journaling that CICS performs for user transactions (depending on the options in the file resource definitions), user applications can also write user journal records using the EXEC CICS WRITE JOURNALNUM command. CICS calls RACF to perform a security check only for attempts to access a journal by a CICS API command, and not for the journaling it performs in response to journaling options in the file resource definition.

CICS uses **journal identifier 01** for its system log, and you should not permit user transactions to write to this. You are recommended to restrict user journaling activity to the user journals, 02–99. The CICS API does not provide a READ command for reading journals from a CICS transaction. For this reason, with proper exercise of control over the installation of applications on your CICS systems, you might consider it unnecessary to add RACF protection for journals that cannot be read from within CICS.

If you decide to implement security for CICS journals, you must do the following:

1. Specify RESSEC(YES) in the CSD resource definition of the transactions that write to journals.
2. Define profiles to RACF in the JCICSJCT or KCICSJCT resource classes (or their equivalent if you have user-defined resource class names) using the CICS journal name (without the suffix letter) to identify the profiles. (The RACF resource name associated with the journal nn is DFHJnn.)

   For example, use the following command to define the system log in the JCICSJCT class, without any access list:

   ```
   RDEFINE  JCICSJCT  DFHJ01  UACC(NONE) NOTIFY(sys_admin_userid)
   ```

   To define journals as members of a profile in the journal resource group class, with an appropriate access list, use the following commands:

   ```
   RDEFINE  KCICSJCT  userjnls UACC(NONE)
                      ADDMEM(DFHJ02, DFHJ03, ..., DFHJnn)
                      NOTIFY(sys_admin_userid)
   PERMIT   userjnls  CLASS(KCICSJCT) ID(group_userid) ACCESS(UPDATE)
   ```
3. Specify SEC=YES as a CICS system initialization parameter (and SECPRFX=YES if you define profiles with the CICS region userid as a prefix).
4. Specify XJCT=YES for the default resource class names of JCICSJCT and KCICSJCT (or XJCT=class_name for user-defined resource class names).

Access authorization levels

If you specify security with SEC=YES as a system initialization parameter, CICS requires a minimum authorization of UPDATE for journal access.

Started and XPCT-checked transactions

A CICS transaction initiated by a terminal user can start other transactions by means of an EXEC CICS START command. Transactions started in this way are known as *started transactions*, and you can use CICS RACF security to control who can start other transactions using the START command.

Started transactions are defined in the ACICSPCT and BCICSPCT resource class profiles. These profiles also control access to transactions specified in certain other EXEC CICS commands, if the transaction issuing the command is defined with RESSEC(YES). The commands affected are:

- COLLECT STATISTICS TRANSACTION
- DISCARD TRANSACTION
- INQUIRE TRANSACTION
- SET TRANSACTION
- INQUIRE REQID
- CANCEL

When a transaction issues an EXEC CICS START TRANSID(tranid) command, CICS calls RACF to check that the user of the transaction issuing the command is authorized for the started transaction.

To implement security for started transactions and for transactions checked against the XPCT class, you must do the following:

1. Specify RESSEC(YES) in the CSD resource definition of the transactions that issue START commands.
2. Define profiles to RACF in the ACICSPCT or BCICSPCT resource classes (or their equivalent if you have user-defined resource class names) using the name of the started transaction to identify the profiles.

   For example, use the following command to define a transaction in the ACICSPCT class, and authorize one user only:

   ```
   RDEFINE  ACICSPCT (tran1, tran2, ..., trann) UACC(NONE)
                    NOTIFY(sys_admin_userid)
   PERMIT  tran1 CLASS(ACICSPCT)  ID(userid) ACCESS(READ)
   PERMIT  tran2 CLASS(ACICSPCT)  ID(userid) ACCESS(READ)
   ```

   To define started transactions as members of a profile in the started transaction resource group class, with an appropriate access list, use the following commands:

   ```
   RDEFINE  BCICSPCT  started_trans  UACC(NONE)
                      ADDMEM(trana, tranb, ..., tranx)
                      NOTIFY(sys_admin_userid)
   PERMIT started_trans  CLASS(BCICSPCT) ID(group_userid) ACCESS(READ)
   ```

3. Specify SEC=YES as a CICS system initialization parameter (and SECPRFX=YES if you define profiles with the CICS region userid as a prefix).
4. Specify XPCT=YES for the default resource class names of ACICSPCT and BCICSPCT (or XPCT=class_name for user-defined resource class names).

Transactions started at terminals

The EXEC CICS START command enables a CICS application program to start another transaction associated with a terminal other than the one from which the start command is issued. For example, the following command issued in CICS transaction *tranid1*, invoked at *termid1*, starts another transaction called *tranid2* at *termid2*:

```
EXEC CICS START
     TRANSID(tranid2)
     AT HOURS('18') MINUTES('50')
     TERMID(termid2)
```

When a TERMID is specified for the started transaction, CICS performs a transaction-attach security check, using the classes TCICSTRN and GCICSTRN, on the userid associated with the terminal (termid2 in this example). You must therefore ensure that the userid associated with the terminal (termid2) is authorized to invoke the transaction. This userid is that of the signed-on user, or the CICS default userid if no user is signed on. If termid2 is *not* authorized, message DFHAC2033 is issued to the user of termid2. The user of the terminal that issued the START command gets a "normal" response. If the started transaction is defined with RESSEC(YES), you must also ensure that the userid associated with the terminal (termid2 in this example) is suitably authorized to access protected resources.

Starting tasks at terminals defined with preset security: Typically, started transactions associated with a terminal are printing tasks, where the specified terminal is a printer. In this case, to associate a specific userid with the terminal, you define the terminal with preset security. See "Preset terminal security" on page 5 for more information.

Transactions started without terminals

The EXEC CICS START command enables a CICS application program to start another transaction that is not associated with any terminal. When no TERMID is specified for the started transaction, the userid associated with the new transaction depends on whether you also specify the USERID option.

Userid of a non-terminal started transaction: The USERID option of the EXEC CICS START command (or the terminal user if no TERMID or USERID is included in the START command) determines the userid for a non-terminal started transaction. Without the USERID option, the non-terminal started transaction has the same userid as the transaction that executed the EXEC CICS START command. If the USERID option is specified on the EXEC CICS START command, the specified userid is used instead.

When an EXEC CICS START command is executed without the TERMID option, CICS performs a surrogate user check to ensure that the transaction is authorized for the userid to be used by the non-terminal started transaction. For information about the link authorization of surrogate users, see "Link security" on page 136. For information about EDF authorization of surrogate users, see "Conditional access processing" on page 26.

Access to resources by a non-terminal started transaction: If the USERID option is not specified on an EXEC CICS START command, the non-terminal started transaction does not always inherit all of the security of the transaction that executed the command. Also, it does not inherit resource access determined by

link security and resource access determined by a userid for EDF when used in dual-screen mode. This means:

- If a transaction-routed transaction executes an EXEC CICS START command, or if an EXEC CICS START command is function shipped, the non-terminal started transaction is not subject to link security.
- If EDF is used in dual-screen mode for a transaction that issues an EXEC CICS START command, the non-terminal started transaction is not subject to resource access determined by the userid of the EDF terminal.

If you want the started transaction to have exactly the same security capabilities as the starting transaction, you should omit the USERID option. Without the USERID option, resource access by the non-terminal started transaction is determined by the signon parameters of the terminal transaction. These include the RACF group and the port of entry at which the terminal user signed on; that is, the terminal or console used to sign on, as shown in the following example:

> A terminal user signs on using the CESN transaction at a terminal with netname 'NETNAMEX'. For RACF, therefore, the port of entry is 'NETNAMEX'. At the CESN screen the terminal user enters userid 'USERID1' and groupid 'GROUPID2'. The terminal user then runs a terminal transaction which executes an EXEC CICS START command without the TERMID option or the USERID option specified. The non-terminal started transaction has resource access determined by userid 'USERID1', groupid 'GROUPID2', and port of entry 'NETNAMEX'.

If a non-terminal transaction is denied access to a resource by RACF, the error message produced can include the terminal signon parameters, userid and groupid. It can also include a port of entry. The userid, groupid, and port of entry can be those inherited from a terminal transaction which started the non-terminal transaction.

If the USERID option is specified on an EXEC CICS START command, the non-terminal started transaction has access to resources determined by the userid specified on the USERID option.

We recommend that you do not specify the current userid of a terminal transaction on the USERID option. The non-terminal started transaction may not have the same resource access as the terminal transaction. The following examples show how the non-terminal started transaction can have different resource access:

Example 1:

RACF conditional access lists can be used by specifying WHEN(TERMINAL(...)) or WHEN(CONSOLE(...)) on the RACF PERMIT command to allow a terminal transaction access to certain resources because the specified port of entry is in use. See "Conditional access processing" on page 26.

If an EXEC CICS START TRANSID USERID command is executed by a terminal transaction that specifies the same userid that the terminal user entered when signing on with CESN, the started transaction has access to resources determined by the specified userid, but not to the resources determined by the port of entry.

The started transaction is not subject to the conditional access list which was effective for the terminal transaction that executed the EXEC CICS START USERID command.

Example 2:

Using RACF you can grant (or deny) group access to a RACF protected resource.

A terminal user can enter a groupid and a userid when signing on with CESN. When the terminal user runs a terminal transaction, the groupid can determine resource access.

If an EXEC CICS START TRANSID USERID command is executed by a terminal transaction that specifies the same userid as that entered by the terminal user when signing on with CESN, the started transaction has access to resources determined by the specified userid. Resource access is not determined by the groupid that the terminal user entered when signing on with CESN. Resource access for the non-terminal started transaction can be determined by the default groupid for the specified userid.

The started non-terminal transaction is not subject to the group access which was effective for the terminal transaction that executed the EXEC CICS START USERID command.

Access authorization levels

CICS requires a minimum authorization of READ for started transactions.

Application program security

You control access to the initial program specified in the transaction resource definition by authorizing the user to initiate the transaction (transaction-attach security). However, CICS application programs can invoke other programs by means of the CICS DISABLE, ENABLE, LINK, LOAD, and XCTL commands. Also, the load status of programs can be altered by the CICS RELEASE and DISABLE commands. Note, however, that there is no separate security check on the RELEASE of programs loaded for task lifetime. This is done on the corresponding LOAD.

You control access to programs invoked using these commands by defining profiles in the CICS application program classes, and which you define to CICS on the XPPT system initialization parameter.

To control which users can invoke or change the load status of other programs, you must do the following:

1. Specify RESSEC(YES) in the CSD resource definition of the transactions that use the above commands.
2. Define profiles to RACF in the MCICSPPT or NCICSPPT resource classes (or their equivalent if you have user-defined resource class names) using the name of the program invoked on the LINK, LOAD, or XCTL command to identify the profiles.

For example, use the following commands to define a program in the MCICSPPT class, and authorize one user only:

```
RDEFINE  MCICSPPT  (prog1, prog2, ..., progn)  UACC(NONE)
                  NOTIFY(sys_admin_userid)
PERMIT  prog1 CLASS(MCICSPPT)  ID(userid) ACCESS(READ)
PERMIT  prog2 CLASS(MCICSPPT)  ID(userid) ACCESS(READ)
```

To define programs as members of a profile in the application program resource group class, with an appropriate access list, use the following commands:

```
RDEFINE  NCICSPPT  cics_programs  UACC(NONE)
                   ADDMEM(proga, progb, ..., progx)
                   NOTIFY(sys_admin_userid)
PERMIT cics_programs CLASS(NCICSPPT) ID(group_userid) ACCESS(READ)
```

3. Specify SEC=YES as a CICS system initialization parameter (and SECPRFX=YES if you define profiles with the CICS region userid as a prefix).
4. Specify XPPT=YES as a CICS system initialization parameter for the default resource class names of MCICSPPT and NCICSPPT (or XPPT=class_name for user-defined resource class names).

Exception for distributed program link (DPL) commands

If CICS finds that a program referenced on an EXEC CICS LINK command is a remote program, it does not perform the security check in the region in which the link command is issued. The security check is performed only in the CICS region in which the linked-to program finally executes.

For example, if CICSA function ships a DPL command to CICSB, where the program then executes, CICSB issues the security check. If the DPL request is function shipped again to CICSC for execution, it is CICSC that issues the security check.

Access authorization levels

CICS requires a minimum authorization of READ for programs.

Temporary storage

Unlike the other resources for which you specify RESSEC(YES), temporary storage queues for which you require RACF protection also require definitions in a CICS control table, the temporary storage table (TST). (You specify them on the DATAID parameter of the DFHTST TYPE=SECURITY macro, as explained in “Other temporary storage security considerations” on page 97.)

Implementing security for temporary storage queues

To implement security for temporary storage queues, you must do the following:

1. Specify RESSEC(YES) in the CSD resource definition of the appropriate transactions.
2. Specify DFHTST TYPE=SECURITY entries in the CICS temporary storage table for the queues on which you want CICS to perform security checking. CICS does not perform any security checks on temporary storage queues that are not defined by TYPE=SECURITY entries in the TST.

3. Define profiles to RACF in the SCICSTST or UCICSTST resource classes (or their equivalent if you have user-defined resource class names), with access lists as appropriate. The resource names you define must correspond to the full temporary storage queue name. Use the following commands to define queues in the SCICSTST class, and authorize users to both read and write to these queues:

   ```
   RDEFINE  SCICSTST  (tsqueue1, tsqueue2, ..., tsqueuen) UACC(NONE)
                      NOTIFY(sys_admin_userid)
   PERMIT tsqueue1 CLASS(SCICSTST) ID(group1, group2) ACCESS(UPDATE)
   PERMIT tsqueue2 CLASS(SCICSTST) ID(group1, group2) ACCESS(UPDATE)
   ```

   To define temporary storage queues as members of a profile in the CICS temporary storage resource group class, with an appropriate access list, use the following commands:

   ```
   RDEFINE  UCICSTST  tsqueue_group  UACC(NONE)
                      ADDMEM(tsqueuea, tsqueueb, ..., tsqueuex)
                      NOTIFY(sys_admin_userid)
   PERMIT tsqueue_group CLASS(UCICSTST) ID(group_userid) ACCESS(UPDATE)
   ```

   For more information about defining temporary storage profiles, see “Other temporary storage security considerations.”

4. Specify SEC=YES as a CICS system initialization parameter (and SECPRFX=YES if you define profiles with the CICS region userid as a prefix).
5. Specify XTST=YES as a CICS system initialization parameter for the default resource class names of SCICSTST and UCICSTST (or XTST=class_name for user-defined resource class names).

Other temporary storage security considerations

You can define the queue names on the DATAID parameter of the DFHTST TYPE=SECURITY macro as follows:

- By specifying a fully identified name that exactly matches the queue name specified on a READQ TS or WRITEQ TS command. This can be from 1 to 8 alphanumeric characters.
- By specifying a generic name, or prefix, that corresponds to the leading alphanumeric characters of a set of queue names.

  It follows that a prefix can only be from 1 to 7 characters, because if you specify the maximum for a queue name it must be the name of a specific temporary storage queue.

When a CICS application issues a temporary storage command (for example, DELETEQ TS, READQ TS or WRITEQ TS) and temporary storage security is in effect, CICS searches the TST for a DATAID that corresponds to the leading characters of the queue name.

Note that if you include a temporary storage queue with hexadecimal characters in a temporary storage queue name, unpredictable results may occur. Also, if a TSQ name contains an imbedded blank, RACF truncates the resource name to that blank.

Access authorization levels

If you specify security with SEC=YES as a system initialization parameter, CICS requires a level of authorization appropriate to the temporary storage queue access intended: a minimum of READ for READQ TS, and a minimum of UPDATE for DELETEQ TS and WRITEQ TS.

Program specification blocks

DL/I program specification blocks (PSBs) are IMS control blocks that describe databases and logical message destinations used by an application program. PSBs consist of one or more program communication blocks (PCBs), which describe an application program's interface to an IMS database.

To implement security for PSBs scheduled in CICS applications, you must:

1. Define profiles to RACF in the PCICSPSB or QCICSPSB resource classes (or their equivalent if you have user-defined resource class names), with access lists as appropriate. The resource profile names you define to RACF must correspond to the names of PSBs specified in CICS PSB schedule commands. For example, use the following commands to define PSBs in the PCICSPSB class, and authorize users to access these queues:

   ```
   RDEFINE  PCICSPSB  (psbname1, psbname2, ..., psbnamen) UACC(NONE)
                      NOTIFY(sys_admin_userid)
   PERMIT psbname1 CLASS(PCICSPSB) ID(group1, group2) ACCESS(READ)
   PERMIT psbname2 CLASS(PCICSPSB) ID(group1, group2) ACCESS(READ)
   ```

   However, RESSEC(NO) does not apply to PSB checks. This parameter is ignored. To define PSBs as members of a profile in the CICS PSB resource group class, with an appropriate access list, use the following commands:

   ```
   RDEFINE  QCICSPSB  psbname_group UACC(NONE)
                      ADDMEM(psbnamea, psbnameb, ..., psbnamex)
                        NOTIFY(sys_admin_userid)
   PERMIT psbname_group CLASS(QCICSPSB) ID(group_userid) ACCESS(UPDATE)
   ```

2. Specify SEC=YES as a CICS system initialization parameter (and SECPRFX=YES if you define profiles with the CICS region userid as a prefix).
3. Specify XPSB=YES as a CICS system initialization parameter for the default resource class names of PCICSPSB and QCICSPSB (or XPSB=class_name for user-defined resource class names).
4. Specify PSBCHK=YES if you want full security for PSBs that are accessed in transaction-routed transactions. This applies to all types of DL/I interface (local, remote, and DBCTL). If you specify PSBCHK=NO, the authority of the remote user is *not used* in transaction-routed transactions.

Note: CICS requires a minimum authorization of READ for PSBs.

If you are using DBCTL, you will also need to read the chapter on security in the *CICS/ESA CICS-IMS Database Control Guide* for information on defining security in a CICS-DBCTL environment.

Security checking of transactions running under CEDF

When a transaction is run under the CEDF transaction, the security processing for the target transaction is determined from the logical OR of RESSEC in the resource definitions for the target transaction and the CEDF transaction.

Table 10 shows the security checking performed for the transaction XSUB for different settings of RESSEC.

Table 10. Security checking of transactions running under CEDF

CEDF	XSUB	Security checking
RESSEC(YES)	RESSEC(YES)	All access to CICS resources cause a security check.
RESSEC(YES)	RESSEC(NO)	All access to CICS resources cause a security check. (Logical OR results in RESSEC on.)
RESSEC(NO)	RESSEC(YES)	All access to CICS resources cause a security check. (Logical OR results in RESSEC on.)
RESSEC(NO)	RESSEC(NO)	Access to CICS resources do not cause a security check. (Logical OR results in RESSEC off.)

To achieve the expected security processing for a transaction when it runs under CEDF, ensure that RESSEC for the CEDF transaction definition is set to NO. CEDF is supplied with RESSEC(YES). Therefore, to change the definition, copy it to another group.

When CEBR and CECI are invoked from within EDF they are transaction-attach checked. The CMDSEC and RESSEC definitions are forced when CEBR or CECI are invoked in this environment, regardless of what is coded in their transaction definitions

When CEDF is used in **two-terminal mode**, it is entered at a different terminal from the transaction being tested. The authorities of the user executing the CEDF transaction are taken into account, as well as those of the user executing the transaction being tested. For each resource accessed by the tested transaction, both users must have access authority, otherwise a NOTAUTH condition is raised. This applies to all resource checks:

- Transaction attach
- CICS resource
- CICS command
- Non-CICS resources accessed through the QUERY SECURITY command
- Surrogate user.

Defining generic profiles for resources

If you control access to CICS transactions by means of transaction-attach security, there is probably only a very small subset of other resource types for which you need a further level of RACF protection. For example, there may be just a few programs in the CICS application program resource class that are particularly sensitive, and a much larger number that constitute no significant risk. In this case, you could protect the few by defining specific RACF profiles for only that subset of resources that are sensitive. You ensure that everyone can access the remaining, nonsensitive, programs by defining a completely generic resource profile, as follows:

```
RDEFINE  MCICSPPT  *  UACC(READ) ...
```

This profile applies to any authorization request for programs not covered by one of the specific profiles. RACF processing logic is such that the most specific profile for any given resource name is always used.

Note that to determine whether a profile is generic, you need check if 'G' appears after the name of the profile when it is listed with RLIST or SEARCH. For example:

SEARCH CLASS(TCICSTRN)

may give the following output:

```
C*
CED% (G)
** (G)
```

The above output shows that both CED% and ** are generic profiles. The C* profile is not generic because it is not followed by (G). The C* profile can be deleted and redefined as a proper generic profile as follows:

```
SETROPTS NOGENERIC(TCICSTRN)
SETROPTS NOGENCMD(TCICSTRN)
RDEL TCICSTRN C*
SETROPTS GENERIC(TCICSTRN)
RDEFINE TCICSTRN C* UACC(NONE)
```

Access to all or access to none?

If RACF cannot find either a specific or generic profile, it returns a “no profile found” condition. However, for the APPL class it returns READ access intent. CICS treats this return code exactly the same as the “user not authorized” return code, and returns the NOTAUTH condition to the CICS application program.

You can either use the completely generic profile to permit access to any resources not otherwise covered by more specific profiles or, to prevent any access, use the UACC(READ|UPDATE) or UACC(NONE) options. For example,

```
RDEFINE  DCICSDCT  *  UACC(NONE)
```

prevents access to any transient data queue not covered by any of the other profiles defined to RACF, and results in RACF writing an SMF record.

On the other hand, you can define files as “public” by the following command:

```
RDEFINE  FCICSFCT  *  UACC(READ)
```

If you are using generic profiles, ensure that generic profile checking has been activated for the CICS RACF resource classes (both the IBM-supplied classes and any installation-defined classes added to the RACF class descriptor table) by issuing a SETROPTS GENERIC(classname) command for any one of the CICS classes having the same POSIT value. This ensures generic checking for all other CICS classes with the same POSIT value. If you change a generic profile, you must issue a SETROPTS GENERIC(classname) REFRESH command. For more information about POSIT values and defining generic classes, see the *System Programming Library: RACF* manual.

Chapter 7. Surrogate user security

This chapter is in two main sections:

- "Where surrogate user checking applies"
- "RACF definitions for surrogate user checking" on page 105

Where surrogate user checking applies

CICS performs surrogate user security checking in a number of situations, using the surrogate user facility of an external security manager (ESM) such as RACF. A surrogate user is one who has the authority to start work on behalf of another user. A surrogate user is authorized to act for that user without knowing that other user's password. To enable surrogate user checking, XUSER=YES must be specified as a system initialization parameter.

If surrogate user checking is in force, it applies to the following:

- The CICS default user
- PLT post-initializing processing
- Preset terminal security
- Started transactions
- The userid associated with a transient data destination.

CICS default user

CICS performs a surrogate user security check against its own userid (the CICS region userid) to ensure that it is properly authorized as a surrogate of the default userid specified on the DFLTUSER system initialization parameter.

Post-initialization processing

If you specify a program list table on a PLTPI system initialization parameter, CICS checks that the region userid is authorized as a surrogate user of the userid specified in the PLTPIUSR system initialization parameter.

The PLTPIUSR system initialization parameter specifies the userid that CICS is to use for PLT programs that run during CICS initialization. All PLT programs run under the authority of the specified userid, which must be authorized to all the resources referenced by the programs.

The scope of PLT security checking is defined by the PLTPISEC parameter. This specifies whether command security checks and resource security checks are to apply to PLTPI programs.

If you do not specify the PLTPIUSR parameter, CICS runs PLTPI programs under the authority of the CICS region userid, in which case CICS does not perform a surrogate user check. However, the CICS region userid must then be authorized to all the resources referenced by the PLT programs. Furthermore, the CICS region userid is associated with any transactions started by PLT programs, and therefore must be authorized to run such transactions.

Preset terminal security

When you install a terminal that is defined with a preset security userid, CICS checks that the userid performing the install is authorized as a surrogate user of the preset userid. This is discussed in “Controlling the use of preset security” on page 68.

Started transactions

CICS performs surrogate user checks when you use the EXEC CICS START command to start a transaction that is not associated with a terminal.

In the following, the userid under which the transaction issuing the START command runs is called the *starting-userid*, and the userid under which the started transaction runs is called the *started-userid*:

- If the TERMID option is specified on the START command, surrogate user checking does not apply. The *started-userid* is inherited from the terminal at which the transaction runs.
- If the USERID option is specified on the START command, the *started-userid* is set to that specified userid.
- If neither TERMID nor USERID is specified on the START command, the *started-userid* is set the same as the *starting-userid*.

CICS requires that all the userids associated with the transaction issuing the START are surrogates of the *started-userid*. CICS also assumes that any userid is always a surrogate of itself. So userids that are the same as *started-userid* are regarded as surrogates already, and the external security manager is not called for them.

A transaction can be associated with userids that are different from *starting-userid* when it is using CICS intercommunication, and when it is using EDF in the two-terminal mode.

Intercommunication and started transactions

If an EXEC CICS START command (without TERMID) is function shipped or is executed from a transaction-routed transaction, the command can be subject to link security. If link security is in effect, CICS also performs a surrogate user check to verify that the userid for link security is authorized as a surrogate user to the userid for the started transaction. The surrogate check is done at this stage even if the USERID is omitted (if the *started-userid* is different from the link userid). For more information see “Link security” on page 136.

EDF in dual-screen mode and started transactions

If an EXEC CICS START command (without TERMID) is executed under control of EDF in dual-screen mode, CICS also performs a surrogate user check, to verify that the userid for the EDF terminal is authorized as a surrogate user of the userid for the started transaction. This check is done even if USERID is omitted, if the *started-userid* is different from the EDF userid.

Surrogate user checking can be subject to link security, If EDF is in use in dual-screen mode, the security of the user executing EDF is also checked. If a NOTAUTH condition occurs with an EXEC CICS START command, this can be due to link security or EDF user security.

Transient data trigger-level transactions

When a transient data queue is defined with a non-terminal trigger-level transaction and a USERID parameter, CICS checks that its own userid (the CICS region userid) is authorized as a surrogate user of the userid specified of the trigger-level transaction.

The userid for a transient data trigger-level transaction that is not associated with a terminal can be specified with the DFHDCT macro resource definition and the EXEC CICS SET TDQUEUE system programming command.

DFHDCT TYPE=INTRA USERID

CICS uses the userid specified on DCT macros for security checking in any trigger-level transactions that are not associated with a terminal. Code this with the userid that you want CICS to use for security checking for the trigger-level transaction specified on the TRANSID operand. USERID is valid only when the destination is defined as DESTFAC=FILE.

The trigger-level transaction runs under the authority of the specified userid, which must be authorized to all the resources used by the transaction.

If you omit the userid from a qualifying trigger-level entry, CICS uses the userid specified on the TYPE=INITIAL macro. If you also omit the userid from the TYPE=INITIAL macro, CICS uses the CICS default userid, specified on the DFLTUSER system initialization parameter. You must ensure that the CICS region userid of any CICS region in which this DCT is installed is defined as a surrogate of all the userids specified in the DCT. This is because, during initialization, CICS performs a surrogate user security check for the CICS region userid against all the userids specified in DFHDCT definitions. If the surrogate security check fails, CICS deactivates automatic transaction initiation by trigger-level for the intrapartition queue for which the surrogate check failed.

EXEC CICS SET TDQUEUE ATIUSERID

The system programming command, EXEC CICS SET TDQUEUE ATIUSERID, specifies the userid for a transient data trigger-level transaction that is not associated with a terminal. The ATIFACILITY must be NOTERMINAL.

CICS performs a surrogate user security check against the userid of the transaction that issues the EXEC CICS SET TDQUEUE command, to verify that the transaction userid is authorized as a surrogate user of the userid specified on the ATIUSERID parameter.

RACF definitions for surrogate user checking

To enable CICS surrogate user checking, you must:

- Define the appropriate SURROGAT class profiles for CICS in the RACF database.
- Authorize CICS surrogate users to the appropriate SURROGAT profiles.

There are two forms of surrogate class profile names that you can define for CICS surrogate user checking. The names of these SURROGAT class profiles must conform to the following naming conventions:

userid.**DFHSTART**

These profile names must be of the form *userid* concatenated with DFHSTART, where *userid* is the userid under which a started transaction is to run.

userid.**DFHINSTL**

These profile names must be of the form *userid* concatenated with DFHINSTL, where *userid* represents one of the following:

- The PLT userid specified on the PLTPIUSR system initialization parameter
- The userid associated with a trigger-level transaction
- The CICS default userid specified on the DFLTUSER system initialization parameter
- The userid specified for preset terminal security.

To authorize a surrogate user to one of these profiles, you must grant READ access.

It is never necessary to define a user as that user's own surrogate. CICS bypasses the surrogate check in this case.

The *RACF Security Administrator's Guide* gives more information about defining surrogate resource classes. You are recommended to refer to it if you need to use RACF facilities such as generic resource classes or RACFVARS profiles to help with making many RACF definitions.

Examples of RACF definitions for surrogate user checking

You define surrogate users to RACF by:

- Defining a *userid.resource_name* profile in the SURROGAT general resource class for each user that requires a surrogate user to act on their behalf. For this purpose you use the RACF RDEFINE SURROGAT command.
- Authorizing each userid that is to act as a surrogate for a user defined in a SURROGAT class profile. For this purpose you use the RACF PERMIT command.

PLT security: For PLT security checking, the CICS region userid must be authorized as a surrogate of the PLT userid defined on the PLTPIUSR system initialization parameter. This means granting the CICS region userid access to a SURROGAT resource class profile owned by the PLT userid, as shown in the following example, where the CICS region userid is CICSHT01, and the PLT security userid is PLTUSER:

```
RDEFINE  SURROGAT  PLTUSER.DFHINSTL  UACC(NONE) OWNER(PLTUSER)
PERMIT   PLTUSER.DFHINSTL  CLASS(SURROGAT) ID(CICSHT01)  ACCESS(READ)
```

In addition to enabling PLT security by defining SURROGAT profiles, you must ensure that when PLT security is active (through the use of the PLTPISEC system initialization parameter) you also add the PLT userid to the access lists of all the resources accessed by PLT programs. For example, if you specify PLTPISEC=RESSEC, you must ensure that the PLT userid is authorized to all the CICS resources for which security is active.

Started transactions: For started transactions, CICS can require as many as three levels of surrogate user. (See “Started transactions” on page 104 for details of the different surrogate users that can be required for a START command.)

For started transaction security at the first level, the userid of the transaction that issues the START command must be authorized as a surrogate for the userid specified on the START command.

For example, a transaction running under USERID2 issues:

```
EXEC CICS START TRANSID('TBAK') USERID('USERID1').
```

USERID2 must be defined to RACF as a surrogate of USERID1 (with READ authority). This is illustrated in the following RACF commands:

```
RDEFINE  SURROGAT  USERID1.DFHSTART  UACC(NONE)  OWNER(USERID1)
PERMIT   USERID1.DFHSTART  CLASS(SURROGAT) ID(USERID2)  ACCESS(READ)
```

For more information about surrogate security, see “Querying a user’s surrogate authority” on page 123.

Chapter 8. CICS command security

CICS command security applies to System Programming (SP)-type commands; that is, commands that require the special CICS translator option, SP. Security checking is performed for these commands when they are issued from a CICS application program, and for the equivalent commands that you can issue with the CEMT master terminal transaction. The commands subject to command security checking can be seen in Table 11.

This chapter discusses security for these commands as follows:

- The commands and objects that are subject to command security checking
- The parameters you need to specify to activate command security checking in your CICS regions
- Special considerations for CEMT
- Authorization failures.

CICS/ESA Front End Programming Interface security uses the same mechanism for authorization as the SP-type commands, using the FEPIRESOURCE resource name. Front End Programming Interface security is not discussed in this book. See the *CICS/ESA Front End Programming Interface User's Guide* for details.

Table 11. Access required for system programming commands

Access required	Command name
READ	COLLECT INQUIRE
UPDATE	PERFORM SET ENABLE DISABLE EXTRACT RESYNC DISCARD

Note: To determine who is allowed to use the (SP) option on the CICS translator, you can use RACF to control who is allowed to load the DFHEITBS table at translation time. For a description of RACF program control, see the *RACF Security Administrator's Guide*. DFHEITBS is the language definition table that defines the SP-type commands and is loaded only on demand.

CICS resources subject to command security checking

For transaction and resource security checking, you identify the resources to RACF using the identifiers you have assigned to them, such as file names, queue names, transaction names, and so on. However, in the case of command security, the resource identifiers are all predefined by CICS, and you must use these predefined names when defining resource profiles to RACF. The full list of resource identifiers that are subject to command security checking, together with the associated commands, is shown in Table 12 on page 110. Note that most of these commands are common to both the CEMT and EXEC CICS interfaces; where they

are unique to one or the other they are prefaced with **CEMT**, or **EXEC CICS**, as appropriate.

Table 12 (Page 1 of 2). CICS resources subject to command security checking

Resource name (see note 1)	**Related CICS command(s)**
AUTINSTMODEL	INQUIRE|DISCARD AUTINSTMODEL
AUTOINSTALL	INQUIRE|SET AUTOINSTALL
CONNECTION	INQUIRE|SET CONNECTION
DLIDATABASE	**CEMT** INQUIRE|SET DLIDATABASE
DELETSHIPPED	INQUIRE|SET]PERFORM DELETSHIPPED
DSNAME	INQUIRE|SET DSNAME
DUMP	PERFORM DUMP **CEMT** PERFORM SNAP
DUMPDS	INQUIRE|SET DUMPDS
EXITPROGRAM	**EXEC CICS** ENABLE PROGRAM (see note 4) **EXEC CICS** DISABLE PROGRAM (see note 4) **EXEC CICS** EXTRACT EXIT (see note 4) **EXEC CICS** RESYNC ENTRYNAME (see note 4)
FEPIRESOURCE	Certain EXEC CICS FEPI commands (see note 3)
FILE	INQUIRE|SET|DISCARD FILE
IRBATCH	**CEMT** INQUIRE IRBATCH
IRC	INQUIRE|SET IRC
JOURNALNUM	INQUIRE|SET JOURNALNUM
LINE	**CEMT** INQUIRE|SET LINE
MODENAME	INQUIRE|SET MODENAME
MONITOR	INQUIRE|SET MONITOR
PARTNER	INQUIRE|DISCARD PARTNER
PITRACE	**CEMT** INQUIRE|SET PITRACE
PROFILE	INQUIRE|DISCARD PROFILE
PROGRAM	INQUIRE|SET|DISCARD PROGRAM
RECONNECT	**CEMT** PERFORM RECONNECT
REQID	**EXEC CICS** INQUIRE|SET REQID
RESETTIME	PERFORM RESETTIME (see note 5)
SECURITY	PERFORM SECURITY REBUILD
SHUTDOWN	PERFORM SHUTDOWN (see note 2)
STATISTICS	INQUIRE|SET STATISTICS, **EXEC CICS** COLLECT STATISTICS, and PERFORM STATISTICS RECORD
STORAGE	INQUIRE STORAGE
SYSDUMPCODE	INQUIRE|SET SYSDUMPCODE (see note 5)
SYSTEM	INQUIRE|SET SYSTEM
TASK	INQUIRE|SET TASK and TASK LIST
TCLASS	INQUIRE|SET|DISCARD TCLASS and INQUIRE|SET|DISCARD TRANCLASS
TDQUEUE	INQUIRE|SET TDQUEUE
TERMINAL	INQUIRE|SET TERMINAL and NETNAME
TRACEDEST	**EXEC CICS** INQUIRE|SET TRACEDEST
TRACEFLAG	**EXEC CICS** INQUIRE|SET TRACEFLAG

Table 12 (Page 2 of 2). CICS resources subject to command security checking	
Resource name (see note 1)	**Related CICS command(s)**
TRACETYPE	**EXEC CICS** INQUIRE|SET TRACETYPE
TRANDUMPCODE	INQUIRE|SET TRANDUMPCODE (see note 5)
TRANSACTION	INQUIRE|SET|DISCARD TRANSACTION
TSQUEUE	**EXEC CICS** INQUIRE TSQUEUE
VOLUME	INQUIRE|SET VOLUME
VTAM	INQUIRE|SET VTAM

Notes:

1. If you are using prefixing, the CICS region userid must be prefixed to the command resource name.
2. Be particularly cautious when authorizing access to these and any other CICS commands that include a SHUTDOWN option.
3. For more information about FEPI security, see the *CICS/ESA Front End Programming Interface User's Guide*.
4. UPDATE authorization is required to run any transaction that uses these commands.
5. See "Resource names for CEMT" on page 114.

If you are running CICS with command security, define resource profiles to RACF, with access lists as appropriate, using the resource names in Table 12 on page 110 as the profile names. Alternatively, you can create resource group profiles in the VCICSCMD class. In the following example, the RDEFINE command defines a profile named CMDSAMP. The commands protected by this profile are specified on the ADDMEM operand. The PERMIT command allows a group of users to issue the commands for INQUIRE.

```
RDEFINE  VCICSCMD CMDSAMP UACC(NONE)
                  NOTIFY(sys_admin_userid)
                  ADDMEM(AUTINSTMODEL, AUTOINSTALL, CONNECTION,
                         DSNAME, TRANSACTION, TRANDUMPCODE,
                         VOLUME, VTAM)
PERMIT CMDSAMP CLASS(VCICSCMD) ID(operator_group) ACCESS(READ)
```

The following example defines a profile called CMDSAMP1 with the same commands in the ADDMEM operand, as in the previous example. The PERMIT command allows a group of users to issue PERFORM, SET, and DISCARD against these commands.

```
RDEFINE  VCICSCMD CMDSAMP1 UACC(NONE)
                  NOTIFY(sys_admin_userid)
                  ADDMEM(AUTINSTMODEL, AUTOINSTALL, CONNECTION,
                         DSNAME, TRANSACTION, TRANDUMPCODE,
                         VOLUME, VTAM)
PERMIT CMDSAMP1 CLASS(VCICSCMD) ID(op_group_2) ACCESS(UPDATE)
```

If you are running CICS with SEC=YES, users require the access levels shown in Table 13 on page 112.

Table 13. Access levels to CICS commands required for SEC=YES and XCMD not equal to NO	
Command	**Access required**
INQUIRE and COLLECT	For these SP commands you need a minimum of READ access.
All others	For these SP commands you need a minimum of UPDATE access.

Parameters for specifying command security

In addition to the SEC and SECPRFX system initialization parameters, which are described in “SEC” on page 54 and “SECPRFX” on page 54, CICS provides the XCMD system initialization parameter and the CMDSEC resource definition option to enable you to specify that you want command security.

XCMD system initialization parameter

The XCMD security parameter is a CICS system initialization parameter. It allows you to specify whether you want command security active in the CICS region, and optionally to specify the RACF resource class name in which you have defined the command security profiles.

If you are using the IBM-supplied RACF resource class names for CICS command profiles (CCICSCMD and VCICSCMD), specify XCMD=YES. If you specify XCMD=YES, CICS requests RACF to build the in-storage profiles from these default resource classes.

If you are using installation-defined resource class names for CICS command profiles, specify XCMD=user_class, and CICS requests RACF to build the in-storage profiles from your own installation-defined resource classes.

If you do not want command security in a CICS region, specify XCMD=NO.

The CMDSEC system initialization parameter

You can force the effect of CMDSEC=YES for all CICS transactions by specifying the CMDSEC=ALWAYS system initialization parameter. In general, this is not recommended for the following reasons:

- For most simple transactions, just controlling access to the transaction is enough to control everything that the transaction can do.
- Invoking a command check for every CICS command consumes extra overhead that reduces the performance of all your transactions.

The CMDSEC option is recommended for installations that need total control of the SP-type commands.

The CMDSEC transaction definition parameter

As described, the XCMD parameter enables command security to be active. You specify which transactions you want command security to apply to by using the CMDSEC option on the transaction resource definition, as follows:

CMDSEC(NO) You do not want command security checking on the transaction.

CMDSEC(YES) You want command security checking on the SP commands in Table 11 on page 109.

For each of these commands issued in a user application, or by the CICS-supplied transactions CEMT and CECI, CICS calls RACF to check that the terminal operator who initiated the transaction has authority to use the command for the specified resource.

Security checking of transactions running under CEDF

When a transaction is run under the CEDF transaction, the security processing for the target transaction is determined from the logical OR of CMDSEC in the resource definitions for the target transaction and the CEDF transaction.

Table 14 shows the security checking performed for the transaction XSUB for different settings of CMDSEC.

Table 14. Security checking for transactions running under CEDF

CEDF	XSUB	Security checking
CMDSEC(YES)	CMDSEC(YES)	All access to CICS commands cause a security check.
CMDSEC(YES)	CMDSEC(NO)	All access to CICS commands cause a security check. (Logical OR results in CMDSEC on.)
CMDSEC(NO)	CMDSEC(YES)	All access to CICS commands cause a security check. (Logical OR results in CMDSEC on.)
CMDSEC(NO)	CMDSEC(NO)	Access to CICS commands do not cause a security check. (Logical OR results in CMDSEC off.)

To achieve the expected security processing for a transaction when it runs under CEDF, ensure that CMDSEC for the CEDF transaction definition is set to NO. CEDF is supplied with RESSEC (YES) and CMDSEC(YES). To change the definitions, copy them to another group.

When CEBR and CECI are invoked from within EDF they are transaction-attach
checked. In the same environment the CMDSEC and RESSEC definitions are
forced regardless of what is coded in their transaction definitions.

When CEDF is used in **two-terminal mode** (the CEDF is entered at a different terminal from the transaction being tested), the authorities of the user executing the CEDF transaction are taken into account, as well as those of the user executing the transaction being tested. For each resource accessed by the tested transaction, both users must have access authority, otherwise a NOTAUTH condition is raised. This applies to all resource checks:

- Transaction attach
- CICS resource
- CICS command
- Non-CICS resources accessed through the QUERY SECURITY command
- Surrogate user.

Note: When EXEC CICS SIGNON, EXEC CICS VERIFY PASSWORD, and EXEC CICS CHANGE PASSWORD commands are issued by a transaction running under CEDF, the password (and new password, where applicable) is blanked out.

CEMT considerations

In general, the resources that the CICS-supplied CEMT master terminal transaction operates on are the same as the equivalent SP-type commands shown in Table 11 on page 109 of the CICS API. If, in addition to normal transaction-attach security, you are using command security, you must ensure that authorized users of CEMT are also authorized for the CICS commands, as appropriate. If a user is authorized to initiate the CEMT transaction, but is not authorized for the resources on which the SP commands in Table 11 on page 109, CICS returns a NOTAUTH condition. To allow your system programmers to use the CEMT command in a command security environment, give them UPDATE access to the group profile protecting commands on which you want them to issue the PERFORM, SET, and DISCARD commands. UPDATE authority should be given to users specifying XPPT=YES and XCMD=YES when they issue a CEMT SET PROG(xxx) NEWCOPY command. You should provide READ access to the group profile protecting the commands on which you want the user to issue only INQUIRE and COLLECT commands.

```
PERMIT profile_name CLASS(VCICSCMD) ID(user or group) ACCESS(READ)
PERMIT profile_name CLASS(VCICSCMD) ID(user or group) ACCESS(UPDATE)
```

Note: If you are using RACF 1.9, and you are starting CICS with security for the first time, ensure that you or your system programmer can run a CEMT PERFORM SECURITY REBUILD. If not, you will have to restart CICS if you need to refresh the resource profiles. For information on the PERFORM SECURITY REBUILD command, see “Refreshing resource profiles in main storage” on page 29.

Resource names for CEMT

In general, the resource names of the CEMT commands correspond to the resource names of the equivalent CICS API command. However, there are some exceptions, and in all these cases it is the API resource name that you use to define the security profile to RACF.

- The CEMT system dump option is spelled differently from the EXEC CICS equivalent. CEMT INQUIRE|SET SYDUMPCODE corresponds to EXEC CICS INQUIRE|SET SYSDUMPCODE.
- The CEMT transaction dump option is spelled differently from the EXEC CICS equivalent. CEMT INQUIRE|SET TRDUMPCODE corresponds to EXEC CICS INQUIRE|SET TRANDUMPCODE.
- The CEMT PERFORM RESET option corresponds to the EXEC CICS PERFORM RESETTIME command.
- The AUXTRACE, INTTRACE, and GTFTRACE options of the CEMT INQUIRE and SET commands all correspond to the TRACEDEST option of the API.

To use the CEMT INQUIRE|SET NETNAME command, you need access to the resource TERMINAL, not NETNAME.

Authorization failures

If you are running with CICS command security, CICS returns the NOTAUTH condition (RESP value 70) to your application, which is the same condition as for a resource security failure. (CICS also issues message DFHXS1111 to the CICS security transient data destination CSCS.) To test for this value in your application, we recommend you code DFHRESP(NOTAUTH) rather than explicitly coding a value. To distinguish between a command security failure and a resource security failure, you must check the RESP2 value. For a command security failure, CICS returns a value of 100 in RESP2. For a resource security failure, a value of 101 is returned in RESP2.

For background information on using RESP and RESP2, see the *CICS/ESA Application Programming Guide*; for programming information, see the *CICS/ESA Application Programming Reference*, and the *CICS/ESA System Programming Reference* manuals.

Chapter 9. Security checking using the QUERY SECURITY command

This chapter describes security checking by the user application using the EXEC CICS QUERY SECURITY command.

The EXEC CICS QUERY SECURITY command enables application programs to request from RACF the level of access a user has to a particular resource. The user in this context is the user invoking the transaction that contains the QUERY SECURITY command.

Issuing the QUERY SECURITY command does not actually grant or deny access to a resource (by issuing a NOTAUTH condition), but instead enables the application program to determine what action to take based on the CICS-value data area (CVDA) values that CICS returns. (For programming information on CVDAs, see the *CICS/ESA Application Programming Reference* manual.)

Note: QUERY SECURITY is **not** affected by the RESSEC and CMDSEC keywords on the transaction definition.

There are two distinct forms of the QUERY SECURITY command, depending on the options chosen.

- QUERY SECURITY RESTYPE
- QUERY SECURITY RESCLASS

(For programming information on the QUERY SECURITY command, see the *CICS/ESA Application Programming Reference* manual.)

How the QUERY SECURITY mechanism works

How the QUERY SECURITY mechanism works depends on:

- Whether SEC=YES or SEC=NO is specified in the system initialization parameters
- Whether SECPRFX=YES or SECPRFX=NO is specified in the system initialization parameters
- Which resource classes are active
- Whether the transaction issuing the request is subject to transaction routing, and if so:
  - Which ATTACHSEC parameter was specified on the connection definition
  - For RESTYPE('PSB') only, whether the PSBCHK system initialization parameter is specified as YES or NO.

SEC system initialization parameter

Table 15 assumes that the relevant resource class is active; for example, that XFCT=YES is specified when issuing QUERY SECURITY RESTYPE('FILE').

Table 15. Effect of SEC parameter on QUERY SECURITY commands

SEC	RACF Access	Query Security			
		Read	**Update**	**Control**	**Alter**
YES	NONE READ UPDATE CONTROL ALTER	notreadable readable readable readable readable	notupdatable notupdatable updatable updatable updatable	notctrlable notctrlable notctrlable ctrlable ctrlable	notalterable notalterable notalterable notalterable alterable
NO	n/a	readable	updatable	ctrlable	alterable

SECPRFX system initialization parameter

If SECPRFX=YES is specified, CICS prefixes the resource with the CICS region userid. For example, issuing:

```
QUERY SECURITY RESTYPE('FILE') RESID('PAYFILE')
```

calls RACF to check the terminal user's access to `cics_region_userid.'PAYFILE'` if SECPRFX=YES is specified. If SECPRFX=NO is specified, `'PAYFILE'` is checked.

Resource class system initialization parameters

Table 15 shows how the QUERY SECURITY RESTYPE command works if the system initialization parameter for the relevant resource class (for example, XFCT) system initialization parameter is active. If, however, the relevant X*name* parameter is *not* active (for example, XFCT=NO has been specified), the resource is READABLE, UPDATABLE, CTRLABLE and ALTERABLE.

Transaction routing

When the QUERY SECURITY command is issued from a transaction that has been routed to a remote system, CICS checks the link user's access to the specified resource, and the terminal users access to the resource, if appropriate. For more information, see "Link security with LU6.2" on page 145, "Link security with LU6.1" on page 183, or "Link security with MRO" on page 192 according to the environment you are using.

In order to perform a check against the terminal user as well as the link user when transaction routing a QUERY SECURITY RESTYPE('PSB') RESID(psb_name), the following conditions must both be satisfied:

- ATTACHSEC on the connection definition must not be LOCAL (that is, it can be IDENTIFY, PERSISTENT, MIXIDPE, or VERIFY)
- PSBCHK=YES must be specified as a system initialization parameter in the remote system.

QUERY SECURITY RESTYPE

The QUERY SECURITY RESTYPE command is used for querying access levels to CICS resources (contained in the classes RACLISTed by CICS at initialization). The response to this command indicates the result of a resource check on this resource. If the resource is not defined to RACF, access is not granted by CICS, and the response is NOTREADABLE. Note that responses returned for category 3 transactions may not reflect that there is no attach time (TRANSATTACH) checking performed on category 3 transactions. The length of the resource name passed to RACF with a RESTYPE request should be the actual maximum length for that resource type.

RESTYPE values

RESTYPE is a resource type that corresponds to one of the X*name* system initialization parameters, and can take any of the values shown in Table 16.

Table 16. QUERY SECURITY RESTYPE values

RESTYPE value	**Xname parameter**
FILE	XFCT
JOURNALNUM	XJCT
PROGRAM	XPPT
PSB	XPSB
SPCOMMAND	XCMD
TDQUEUE	XDCT
TRANSACTION	XPCT
TRANSATTACH	XTRAN
TSQUEUE	XTST

RESID values

In all cases (except for the SPCOMMAND resource type), the resource identifiers (RESID values) are defined by your installation.

When defining RESID values, you should be aware of the effects of using blanks (X'40') in resource identifiers. For example, in:

```
QUERY SECURITY RESTYPE('PSB') RESID('A B')
```

the blank delimits the RESID and causes RACF to use a resource name of A.

For SPCOMMAND, the identifiers are predetermined by CICS. Table 17 lists the possible RESID values for SPCOMMAND.

Table 17. RESID values for RESTYPE(SPCOMMAND)

AUTINSTMODEL	MODENAME	SYSTEM
AUTOINSTALL	MONITOR	TASK
CONNECTION	PARTNER	TCLASS
DLIDATABASE	PROFILE	TDQUEUE
DSNAME	PROGRAM	TERMINAL
DUMP	REQID	TRACEDEST
DUMPDS	RESETTIME	TRACEFLAG
EXITPROGRAM	SECURITY	TRACETYPE
FEPIRESOURCE	SHUTDOWN	TRANDUMPCODE
FILE	STATISTICS	TRANSACTION
IRC	STORAGE	TSQUEUE
JOURNALNUM	SYSDUMPCODE	VOLUME
		VTAM

QUERY SECURITY RESTYPE enables application programs to request from RACF the level of access a terminal user has to the specified resource for the environment in which the transaction is running.

Before calling RACF, CICS checks that the resource is installed. If the resource does not exist, CICS does not call RACF and returns the NOTFND condition. However, note that this check is *not* made for PSBs.

When the RESTYPE is TRANSATTACH and the transaction specified on the RESID parameter is unknown in the local region, a NOTFND condition will be returned. However, if dynamic transaction routing is being used, there is no need for the transaction to be installed in the terminal-owning region. The transaction specified on the DTRTRAN system initialization parameter will be attached if an unknown transaction identifier is entered.

Application programmers should be aware that the NOTFND condition does not necessarily indicate that a terminal user will be unable to enter a transaction identifier, because the transaction can be routed dynamically.

Examples of values returned by QUERY SECURITY RESTYPE

This section gives a number of examples of the values returned by QUERY SECURITY RESTYPE, depending on what has been specified in the system initialization parameters.

SEC=NO

When SEC=NO is specified, issuing:

```
QUERY SECURITY RESTYPE('FILE') RESID('PAYFILE') ALTER(alter_cvda)
```

returns:

```
alter_cvda = DFHVALUE(ALTERABLE)
```

because SEC=NO means that no security checking is done for the entire CICS region.

SEC=YES and XFCT=NO

When SEC=YES and XFCT=NO are specified, issuing:

```
QUERY SECURITY RESTYPE('FILE') RESID('PAYFILE') ALTER(alter_cvda)
```

returns:

```
alter_cvda = DFHVALUE(ALTERABLE)
```

because XFCT=NO means that no security checking is done for files.

SEC=YES, XDCT=YES, and SECPRFX=NO

When SEC=YES, XDCT=YES, and SECPRFX=NO are specified, issuing:

```
QUERY SECURITY RESTYPE('TDQUEUE') RESID('TDQ1') READ(read_cvda)
```

returns:

```
read_cvda = DFHVALUE(READABLE)
```

if the user has READ (or higher) access to `'TDQ1'` in the DCICSDCT class or the ECICSDCT group class.

SEC=YES, XTRAN=YES, and SECPRFX=YES

When SEC=YES, XTRAN=YES, and SECPRFX=YES are specified, issuing:

```
QUERY SECURITY RESTYPE('TRANSATTACH') RESID('TRN1') READ(read_cvda)
```

returns:

```
read_cvda = DFHVALUE(NOTREADABLE)
```

if the user *does not* have READ (or higher) access to `cics_region_userid.TRN1` in the TCICSTRN class or GCICSTRN group class.

SEC=YES, XCMD=$USRCMD, and SECPRFX=NO

When SEC=YES, XCMD=$USRCMD, and SECPRFX=NO are specified, issuing:

```
QUERY SECURITY RESTYPE('SPCOMMAND') RESID('VTAM') UPDATE(updt_cvda)
```

returns:

```
updt_cvda = DFHVALUE(UPDATABLE)
```

if the user has UPDATE access (or higher) to `'VTAM'` in the C$USRCMD or V$USRCMD class.

QUERY SECURITY RESCLASS

The QUERY SECURITY RESCLASS command is used when you want to query access levels for non-CICS resources. RESCLASS is the name of a valid RACF general resource class, such as TERMINAL, FACILITY, or a similar installation-defined resource class. See “Other IBM-supplied RACF resource class names affecting CICS” on page 29. The class name identified by RESCLASS is treated literally, with no translation.

Note: The RACF classes DATASET, GROUP and USER do not appear in the class descriptor table (CDT), which means that you cannot query against these classes.

Prefixing, as specified in the SECPRFX system initialization parameter, does not apply to QUERY SECURITY RESCLASS. That is, CICS does *not* prefix the RESID with the CICS-region userid before calling RACF.

If SEC=NO is specified in the system initialization parameters, QUERY SECURITY RESCLASS always returns READABLE, UPDATABLE, CTRLABLE and ALTERABLE.

For QUERY SECURITY RESCLASS, both the RESID *and* the RESIDLENGTH option must be specified. The maximum length of a resource (RESID) within a RACF class is specified in the class descriptor table (CDT). When defining RESID values, you should be aware of the effects of including blanks (X'40') in RESIDs. For example, in:

```
QUERY SECURITY RESCLASS('MYCLASS') RESID('MY PROFILE') RESIDLENGTH(10)
```

the presence of a blank causes an INVREQ condition. This is because RACF does not allow blanks to be embedded in a profile name.

Because CICS rejects access on a resource request from RACF that is not found, NOTFND is returned in response to RESTYPE.

Note: To determine access to CICS resources you should normally use RESTYPE, when the resource class is determined by the *Xname* system initialization parameter. However, if for special reasons, you want to inquire about specific CICS resource classes, you should note that the class name must be the member class, and *not* the group class; that is, CCICSCMD, and not VCICSCMD. The profiles in the grouping class are checked automatically if the member class has been RACLISTed. For example, if SEC=YES, and XCMD=YES are specified, both CCICSCMD and VCICSCMD are RACLISTed in the CICS region, which means that QUERY SECURITY RESCLASS('CCICSCMD') checks profiles in both CCICSCMD and VCICSCMD.

CICS can RACLIST groups only if the relevant X*name* classes are active (for example, XCMD=YES or XCMD=$USRCMD).

Issuing QUERY SECURITY RESCLASS('TERMINAL') checks profiles in both TERMINAL and GTERMINL (the terminal grouping class) only if the TERMINAL class has been RACLISTed at the system level by the command:

```
SETROPTS RACLIST(TERMINAL)
```

For *non-CICS* resource classes, you can issue the SETROPTS RACLIST(classname) command to perform a global RACLIST. See “Specifying user-defined resources to RACF” on page 218 for details.

Querying a user’s surrogate authority

To query a user’s surrogate authorities, you can use the QUERY SECURITY command with RESCLASS('SURROGAT') option. You also need to specify the RESID and RESIDLENGTH options. The RESID value you should provide is described in “RACF definitions for surrogate user checking” on page 105. However, this command is ***not*** controlled by the XUSER system initialization parameter, so you might obtain an unexpected response of NOTREADABLE if XUSER=NO has been specified. For example, to check whether the current user is allowed to start a transaction with a new userid of NEWUSER, when XUSER=YES is specified, issue the command:

```
QUERY SECURITY RESCLASS('SURROGAT') RESID('NEWUSER.DFHSTART')
RESIDLENGTH(16)  READ(read cvda)
```

Logging for QUERY SECURITY RESTYPE and RESCLASS

You can control logging on the QUERY SECURITY command by specifying one of the following options:

- LOG
- NOLOG
- LOGMESSAGE(cvda), where cvda value is LOG=54 or NOLOG=55.

The default is LOG.

If logging is in effect, and the terminal user does not have the requested access to the specified resource, message DFHXS1111 is issued to the CICS security transient data destination CSCS. Where relevant, RACF message ICH408I is also issued. SMF records may also be recorded, depending on the auditing and logging options that have been specified for that resource. For more information, see the *RACF Auditor’s Guide*.

For programming information about CVDAs, refer to the *CICS/ESA System Programming Reference* manual.

Uses for QUERY SECURITY RESTYPE and RESCLASS

You can use the two forms of the QUERY SECURITY command in a number of different ways to customize resource security checking within an application. This section gives a number of examples of doing so.

Changing the level of security checking performed

You can use QUERY SECURITY to perform a different level of security checking from that which CICS would perform for application programs that specify RESSEC(YES) or CMDSEC(YES).

For example, suppose a transaction has RESSEC(YES) and contains a number of EXEC CICS READ FILE commands and a number of EXEC CICS WRITE FILE commands. For every command, CICS performs a security check to ensure that the terminal user has access to the relevant file, even though the same file may be being accessed each time. An alternative to this is to switch off security checking at the transaction level by specifying RESSEC(NO) on the transaction definition and then, when the application starts, execute a command such as:

```
EXEC CICS QUERY SECURITY RESTYPE('FILE') RESID(file_name) UPDATE(cvda)
```

This command allows the transaction to continue without any further calls to RACF.

Note: Switching resource security checking off, using RESSEC(NO) means that **all** resource checks — not just of files as in the above example — are bypassed.

Checking which transactions to offer a user

You can use the QUERY SECURITY command to check whether a user is authorized to use a particular transaction *before* displaying the transaction code as part of an introductory menu. When you use the command for this purpose, you will probably want to avoid logging the checks for users who are not allowed to use certain transactions. To do this, use the NOLOG option.

Example of use of QUERY SECURITY RESCLASS

Normal CICS resource security checking for files operates at the file level only. You can use QUERY SECURITY to enable your application to control access to data at the *record* or *field* level.

To do this, define resource names (which represent record or fields within particular files) with the appropriate access authorizations for the records or fields you want to control. You could define these resources in an installation-defined RACF general resource class and then use the QUERY SECURITY RESCLASS command to check a terminal user's access to a specific field within a file before displaying or updating the field. (The application logic would determine which field.) For example:

```
QUERY SECURITY RESCLASS('$FILEREC') RESID('PAYFILE.SALARY')
RESIDLENGTH(14) READ(read_cvda) NOLOG
```

where '$FILEREC' is an installation-defined RACF general resource class. For more information, see "Designing applications to use the user-defined resources" on page 220.

Chapter 10. Security for CICS-supplied transactions

This chapter discusses security for CICS-supplied transactions, and contains a number of recommendations to ensure that your CICS regions are adequately protected. Where applicable, it describes the recommended security specifications that you will need for the CICS-supplied transactions that are defined in the group list DFHLIST, and stored in the CICS system definition data set (CSD). These recommendations cover all CICS-supplied transactions — those that are intended for use from a user terminal or console, and those that are for CICS internal use only. (For information about the CICS-supplied groups of resource definitions, and the DFHLIST group list, see the *CICS/ESA Resource Definition Guide*.)

It discusses the following:

- Categories of CICS-supplied transactions
- Protecting CICS-supplied transactions.

By default, all CICS transactions are subject to RACF protection (with the exception of category 3 transactions—see “Category 3 transactions” on page 131), unless you run your CICS regions with transaction security switched off. You can do this either by:

- Specifying the system initialization parameter SEC=NO, which switches off all security checking, or
- Specifying the system initialization parameter XTRAN=NO, which switches off transaction-attach security checking only.

There is no parameter on the transaction resource definition that allows you to run with transaction security on some transactions but not others. If you are running
| with transaction security (SEC=YES and XTRAN=YES), CICS issues a security
| check for every transaction attach, other than transactions within category 3, to
| establish whether the user is permitted to run that transaction.

APAR PQ07039

27/10/97 MJO

CICS-supplied transactions CDBN and CSXM are not subject to security checking,
and are exempt from security categorization. Any security definitions for these
transactions are redundant.

Categories of CICS-supplied transactions

For the purposes of this description, we divide the RACF profile definitions that you should specify for your CICS-supplied transactions into three categories. Each transaction is identified within a category that describes its use within CICS. Each category specifies the recommended security specifications you need, in terms of both the CICS transaction definitions and the corresponding RACF profiles.

The three categories contain all of the required CICS transactions, which are generated in their designated groups when you initialize your CICS system definition data set (CSD). The CSD does not include the CICS sample transactions (those that are in groups starting with DFH$). Sample applications should not

require RACF protection, because you are unlikely to install them on a CICS production system.

For information about the security requirements for transactions on earlier releases of CICS, see Chapter 19, “Coexistence with previous CICS releases” on page 235.

Category 1 transactions

Category 1 transactions are never associated with a terminal — that is, they are for CICS internal use only, and should not be invoked from a user terminal. CICS checks that the region userid has the authority to attach these transactions.

However, if the region userid is not authorized to access all of the category 1 transactions, CICS issues message DFHXS1113 and fails to initialize.

For category 1 transactions, specify the following:

To CICS RESSEC(NO) and CMDSEC(NO) on the transaction resource definition

To RACF UACC(NONE) and AUDIT(FAILURES) in the corresponding transaction profiles. AUDIT(FAILURES) is the default and need not be specified. The access list should contain only userids (or groups containing userids) that can be specified as the CICS region userid.

For example:

```
RDEFINE GCICSTRN CICSCAT1 UACC(NONE)
         ADDMEM(CSKP CSPQ CDBD . . . . . CXRE CSNE)
         NOTIFY(security_admin_userid)
         OWNER(userid or groupid)
PERMIT CICSCAT1 CLASS(GCICSTRN) ID(cat1grp1,..,cat1grpz) ACCESS(READ)
```

By defining these transactions to RACF with UACC(NONE), and an access list, you prevent any terminal user initiating these transactions (accidentally or otherwise). It is important that you do this, because permitting the initiation of these transactions at a terminal has unpredictable results. The sample CLIST DFH$CAT1 has been provided to help you define the category 1 profiles to RACF. The sample CLIST can be seen in library *CTS120.CICS520.SDFHSAMP*)Table 18 lists the category 1 transactions.

APAR PQ05860

MJO 01/08/97 (CRMD added)

APAR PN58847 and ERCF 10830

MJO 7/11/94(CRMD added)

Table 18 (Page 1 of 2). Category 1 transactions

CSD group	Transaction	Program invoked	Description
DFHBMS	CSPQ	DFHTPQ	Terminal page cleanup (BMS)
DFHDBCTL	CDBD	DFHDBDI	Provides DBCTL disable function
	CDBO	DFHDBCT	Provides DBCTL control function

Table 18 (Page 2 of 2). Category 1 transactions			
CSD group	**Transaction**	**Program invoked**	**Description**
DFHFEPI	CSZI	DFHSZRMP	Front End Programming Interface
DFHISC	CRSQ	DFHCRQ	Remote schedule purging (ISC)
	CSNC	DFHCRNP	Interregion control program (MRO)
DFHOPCLS	CSFU	DFHFCU	Opens user file-control managed files
DFHRMI	CRSY	DFHRMSY	Resource manager resynchronization program
DFHSIGN	CESC	DFHCESC	Processes time-out and signoff for idle terminals
DFHSPI	CATA	DFHZATA	Autoinstall automatic terminal definition
	CATD	DFHZATD	Autoinstall terminal deletion
	CDTS	DFHZATS	Remote single delete transaction
	CITS	DFHZATS	Remote autoinstall transaction
	CMTS	DFHZATS	Remote mass delete transaction
	CFTS	DFHZATS	Remote mass flag transaction
	CRMD	DFHZATMD	Provides remote mass flag delete transaction
	CRMF	DFHZATMF	Provides remote mass flag transaction
DFHSTAND	CSTE	DFHTACP	Process terminal abnormal conditions
	CXCU	DFHZXCU	Performs XRF tracking catch-up
	CXRE	DFHZXRE	Reconnects terminals following XRF takeover
DFHVTAM	CSNE	DFHZNAC	VTAM node error recovery
N/A	CPLT	DFHSIPLT	Initializes PLT processing
N/A	CSKP	DFHRMXN3	Writes system log activity keypoint
N/A	CSSY	DFHAPATT	Entry point attach
N/A	CGRP	DFHZCGRP	VTAM persistent sessions transaction
N/A	COVR	DFHZCOVR	Open VTAM retry transaction
N/A	CSTP	DFHZCSTP	Terminal control transaction

Category 2 transactions

Category 2 transactions are either initiated by the terminal user, or are associated with a terminal. You should restrict authorizations to initiate these transactions to userids belonging to specific RACF groups.

For the CICS resource definitions, the IBM-supplied transactions are defined with the recommended RESSEC and CMDSEC options. In particular, CECI, CEDF, CEMT, and CEST are all supplied with RESSEC(YES) and CMDSEC(YES). The mirror transactions are defined with RESSEC(YES). If you need to change any of these definitions, you can do so by copying them to another group. We recommend that you do *not* change the supplied definitions of any other transactions.

For most category 2 transactions, you are recommended to specify the following to RACF:

- UACC(NONE) and AUDIT(FAILURES) in the transaction profile.

  AUDIT(FAILURES) is the default, and need not be specified.

- Access list as appropriate.

It is unlikely that you will want to give all users access to all of the transactions in this category, and you should consider defining them in several subcategories. In the examples that follow, the category 2 transactions are further subdivided into a number of groups. Please note that these are only examples. You can choose to group CICS transactions in the ways that best suit your installation's needs.

• SYSADM, containing: CDBT, CBRC, CEMT, CETR, and CEDA

- DEVELOPER, containing: CEDF, CEBR, CECI, CECS, and CEDB
- INQUIRE, containing: CDBI and CEDC
- OPERATOR, containing: CWTO, CRTE, CMSG, CEST, and CEOT
- INTERCOM, containing: CEHP, CEHS, CPMI, CRTE, CSMI, CSM1, CSM2, CSM3, CSM5, and CVMI

  If function shipping is being used, the mirror transactions must be available to remote users in a function shipping environment. When a database or file resides on another CICS region, CICS function ships the request to access the data, and this request runs under one of the CICS-supplied mirror transactions. This means that:

  – The terminal user running the application must be authorized to use the mirror transaction. (See Chapter 5, “Transaction security” on page 77.)

  – The terminal user must also be authorized to use the data that the mirror transaction accesses. (See Chapter 6, “Resource security” on page 83.) The mirror transactions are supplied with RESSEC(YES) defined, so even if the user's transaction specifies RESSEC(NO), the mirror transaction fails if the user is not authorized to access the data.

    If you do not use resource security checking, you must change the mirror transaction definitions to specify RESSEC(NO). Because the mirror transactions are an IBM-protected resource, you must first copy these definitions into your own groups and then change them.

- ALLUSER, containing CMAC and CSGM — the CICS “messages and codes” and “good morning” transactions, respectively. We recommend you define CMAC and CSGM (or, if your installation does not use CSGM, whatever transaction is defined as GMTRAN) as UACC(READ) in a group of their own, because all users need access to them. If your installation uses CSGM as its “good morning” transaction, users who are not authorized to use CSGM will receive message DFHAC2002 when they attempt to use CICS. You should also include your “goodnight” transaction in this group, if you defined one with the GNTRAN system initialization parameter

• WEBUSER, containing: CWBA, CWBC, CWBG, and CWBM

| The sample CLIST DFH$CAT2 has been provided to help you define the category
| 2 profiles to RACF. If you want to use this example setup, review this CLIST and
| make the changes necessary for your installation before running it. If you want to
| use a different setup, you can adapt this CLIST, or provide your own.

Figure 5 shows how to use RDEFINE and PERMIT commands to define the example groups for category 2 transactions.

```
  RDEFINE GCICSTRN SYSADM UACC(NONE)
#          ADDMEM(CDBC,CDBT,CBRC,CEMT,CETR,CEDA)
           NOTIFY(security_admin_userid)
           OWNER(userid or groupid)
  PERMIT   SYSADM  CLASS(GCICSTRN) ID(sysgrp1,..,sysgrpz) ACCESS(READ)
  RDEFINE GCICSTRN DEVELOPER UACC(NONE)
           ADDMEM(CEDF,CEBR,CECI,CECS,CEDB)
           NOTIFY(security_admin_userid)
           OWNER(userid or groupid)
  PERMIT   DEVELOPER  CLASS(GCICSTRN) ID(devgrp1,..,devgrpz) ACCESS(READ)
  RDEFINE GCICSTRN INQUIRE UACC(NONE)
           ADDMEM(CDBI,CEDC)
           NOTIFY(security_admin_userid)
           OWNER(userid or groupid)
  PERMIT   INQUIRE  CLASS(GCICSTRN) ID(inqgrp1,..,inqgrpz) ACCESS(READ)
  RDEFINE GCICSTRN OPERATOR UACC(NONE)
           ADDMEM(CWTO,CRTE,CMSG,CEST,CEOT)
           NOTIFY(security_admin_userid)
           OWNER(userid or groupid)
  PERMIT   OPERATOR  CLASS(GCICSTRN) ID(opsgrp1,..,opsgrpz) ACCESS(READ)
  RDEFINE GCICSTRN INTERCOM UACC(NONE)
           ADDMEM(CEHP,CEHS,CPMI,CSMI,CSM1,CSM2,CSM3,CSM5,CVMI)
           NOTIFY(security_admin_userid)
           OWNER(userid or groupid)
  PERMIT   INTERCOM  CLASS(GCICSTRN) ID(intrgrp1,..,intrgrpz) ACCESS(READ)
  RDEFINE GCICSTRN ALLUSER UACC(READ)
           ADDMEM(CMAC,CRTX,CSGM)
           NOTIFY(security_admin_userid)
           OWNER(userid or groupid)
#  PERMIT   WEBUSER  CLASS(GCICSTRN) ID(webgrp1,..,webgrpz) ACCESS(READ)
#  RDEFINE GCICSTRN WEBUSER UACC(NON)
#           ADDMEM(CWBA,CWBC,CWBG,CWBM)
#           NOTIFY(security_admin_userid)
#           OWNER(userid or groupid)
```

Figure 5. Example of defining groups for category 2 transactions

Notes:

1. With RESSEC(YES) and CMDSEC(YES) defined for these transactions, you must ensure that the user groups authorized to use the transactions are also authorized to access the CICS resources and commands the transactions use.
2. If you protect a resource with a resource group profile, you should avoid protecting the same resource with another profile. If the profiles are different (for example, if they have different access lists), RACF merges the profiles for use during authorization checking. Not only can the merging have a performance impact, but it can be difficult to determine exactly which access

authority applies to a particular user. (See the *RACF Security Administrator's Guide* for further information.)

Table 19 lists the category 2 transactions.

Table 19 (Page 1 of 2). Category 2 transactions

CSD group	Transaction	Program invoked	Description
DFHCMAC	CMAC	DFHCMAC	Displays CICS messages online
DFHCONS	CWTO	DFHCWTO	Writes to console operator
DFHDBCTL	CDBC	DFHDBME	DBCTL interface menu transaction
	CDBI	DFHDBIQ	DBCTL interface inquiry transaction
	CDBM	DFHDBMP	DBCTL operator transaction
	CDBT	DFHDBDSC	DBCTL interface disconnection transaction
DFHDB2	DSNC	DSN2COM1	DB2 attachment facility transaction
DFHDBDSC	CDBT	DBCTL	Provides disconnection transaction
DFHEDF	CEDF	DFHEDFP	Provides execution diagnostic facility
	CEBR	DFHEDFBR	Browse temporary storage
DFHFE	CSFE	DFHFEP	Tests Field Engineering terminal
DFHINDT	CIND	DFHINDT	Provides the in-doubt test tool
DFHINTER	CECI	DFHECIP	CICS command interpreter
	CECS	DFHECSP	Checks CICS command syntax
DFHISC	CEHP	DFHCHS	Provides CICS OS/2 remote server mirror
	CEHS	DFHCHS	Provides CICS/VM remote server mirror
	CPMI	DFHMIRS	Provides CICS OS/2 LU6.2 mirror
	CRTE	DFHRTE	Provides start transaction routing session
	CRTX	N/A	Provides default dynamic routing transaction
	CSMI	DFHMIRS	Provides ISC mirror transaction
	CSM1	DFHMIRS	Provides ISC SYSMSG model
	CSM2	DFHMIRS	Provides ISC scheduler model
	CSM3	DFHMIRS	Provides ISC queue model
	CSM5	DFHMIRS	Provides ISC DL/I model
	CVMI	DFHMIRS	Provides LU6.2 synclevel 1 mirror
DFHMSWIT	CMSG	DFHMSP	Provides message switching
DFHOPER	CEMT	DFHEMTP	Processes master terminal command
	CEOT	DFHEOTP	Inquires on user's own terminal status
	CEST	DFHESTP	Processes supervisor terminal command
	CETR	DFHCETRA	Provides inquire and set trace options

Table 19 (Page 2 of 2). Category 2 transactions			
CSD group	**Transaction**	**Program invoked**	**Description**
DFHSDAP	CESD	DFHCESD	Provides shutdown assist transaction
DFHSPI	CEDA	DFHEDAP	Provides perform resource definition online—full
	CEDB	DFHEDAP	Provides perform resource definition online—restricted
	CEDC	DFHEDAP	Views resource definitions online
DFHVTAM	CSGM	DFHGMM	Provides CICS good morning message
DFHWEB	CWBA	DFHWBA	CICS web interface alias transaction
	CWBC	DFHWBC00	CICS web interface connection manager transaction
	CWBG	DFHWBGB	CICS web interface cleanup transaction
	CWBM	DFHWBM	CICS web interface server controller transaction

Category 3 transactions

Category 3 transactions are either initiated by the terminal user or associated with a terminal. *All CICS terminal users*, whether they are signed on or not, require access to transactions in this category. For this reason, category 3 transactions are exempt from any security check, and CICS permits any terminal user to initiate these transactions.

For category 3 transactions you are recommended to specify RESSEC(NO) and CMDSEC(NO) on the CICS transaction resource definition. These transactions should be defined to RACF, but this definition does not affect actual task attach-time processing. It is used only for QUERY SECURITY purposes.

Table 20 lists the category 3 transactions.

Table 20 (Page 1 of 2). Category 3 transactions			
CSD group	**Transaction**	**Program invoked**	**Description**
DFHHARDC	CSPP	DFHP3270	3270 print support
DFHBMS	CSPG	DFHTPR	BMS terminal paging
	CSPS	DFHTPS	BMS paging transaction scheduler
DFHISC	CLS1	DFHZLS1	ISC LU services model
	CLS2	DFHLUP	ISC LU services model
	CLS3	DFHCLS3	ISC LU services model
	CLS4	DFHCLS4	Password expiry management
	CMPX	DFHMXP	ISC local queuing shipper
	CRSR	DFHCRS	ISC remote scheduler
	CSSF	DFHRTC	Cancel CRTE transaction routing session
	CXRT	DFHCRT	Transaction routing relay

Table 20 (Page 2 of 2). Category 3 transactions			
CSD group	**Transaction**	**Program invoked**	**Description**
DFHRSEND	CSRS	DFHZRSP	3614 message synchronization
DFHSIGN	CESN	DFHSNP	Terminal user signon
	CESF	DFHSNP	Terminal user signoff
	CEGN	DFHCEGN	Goodnight transaction scheduler
DFHSPI	CATR	DFHZATR	Autoinstall restart terminal delete
DFHSTAND	CQRY	DFHQRY	ATI query support
	CSAC	DFHACP	Program abnormal condition processor
DFHVTAMP	CSCY	DFHCPY	3270 screen print
	CSPK	DFHPRK	3270 screen print support
	CSRK	DFHRKB	3270 screen print — release keyboard

Part 3. Intercommunication security

This part discusses the planning and implementing of security in an intersystem communication (ISC) environment, using LU6.2 or LU6.1, or in a multiregion operation (MRO) environment. This part contains the following chapters:

- Chapter 11, “Overview of intercommunication security” on page 135, which introduces the concepts of bind-time, link, user, transaction, and resource security in an intercommunication environment
- Chapter 12, “Implementing LU6.2 security” on page 141, covering bind-time, link, user, transaction, resource, and command security; plus transaction routing, and function shipping
- Chapter 13, “APPC password expiration management” on page 161, which contains information on evaluating and using APPC password expiration management
- Chapter 14, “Implementing LU6.1 security” on page 183, covering link, transaction, resource, and command security; plus function shipping
- Chapter 15, “Implementing MRO security” on page 189, covering bind-time, link, user, transaction, resource, and command security; plus transaction routing and function shipping.

Chapter 11. Overview of intercommunication security

This chapter gives an overview of how security works when CICS systems are interconnected or connected to other compatible systems.

It is organized under the following main topics:

Introduction

In a single CICS system, you use security to make sure that terminal users can access only those parts of the system they need to work with. For interconnected systems, the same basic principles apply, but you must also include definitions for connections, sessions, and partners. You also need to allow for the fact that users of one CICS system can initiate transactions and access resources in another CICS system.

We assume that you are already familiar with setting up security for a single CICS system, as described in Part 1, "Introduction" on page 1 and Part 2, "Implementing RACF protection for a single-region CICS" on page 37.

In particular, you should understand the following concepts:

- User signon. (See "Signon process" on page 63.)
- How the relationship between user security and transaction security determines which transactions a particular user is allowed to invoke. (See Chapter 4, "Verifying CICS users" on page 63 and Chapter 5, "Transaction security" on page 77.)
- How resource security determines which other resources a user is allowed to access. (See Chapter 6, "Resource security" on page 83.)

An interconnected group of CICS systems differs from a single CICS system in that you may have to define a user profile or group profile more than once. (See "RACF user profiles" on page 12 and "RACF group profiles" on page 18 for information on defining these profiles.) That is, you may have to define these profiles in every CICS system that is using a separate RACF database, and in which a user is likely to want to attach a transaction or access a resource. When planning these profiles, you must consider all cases in which a user could initiate function shipping, transaction routing, asynchronous processing, distributed program link, or distributed transaction processing. (For descriptions of these methods of intercommunication, see the *CICS/ESA Intercommunication Guide* and the *CICS/ESA Distributed Transaction Programming Guide*.)

Planning for intercommunication security

Intercommunication security in a CICS system is concerned with incoming requests for access to CICS resources, rather than with requests that are sent to other systems.

The security problem with incoming requests occurs when a particular user at a particular remote system is trying to access resources of your CICS system. Is this access authorized, or should it be rejected?

The following sections describe the points in the processing of an incoming request at which you can apply security checks.

Bind-time security

The first requirement is for a session to be established between the two systems. This does not, of course, happen on every request; a session, once established, is usually long-lived. Also, the connection request that establishes the session can, depending on the circumstances, be issued either by the remote system or by your CICS system. However, the establishment of a session presents the first potential security exposure for your system.

Your security concern is to prevent unauthorized remote systems from being connected to your CICS system; that is, to ensure that the remote system is *really* the system that it claims to be. This level of security is called **bind-time security** (also known as **systems network architecture (SNA) session security**). It can be applied when a request to establish a session is received from, or sent to, a remote system.

Note: We use the term **bind** to refer both to the **SNA BIND** command that is used to establish SNA sessions between systems, and to the **CICS connection request** that is used to establish MRO sessions for CICS interregion communication.

You can specify bind-time security for LU6.2 and multiregion operation (MRO) links, but *not* for LU6.1 links. For information on defining bind-time security in your system, see either “Bind-time security with LU6.2” on page 141 or “Bind-time security with MRO” on page 189, depending on the environment you are using.

Link security

Each link between systems is given an authority defined by a userid.

It is important to note that users cannot access any transactions or resources over a link that is itself unauthorized to access them. This means that each user’s authorization is a subset of the link’s authority as a whole.

To limit the remote system’s access to your transactions and resources, you use **link security**. Link security is concerned with the single user profile that you assign to the remote system as a whole. Like user security in a single-system environment, link security governs:

- **Transaction security**. This controls the link’s authority to attach specific transactions.
- **Resource security**. This controls the link’s authority to access specific resources. This applies for transactions, executing on any of the sessions from

the remote system, that have RESSEC(YES) specified in their transaction definition.

- **Command security**. This controls the link's authority for the commands that the attached transaction issues. This applies for transactions, executing on any of the sessions from the remote system, that have CMDSEC(YES) specified in their transaction definition.
- **Surrogate user security**. This controls the link's authority to START transactions with a new userid, and to install resources with an associated userid.

For more information, see "Transaction, resource, command, and surrogate user security."

Link security with MRO

See the section "Link security with MRO" on page 192, in Chapter 15, "Implementing MRO security" on page 189.

Link security with LU6.2

See the section "Link security with LU6.2" on page 145, in Chapter 12, "Implementing LU6.2 security" on page 141.

Link security for LU6.1

See the section "Link security with LU6.1" on page 183, in Chapter 14, "Implementing LU6.1 security" on page 183.

User security

In addition to the security profile that you set up for the link, you may want to further restrict each remote user's access to the transactions, commands, and resources in your system. This is done by specifying the appropriate ATTACHSEC parameters in the CONNECTION definition. This ***user security***, like link security, distinguishes between transaction, resource, command, and surrogate security. User security can never ***increase*** a user's authority above that of the link. For more information, see "Transaction, resource, command, and surrogate user security."

For information on defining user security in your system, see either "User security with LU6.2" on page 146 or "User security with MRO" on page 193, depending on the environment you are using.

You cannot specify user security for LU6.1 links. For LU6.1, the user security is taken to be the same as the link security.

Transaction, resource, command, and surrogate user security

The last step in defining security for your system is to make sure that the access parameters match the profiles you have defined for your transactions, resources, commands, and surrogate users for the link and the individual remote users. For information on defining these levels of security in a single-system environment, see Chapter 5, "Transaction security" on page 77, Chapter 6, "Resource security" on page 83, and Chapter 8, "CICS command security" on page 109.

Resources and commands are unsecured unless you explicitly request security protection in your transaction definitions.

For information on defining transaction and resource security in your system, see one of the following, depending on the environment you are using:

- "Transaction, resource, and command security with LU6.2" on page 153
- "Transaction, resource, and command security with LU6.1" on page 184
- "Transaction, resource, and command security with MRO" on page 196.

Summary of intercommunication security levels

Figure 6 on page 139 shows bind-time, transaction, resource, and command security, and how CICS enforces these levels of security under the LU6.2, MRO, and LU6.1 protocols. It also shows how the two levels of authorization (user and link) are involved at the three security levels.

For guidance on choosing between these environments, see the *CICS/ESA Intercommunication Guide*.

Note: You also need to define profiles for your resources and users to RACF, as described for single systems in Chapter 2, "RACF facilities" on page 9.

Implementing intercommunication security

Security in the intercommunication environment is implemented through resource definition and RACF profiles. The following chapters tell you how to define your intersystem links, according to the environment you are using:

- Chapter 12, "Implementing LU6.2 security" on page 141
- Chapter 14, "Implementing LU6.1 security" on page 183
- Chapter 15, "Implementing MRO security" on page 189.

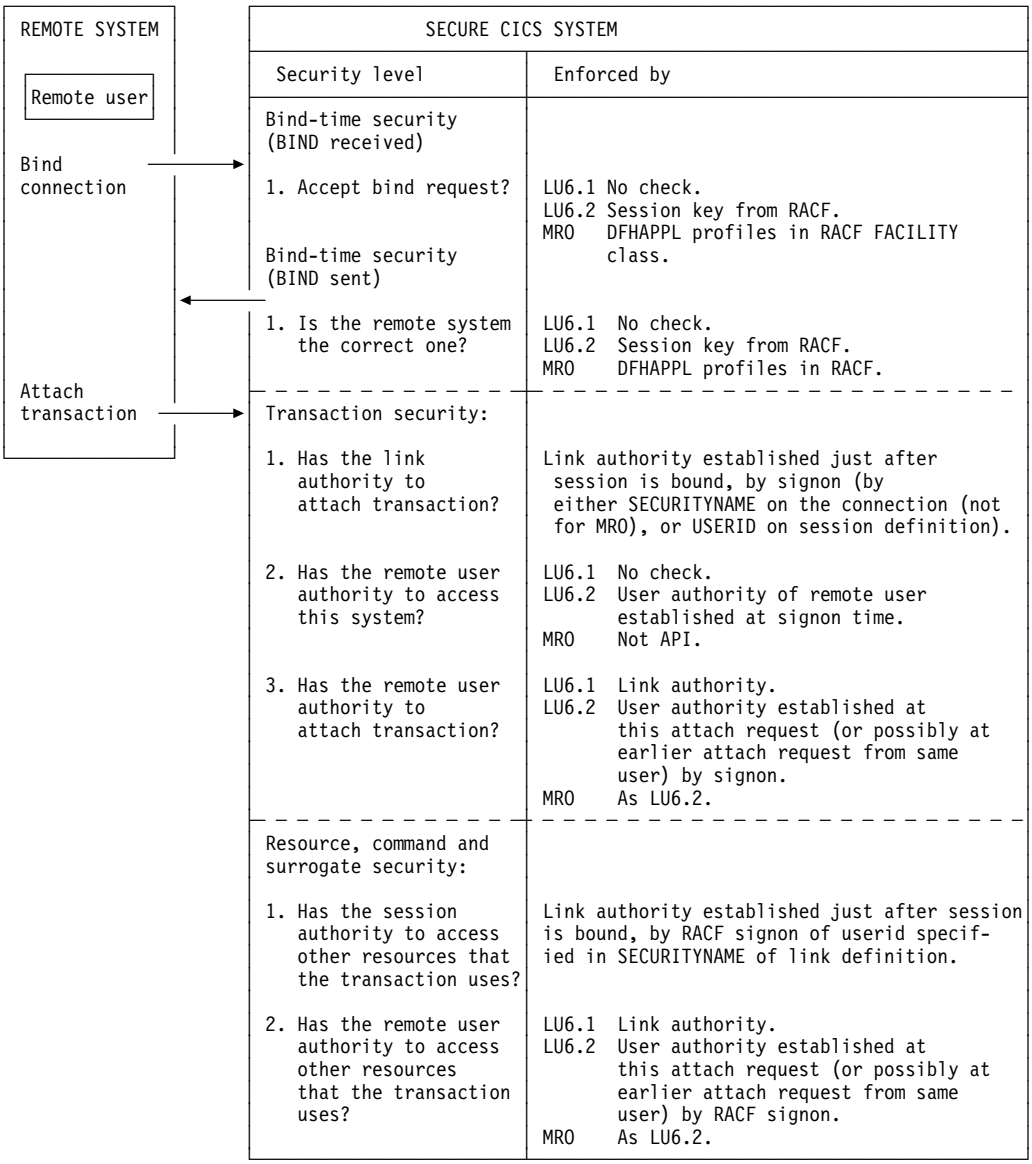

Figure 6. Summary of intersystem and interregion security

Chapter 12. Implementing LU6.2 security

This chapter tells you how to implement security for LU6.2. It is organized under the following topics:

Bind-time security with LU6.2

A security check can be applied when a request to establish an APPC session is received from, or sent to, a remote system, that is, when the session is bound. This is called **bind-time security** (or, in SNA terms, **session security**), and is part of the CICS implementation of the LU6.2 architecture. Its purpose is to prevent an unauthorized system from binding a session to one of your CICS systems.

Bind-time security is optional in the LU6.2 architecture. You must not, of course, specify bind-time security if the remote system does not support it. SNA defines how session security is to be applied, and CICS conforms to this architecture. If
you want to connect to a system other than CICS/ESA, or CICS/MVS, you must
make sure the other system is also compatible with this architecture.

When you define an LU6.2 connection to a remote system, you assume that all inbound bind requests originate in that remote system, and that all outbound bind requests are routed to the same system. However, where there is a possibility that a transmission line might be switched or broken into, you must guard against unauthorized session binds by specifying session security at both ends of the connection.

| For a bind request to succeed, both ends must hold the same **session key**, which
| is defined to RACF. When a session is bound, the action CICS takes depends on:

- How you specified the SEC and XAPPC system initialization parameters
| - How you specified the BINDSECURITY option on the CONNECTION resource
| definition in the CSD
| - Whether you have defined an APPCLU security profile for the link.

If you have SEC=YES and XAPPC=YES in your SIT, and BINDSECURITY(YES) in your CSD connection definition, and BINDSECURITY(YES) is also specified for the partner, system, a bind security validation will be attempted.

If you have BINDSECURITY(NO), then the SIT specification is immaterial.

Table 21 summarizes what happens.

Table 21. APPC bind-time security — relationship to resource definition

SEC value	XAPPC value	BINDSECURITY value	RACF APPCLU profile	Resulting CICS action
YES	YES	YES	Defined (See note 1)	CICS extracts the APPCLU profile from RACF at bind-time to verify the remote system
YES	YES	YES	Not defined	CICS is unable to extract the APPCLU profile from RACF and therefore rejects the bind
Any value	Any value	NO	Any value	CICS does not perform any bind validation. Bind not rejected for security reasons.
YES	NO	YES	Any value	CICS is unable to validate the bind, which is rejected
NO	Any value	YES	Any value	CICS is unable to validate the bind, which is rejected
NO	Any value	YES	Any value	CICS is unable to validate the bind, which is rejected

Notes:

1. If the RACF APPCLU profile is defined, but the session segment is locked or expired, or no value is specified for SESSKEY, the bind request is always rejected.
2. The table shows the response when the partner has specified BINDSECURITY(YES).

Example of defining an APPCLU profile

You can define an APPCLU profile as follows:

```
RDEFINE  APPCLU  netid.luid1.luid2 UACC(NONE)
         SESSION(SESSKEY(session-key)) AUDIT(ALL(READ))
         NOTIFY(CICS RACF Administrator)
```

In this example:

netid is the network ID, as specified on the NETID parameter in the VTAM startup member (ATCSTRxx) of SYS1.VTAMLST.

luid1 is the APPLID of the system on which the CONNECTION definition is installed.

luid2 is the NETNAME, as specified in the CONNECTION definition.

SESSKEY is the 16-hexadecimal-digit or 8-character password that matches the session key of the remote system. Hexadecimal digits must be enclosed in quotes, for example, `SESSKEY(X'0123456789ABCDEF')`.

The AUDIT and NOTIFY keywords are discussed in “Auditing bind-time security” on page 143.

You can also use the following options on the SESSION keyword. if required:

- INTERVAL, which you can use to specify the number of days for which the SESSKEY is to remain valid.
- LOCK, which can be used to stop the link being acquired by new sessions.

Defining bind-time security

You define bind-time security in the CONNECTION definition, although you must also choose the appropriate system initialization parameters. Figure 7 shows how to define APPC external session security, for which you need to specify the BINDSECURITY option.

```
CEDA DEFINE CONNECTION(name)
  GROUP(groupname)
  ACCESSMETHOD(VTAM)
  SECURITYNAME(name)
  PROTOCOL(APPC)
  NETNAME(name)
  BINDSECURITY(YES)
```

Note: For APPC terminals defined as a TERMINAL-TYPETERM pair, the BINDSECURITY operand is on the TERMINAL definition.

Figure 7. Bind-time security

Auditing bind-time security

If security is active (SEC=YES specified in the system initialization parameters), CICS performs bind security auditing. The following conditions are considered bind failures, and cause RACF to write an SMF record, and to issue a message.

- Session key does not match partner's.
- Session segment is locked.
- Session segment has expired.
- Session key is null.
- Session segment does not exist.
- Session segment retrieval was unsuccessful.
- Session bind was unsuccessful.

The following conditions are considered bind successes, and cause RACF to write an SMF record, but *not* to issue a message:

- Session was successfully bound.
- Session key will expire in less than 6 days.

An SMF record is written if either of the following is true:

- The profile's audit option is set (AUDIT(ALL(READ))).
- SETROPTS LOGOPTIONS(ALWAYS(APPCLU)) is set.

Two things happen when an SMF audit record is written:

- The userid specified in the profile's notify option is sent message ICH70005I. It is suggested that you specify the TSO userid of a RACF administrator who is responsible for the APPCLU class.
- The security console (any MVS console with a routing code of 9) receives message ICH415I, which contains text similar to message ICH70005I.

These audit records can be extracted from SMF and listed using the following sample RACF Report Writer control statements:

```
//RACFRW EXEC PGM=IKJEFT01
//SORTWKxx DD your sort files
//SYSPRINT DD SYSOUT=*
//SYSTPRT DD SYSOUT=*
//RSMFIN DD DSN=your smf dumped data, DISP=SHR
//SYSTIN DD *

RACFRW TITLE('Bind Security Reports') GENSUM
SELECT PROCESS
EVENT APPCLU
LIST SORT(DATE,TIME)
END

//
```

The RACF Report Writer is described in the *RACF Auditor's Guide*.

Changing RACF profiles—a warning

Take care when changing RACF profiles for APPC connections that are in use. If you are using RACF 1.9 the change in the RACF APPCLU profile is recognized by CICS only after a PERFORM SECURITY REBUILD command is issued. If you are using RACF 2.1, the change in the profile is recognized by CICS after a SETROPTS RACLIST(APPCLU) REFRESH command is issued. Bind-time security processing occurs when each session in a connection is acquired. If all of the sessions in a connection are not acquired and the APPC profile becomes invalid, then attempts to establish any of the unacquired sessions cause a bind security failure. This can cause transactions that attempt to allocate one of these unused sessions to be suspended indefinitely.

Reasons for invalid profiles

An APPC profile can become invalid for a number of reasons, for example:

- Expiration of the session key
- A change of the session key and a CICS security rebuild in one system without the corresponding change and rebuild in the other system
- The profile being locked while CICS security rebuild takes place.

Sessions that are already acquired still continue to function normally if there is bind security failure of another session. If you are using expiring session keys, then the connection can still be used after the expiry date, if any of the sessions on the connection were acquired (and have remained acquired) before the date of expiration. Hence, you see the effect of an expiring session key only when the connection (or session) is acquired.

Note: No warning messages are produced stating that the session key is about to expire. However, an SMF record can be written when a key that is due shortly to expire is used. Therefore, you can use the RACF Report Writer regularly to find out which keys need maintenance. Otherwise, if expiring session keys are used, you must remember when the keys are due to expire. You must also take appropriate action to minimize any disruption that may occur due to the connection being unavailable due to an expired session key. For example, you should plan for the changing of the session keys, security rebuilds (for both CICS systems) and for the possibility of having to reacquire the connection.

The problem of APPC profiles becoming invalid while the connection is in use can be avoided by specifying AUTOCONNECT(YES) or AUTOCONNECT(ALL) on the SESSIONS definition. This causes all sessions to be established (acquired) when the connection is acquired.

Link security with LU6.2

Link security further restricts the resources a user can access, depending on the remote system from which they are accessed. The practical effect of link security is to prevent a remote user from attaching a transaction or accessing a resource for which the link userid has no authority.

Link security can be associated with a connection or a session, depending on whether you want to control the link security for each group of sessions separately:

- To define link security for a connection as a whole, specify the SECURITYNAME parameter in the CONNECTION definition.
- To define link security for individual groups of sessions within a connection, specify the USERID in the SESSIONS definition as a user identifier.

| Each link between systems is given an authority defined by a link userid. A link
| userid for LU6.2 is a userid defined on your sessions definition for this connection.
| If not defined, the link userid is the SECURITYNAME userid specified on the
connection definition. If there is no SECURITYNAME, the link userid is the default
| userid.

| You can never transaction route or function ship to CICS without having at least
| one security check, but the security checks are minimized if the two regions
| involved are ***equivalent systems***. This term means the same thing for LU6.1,
LU6.2 and MRO. If the link userid matches the local region userid, you have
| equivalent systems.

| If you do have equivalent systems, only one security check is made. This will
either be against the default user (for ATTACHSEC=LOCAL) or against the userid
| that is in the received FMH-5 attach request (ATTACHSEC=non-LOCAL).

| If you do not have equivalent systems for ATTACHSEC=LOCAL, resource checks
| are done only against the link userid. For ATTACHSEC=non-LOCAL there are
| always two resource checks. One is against the link userid and the second is
| against the userid received from the remote user in the attach request.

| If a failure occurs in establishing link security, the link is given security of the local
| region's default user. This would happen, for example, if the preset session userid
| had been revoked.

If a value is present on the USERID parameter of the SESSIONS definition, the value overrides any value specified on the SECURITYNAME parameter.

User security with LU6.2

User security causes a second check to be made against a user signed onto a terminal, in addition to the link security described in “Link security with LU6.2” on page 145. After reading the following descriptions, you should consider whether you want the extra level of security checking that user security provides.

You can specify the following levels of user security using the ATTACHSEC parameter of the CONNECTION definition:

- *LOCAL*, which you specify if you do ***not*** want to make a further check on users by requiring a user identifier or password to be sent. Choose LOCAL if you do not want user security because you consider that the authority of the link is sufficient for your system. See “Specifying user security in link definitions” on page 148 for information on doing so.
- *Non-LOCAL*, which you specify if you *do* want to make a further check on users by requiring a user identifier, or a user identifier and a password to be sent. Non-LOCAL includes the following types of checking:
  - IDENTIFY

    a user identifier must be sent, but no password is requested
  - VERIFY

    a password must also always be sent
  - PERSISTENT VERIFICATION

    password is sent on the first attach request for a user
  - MIXIDPE

    either identify or persistent verification

Note: “Non-LOCAL user security verification” describes these types of user checking further. See “Specifying user security in link definitions” on page 148 for information on specifying them.

Non-LOCAL user security verification

In a CICS-to-CICS system connection, where you have a terminal-owning region (TOR), an application-owning region (AOR), and a data-owning region (DOR), the terminal operator signs on to the TOR, attaches a transaction in the AOR, and accesses resources in the DOR. If all three systems implement non-local user security, then the same operator is registered as a user in each of them. The usual procedure is for the operator to sign on to the TOR with a password. CICS assumes that the password is valid for the entire systems complex, and that it does not need to be passed on to the AOR and the DOR for further verification. All that is needed is for the AOR and the DOR to IDENTIFY the user, who is then signed on without a password. Therefore, the password is not sent with the attach request to the AOR. This is considered to be more secure, because the password is not passed on a network.

Specify IDENTIFY when you know that CICS can “trust” the remote system to verify its users (by some sort of signon mechanism) before letting them use the link. Use IDENTIFY if you want user security for CICS-to-CICS communication (because CICS does not support password flows on CICS-to-CICS connections).

If the front end does not have a security manager — for example, if it is a programmable workstation (PWS) — it is often not possible to VERIFY the user by means of a user identifier and password before the attach request reaches the AOR. The AOR must then receive both user identifier and password from the front end so that it can verify the user itself by a sign-on with password.

Specify VERIFY if you have reasons for wanting your own system to verify the remote system’s users even if they have already been checked by the remote system itself, or if the remote system does not have a security manager and therefore cannot verify its own users.

If programmable workstations make repeated requests to attach transactions in the AOR, performance suffers because of many verifications. The LU6.2 architecture, which defines these security procedures, allows persistent verification to reduce the software overhead. Using this protocol, the first attach request contains a user identifier and a password, but once the user has signed on, only the user identifier is needed for all the attach requests that follow.

Specify PERSISTENT to reduce the verification overhead if remote users repeatedly send attach requests. However, the remote system must be able to cooperate in the management of persistent verification by keeping a list of users who are currently signed on.

Some remote APPC systems have mixed signon requirements that vary from conversation to conversation (for example, CPI communications conversations). In this case, CICS must accept incoming identify or persistent requests.

To decide which of these types of user verification to use, you need to know how far the remote system is capable of managing its own security and, if it cannot, how far it must depend on the CICS system you are defining.

- Do you need to know the user identifier? If not, use LOCAL.
- Can the remote system verify its own users? If so, use IDENTIFY. If not, can it send a user identifier and a password with the attach request? If so, use VERIFY for PWS-to-CICS communication.
- Does the remote system support persistent verification by keeping track of its user identifiers and passwords? If you are using PWS-to-CICS communication, you may want to specify PERSISTENT, or MIXIDPE if you are using both CICS-to-CICS and PWS-to-CICS.

You specify these levels of checking for each connection using the ATTACHSEC operand of the CONNECTION definition, as described in “Specifying user security in link definitions” on page 148.

Specifying user security in link definitions

The level of user security you require for a remote system is specified in the ATTACHSEC operand in the CONNECTION definition, as shown in Figure 8.

This section describes how CICS interprets the parameters of the ATTACHSEC operand of the CONNECTION definition. However, special rules apply for CICS transaction routing, as described in "Transaction routing security with LU6.2" on page 155. Figure 8 shows an example of defining ATTACHSEC using CEDA.

```
CEDA DEFINE CONNECTION(name)
  GROUP(groupname)
  .
  ATTACHSEC(LOCAL|IDENTIFY|VERIFY|PERSISTENT|MIXIDPE)
```

Note: For APPC terminals defined as a TERMINAL-TYPETERM pair, the ATTACHSEC operand is on the TERMINAL definition.

Figure 8. Defining signon level for user security

The ATTACHSEC operand specifies the signon requirements for incoming transaction attach requests. It has no effect on attach requests that are issued by your system to a remote system; these are dealt with in the remote system.

When an APPC session is bound, each side tells the other the level of attach security user verification that will be performed on its incoming requests. There is no negotiation on this.

Meanings of ATTACHSEC operand

The following are the possible operands of ATTACHSEC:

LOCAL
specifies that a user identifier is not to be supplied by the remote system, and if one is received, the attach fails. CICS makes the user security profile equivalent to the link security profile. You do not need to specify RACF profiles for the remote users. LOCAL is the default value.

IDENTIFY
specifies that a user identifier is expected on every attach request. All remote users of a system must be identified to RACF.

If an attach request with both a user identifier and a password is received on a link with ATTACHSEC(IDENTIFY), CICS does not reject the attach request. CICS handles the attach request as if the connection was defined with ATTACHSEC(VERIFY).

If a null (X'00') character user identifier or an unknown user identifier is received, CICS rejects the attach request and unbinds the session.

If the connection is to an earlier release of CICS/ESA than version 3.2.1, see "Attach-time security and the USEDFLTUSER option" on page 153.

VERIFY
If a userid and an invalid password, or a userid and no password is received for verification, the attach will be rejected. If no userid is received, CICS applies the security capabilities of the default user.

The rules that apply to the checking of the user identifier for ATTACHSEC(IDENTIFY) also apply for ATTACHSEC(VERIFY). If a valid user identifier is received but the password verification fails then CICS rejects the attach request.

All CICS systems except CICS OS/2 can verify the security attributes of their users with an external security manager. CICS OS/2 does not have an external security manager and so is regarded as an insecure system. If CICS OS/2 is the terminal-owning region (TOR) connected to CICS/ESA, the ATTACHSEC=VERIFY option should be used in the LU6.2 connection definition on the CICS/ESA application owning region (AOR). The appropriate adjustments should also be made to the communications manager on the CICS OS/2 system so that the password and userid of the user signing on to CICS OS/2 are sent. (See the *CICS OS/2 Intercommunication Guide*, SC33-0826) for details of the communications manager changes that need to be made.) CICS/ESA is then able to VERIFY the user by performing a signon with password.

Note: Products other than CICS can connect to a CICS/ESA AOR via an LU6.2 link. They then use the SNA LU6.2 FMH-5 ATTACH mechanism to start a transaction on the CICS AOR. Where this mechanism is being used from an insecure system, the ATTACHSEC=VERIFY option should be used on the connection definition to protect the transaction on the AOR. (See “SNA profiles and attach-time security” on page 152 for information about the SNA LU6.2 ATTACH FMH-5.) For more information about ATTACHSEC and USEDFLTUSER, see “Attach-time security and the USEDFLTUSER option” on page 153.

PERSISTENT
specifies that a user identifier and a user password are required with the first attach request for a new user, but all following attach requests for the same user need supply only a user identifier. (All remote users of a system must be identified to RACF.) The first attach signs on the user, even if the attach request is later unsuccessful because the user is not authorized to attach the transaction.

Note: PERSISTENT cannot be used for CICS-to-CICS communication.

MIXIDPE
specifies that the sign-on level for the remote user is determined by parameters sent with the attach request. The possibilities are: PERSISTENT or IDENTIFY.

Signon status

With the ATTACHSEC parameters IDENTIFY, MIXIDPE, PERSISTENT, and VERIFY, the remote user remains signed on after the conversation associated with the first attach request is complete. CICS then accepts attach requests from the same user without a new signon until either of the following occurs:

- The period specified in the system initialization parameter USRDELAY elapses after completion of the last transaction associated with the attach request for this user.

  When you are running remote transactions, over ISC and IRC links, USRDELAY defines the time for which entries can remain signed onto the remote CICS region. For information on specifying USRDELAY, see the *CICS/ESA System Definition Guide*. See the *CICS/ESA Performance Guide* for information on tuning.

- The CICS system is terminated.

If you alter the RACF profile of a signed-on remote user (for example, by revoking the user), CICS continues to use the authorization established at the first attach request until the entry is removed from the signon list.

Password checking

If you are using ATTACHSEC(PERSISTENT) (or ATTACHSEC(MIXIDPE) being treated as ATTACHSEC(PERSISTENT)), CICS maintains a table for each remote system called the ***persistent verification (PV) signed-on-from list***. This is a list of users whose passwords have been checked and who do not require a further password check as long as the entry remains in the list. Entries remain in the list until:

- The period specified in the system initialization parameter PVDELAY elapses after the user's signon entry was last used.

  PVDELAY defines how long entries can remain in the PV signed-on-from list for the remote system, which means that their passwords do not need to be rechecked for every attach request. For information on specifying a value for PVDELAY, see the *CICS/ESA System Definition Guide*. See the *CICS/ESA Performance Guide* for information on tuning.

- The connection with this system is terminated because: CICS is restarted, the connection is lost, or CICS receives an invalid attach request from the user.

When persistent verification is in operation for a remote user, and that user is removed from the PV signed-on-from list, CICS informs the remote system by issuing a signoff request for the user to remove the entry from the PV signed-on-to list in the remote system.

If you specify ATTACHSEC(VERIFY), the remote user's password is checked for *every attach request*. This is to ensure that the user has authority to access this system, to verify that this password is correct, and to establish security authorities for the user.

Information about remote users

Information about the user can be transmitted with the attach request from the remote system. This means that you can protect your resources not only on the basis of which remote system is making the request, but also on the basis of which actual user at the remote system is making the request.

This section describes some of the concepts associated with remote-user security, and how CICS sends and receives user information.

You have to define your users to RACF. If a remote user is not defined to RACF, any attach requests from that remote user are rejected.

CICS remote-user security for LU6.2 links implements the LU6.2 architecture. The LU6.2 architecture allows user identifiers, user passwords, and user profiles to be transmitted with requests to attach a transaction.

| User profiles can be transmitted instead of, or in addition to, user identifiers. The
| profile name, if supplied, is treated as the groupid.

APAR PQ15449

MJO 3/6/98

If the user has been added to the front-end system with a groupid explicitly specified, (for example, in EXEC CICS SIGNON or by filling in the GROUPID parameter on the CESN panel) this will be propagated by CICS in outbound attach FMHs for LU6.2 links when ATTACHSEC(IDENTIFY) has been specified in the CONNECTION definition.

If the groupid defaults when the user was originally added to the front-end system, the profile field will not be included in the outbound FMH5. If the groupid is passed to the back-end system, the groupid will be used as part of ADD_USER proceeding on the back-end. The userid must be defined as a member of the group passed in the ESM on the back-end for the ADD_USER to be successful.

It is advisable to use the PLTPIUSR system initialization parameter if there is a possibility that a task started by PLTPI processing will access remote resources. This avoids problems in the remote region where the user ID is not in the same group as the user in the local region. This is because the PLTPI user in the local region is not added with an explicit groupid, and as a result the groupid is not propagated to the remote region.

CICS sends userids on ATTACHSEC(IDENTIFY) conversations. Table 22 on page 152 shows how CICS decides which userid to send.

Table 22. Attach-time user identifiers — LU6.2

Characteristics of the local task	**User identifier sent by CICS to the remote system**
Task with associated terminal — user identifier	Terminal user identifier
Task with associated terminal — no user signed on and no USERID specified in the terminal definition	Default user identifier for the local system
Task with no associated terminal or USERID started by interval control START command (if using function shipping or distributed transaction processing (DTP))	User identifier for the task that issued the START command
Task started with USERID option	User identifier specified on the START command
CICS internal system task	CICS region userid
Task with no associated terminal started by transient data trigger	User identifier specified on the DCT that defines the queue
Task with associated terminal started by transient data trigger	Terminal user identifier
Task started from PLTPI	PLTPIUSR

Signing on the remote user has two purposes:

- To ensure that the remote user is allowed to access the CICS system
- If the sign-on is successful, to establish the authority for the remote user.

CICS signs off the remote user under the circumstances described in “Signon status” on page 149.

SNA profiles and attach-time security

CICS implementation of the LU6.2 attach-time security in CICS 4.1 conforms strictly to the architecture. In particular the following should be noted:

- The introduction of SNA profile support and the conformance to SNA attach-time security processing may cause migration problems.
- Profile support means that badly coded profiles sent in an attach FMH-5 cause certain attach requests to be rejected.
- The checks to prevent problems in the access security subfields of an FMH-5 are:
  - Check for unrecognized subfield
  - Check for invalid length subfield
  - Check for multiple subfields of the same type.
- The full 10–character userid and password are accepted.

  APAR PN80103

  MJO. 13 May 1996

  Any trailing blanks (X'40') are removed before passing to the security manager, which either rejects the attach request, or converts the userid and password into 8–character form before proceeding.
- Attach requests are rejected if they do not contain security parameters in an FMH-5 unless the USEDFLTUSER parameter has been coded on the connection.
- Attach requests are rejected if they have a blank, or zero-length userid parameter in the attach FMH-5. See “Attach-time security and the USEDFLTUSER option” on page 153 for a description of the exception where zero-length userids may be accepted for ATTACHSEC-VERIFY.
- Valid SNA profiles received are treated as the ESM groupid with which the userid in the FMH-5 will be associated after the userid in the FMH-5 is signed on.
- When a SNA profile is received and the connection had ATTACHSEC=PERSISTENT, it is validated to conform to the architecture. It is not used to further qualify users in the signed-on-from list. This also applies to persistent signed-on flows received on a connection that has ATTACHSEC=MIXIDPE specified.

Attach-time security and the USEDFLTUSER option

In earlier releases of CICS/ESA a user who was not signed on would not have an associated userid. In CICS/ESA 4.1, coding USEDFLTUSER on the connection indicates that the default user can be used. The following types of incoming attach FMH 5 are only accepted by CICS/ESA 4.1 if the USEDFLTUSER option is coded on the connection:

- An FMH-5 with an ATTACHSEC of IDENTITY not containing a userid subfield, for example, from a CICS/XA or a CICS/VSE system.
- An FMH-5 with an ATTACHSEC of VERIFY containing userid and password subfields which have zero-length, for example, from certain non-EBCDIC based systems).

  APAR PN78900

  MJO 26/2/96

- An FMH-5 with an ATTACHSEC of VERIFY containing an access security
information field (ASIF) length field of zero.

If the user does not specify the USEDFLTUSER option in these exceptions, the
expected protocol violation occurs, a message is generated, and the attach fails.

Transaction, resource, and command security with LU6.2

As in a single-system environment, users must be authorized to:

- Attach a transaction
- Access all the resources that the transaction is programmed to use. These levels are called *transaction security*, *resource security*, *surrogate user security*, and *command security*.

Transaction security

As in a single-system environment, the security requirements of a transaction are specified when the transaction is defined, as described in Chapter 5, “Transaction security” on page 77.

In an LU6.2 environment, two basic security requirements must be met before a transaction can be initiated:

- The link must have sufficient authority to initiate the transaction.
- If anything other than ATTACHSEC(LOCAL) has been specified, user security is in force. The “user” who is making the request must therefore have sufficient authority to access the system and to initiate the transaction.

Note: Transaction security also applies to the mirror transactions. See “Function shipping security with LU6.2” on page 156.

Resource and command security

Resource and command security in an intercommunication environment are handled in much the same way as in a single-system environment.

Resource and command security checking are performed only if the installed TRANSACTION definition specifies that they are required; for example, on the CEDA DEFINE TRANSACTION command, as shown in Figure 9 on page 154.

```
CEDA DEFINE TRANSACTION
  .
  RESSEC(YES)
  CMDSEC(YES)
  .
```

Figure 9. Specifying resource and command security for transactions

If a TRANSACTION definition specifies resource security checking, using RESSEC(YES), both the link and the user must also have sufficient authority for the resources that the attached transaction accesses.

If a TRANSACTION definition specifies command security checking, using CMDSEC(YES), both the link and the user must also have sufficient authority for the SP commands shown in Table 11 on page 109 that the attached transaction issues.

For further guidance on specifying resource and command security, see Chapter 6, "Resource security" on page 83 and Chapter 8, "CICS command security" on page 109.

NOTAUTH exceptional condition

If a transaction tries to access a resource, but fails the resource security checks, the NOTAUTH condition occurs.

When the transaction is the CICS mirror transaction, the NOTAUTH condition is returned to the requesting transaction, where it can be handled in the usual way.

Transaction routing security with LU6.2

In transaction routing, the authority of a user to access a transaction can be tested in both the TOR and the AOR.

In the TOR, a test is made to ensure that the user has authority to access the transaction defined as remote, just as if it were a local transaction. This test determines whether the user is allowed to run the relay program.

The terminal through which the transaction is invoked must be defined on the remote system (or defined as "shippable" in the local system), and the terminal operator needs RACF authority if the remote system is protected. The way in which the terminal on the remote system is defined affects the way in which user security is applied:

- If the definition of the remote terminal does not specify the USERID parameter:
  - For links defined with ATTACHSEC(IDENTIFY), the transaction security and resource security of the user are established when the remote user is signed on. The userid under which the user is signed on, whether explicitly or implicitly (in the DFLTUSER system initialization parameter) has this security capability assigned in the remote system.

– For links defined with ATTACHSEC(LOCAL), transaction security, command security, and resource security are limited by the authority of the link.

In both cases, tests against the link security are made as described in “Link security with LU6.2” on page 145.

Note: During transaction routing, the 3-character operator identifier from the TOR is transferred to the surrogate terminal entry in the AOR. If the surrogate terminal was shipped in, this identifier is not used for security purposes, but it may be referred to in messages.

When transaction routing PSB requests, the following conditions must both be satisfied:

- ATTACHSEC on the connection definition must not be LOCAL (that is, it can be IDENTIFY, PERSISTENT, MIXIDPE, or VERIFY)
- PSBCHK=YES must be specified as a system initialization parameter in the remote system.

Preset security terminals and transaction routing

A terminal has preset security if a value is specified on the USERID parameter of the terminal definition. When considering the security aspects of transaction routing from a preset security terminal, you must remember that preset security is an attribute of the terminal rather than of the user who initiated the transaction routing request.

During transaction routing, a surrogate terminal is created in the AOR to represent the terminal at which the transaction routing request was issued. Whether the surrogate terminal has preset security or not depends upon a number of factors:

- If a remote terminal definition exists in the AOR for the terminal at the TOR, and specifies the USERID parameter, the surrogate terminal is preset with this userid. If the USERID parameter is not coded, the surrogate terminal does not have preset security.
- If a remote terminal definition does not exist in the AOR, the preset security characteristics of the surrogate terminal are determined from the terminal definition shipped from the TOR. If the shipped terminal definition has preset security, the surrogate also has preset security, unless the connection to the AOR is defined with ATTACHSEC=LOCAL, in which case any preset security information shipped to the AOR is ignored.

CICS routing transaction, CRTE

You can use the CICS routing transaction, CRTE, with LU6.2 to run transactions that reside on a connected remote system, instead of defining these transactions as remote in the local system. CRTE is particularly useful for infrequently used transactions, or for transactions such as CEMT that reside on all systems.

Security checking done in the AOR for CRTE does not depend on what is specified by ATTACHSEC. Instead, security checking depends on whether the user signs on while using CRTE:

- If the user does **not** sign on, the surrogate terminal created is associated with the AOR default user. When a transaction is run, the security checks are

carried out against this default user. A check is also done against the link userid to see whether the routing application itself has authority to access the resource.

- When a user **does** sign on to the AOR, using the CESN transaction while running CRTE, the surrogate already created then points to the userid of the signed-on user. For transactions attempting to access resources, security checking is done against the signed-on user's userid in the surrogate and the link userid.

For more information on CRTE, see the *CICS/ESA CICS-Supplied Transactions* manual and the *CICS/ESA Intercommunication Guide*.

Function shipping security with LU6.2

When CICS receives a function-shipped request, the transaction that is invoked is the **mirror transaction**. The CICS-supplied definitions of the mirror transactions all specify resource, but not command, security checking. This means that you are prevented from accessing the remote resources if either the link or your userid profile on the other system do not have the necessary authority.

If the CICS-supplied definitions of the mirror transactions are not what your security strategy needs, you can change them by copying the definitions in group DFHISC into your own group, changing them and then reinstalling them. For more information, see "Category 2 transactions" on page 127.

If you include a remote resource in your resource definitions, you can arrange for security checking to be done locally, just as if the resource were a local one. Also, the system that owns the resource can be made to apply an independent check, if it is able to receive the user identifier. You can therefore choose to apply security restrictions on both sides, on either side, or not at all.

Note: Be aware that if you specify the SYSID option on a function-shipped request, security checking is done in the remote system but is *bypassed in the local system*. Figure 10 on page 157 summarizes what happens.

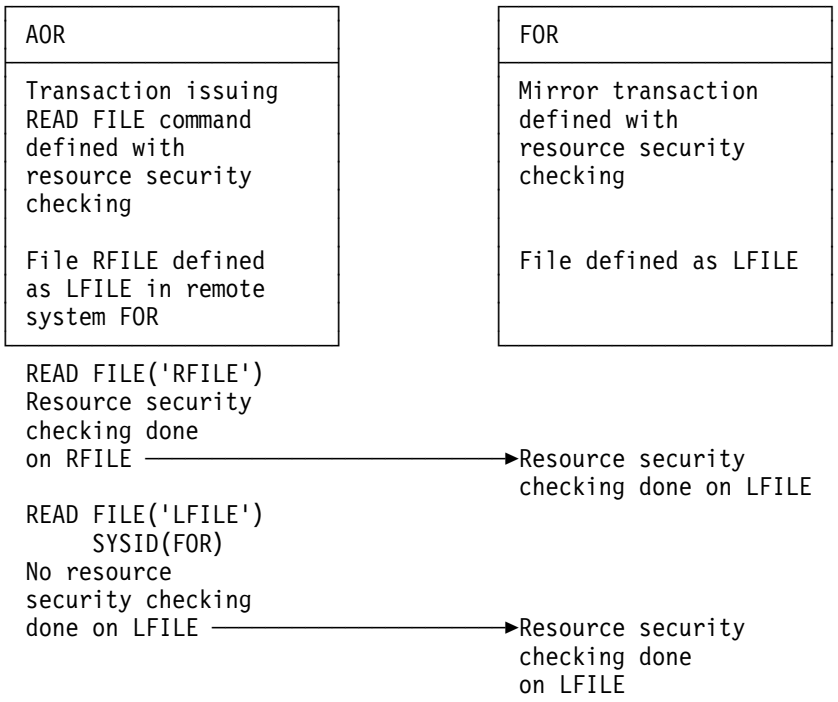

Figure 10. Security checking done with and without SYSID

For programming information on specifying the SYSID option, see the *CICS/ESA Application Programming Reference* manual.

Distributed program link security with LU6.2

The CICS distributed program link (DPL) facility enables a CICS program (the client program) to call another CICS program (the server program) in a remote CICS region. DPL is used when the SYSID option on the EXEC CICS LINK PROGRAM command, or the REMOTESYSTEM option of the program resource definition, specifies a remote CICS region.

When the SYSID option on the EXEC CICS LINK command specifies a remote CICS system, the client region does not perform any resource security checking, but leaves the resource check to be performed in the server region.

The server program in the remote region is executed by a mirror transaction, in a similar way to other function-shipped CICS requests. However, the transaction name associated with the mirror depends on how the EXEC CICS LINK PROGRAM is invoked in the client region. You must be aware of the transaction name because normal attach security applies to the mirror transaction:

- If the TRANSID option is specified on the DPL command, the specified transaction name is used for the mirror.
- If the TRANSID option is omitted from the DPL command, but the TRANSID option is used in the program resource definition in the client region, the name for the mirror is taken from the program's TRANSID specification.

Otherwise, a default name for the mirror transaction is used, and this depends on the origin and LU6.2 sync level of the conversation:

- If the client program is executing in a CICS OS/2 system, the transaction name for the mirror is **CPMI**.
- If synclevel 1 is being used, the default transaction name for the mirror is **CVMI**. This transaction name is used:
  - If the SYNCONRETURN option is specified on the DPL command in the client region
  - If the LU6.2 CONNECTION definition specifies SINGLESESS(YES)
  - If the connection is by means of an LU6.2 terminal; that is, a terminal whose resource definition uses a TYPETERM with a specification of DEVICE(APPC).
- If sync level 2 is being used, the default transaction name is **CSMI**. This transaction name is used when none of the other previous conditions is met.

You must authorize your users to access the transaction name that the mirror runs under. The userids to be authorized depend on whether LOCAL or non-LOCAL attach security is being used, and are described in "Security checking done in AOR with LU6.2." If the mirror transaction is defined with RESSEC(YES) in the server region, these userids must also be authorized to access the server program that is being linked to by the mirror. If the server program accesses any CICS resources, the same userids must be authorized to access them. If the server program invokes any SP-type commands, and the mirror transaction is defined with CMDSEC(YES) in the server region, the same userids must be authorized to access the commands.

If the mirror transaction cannot be attached because of security reasons, the NOTAUTH condition is not raised, but the TERMERR condition is returned to the issuing application in the client region. If the mirror transaction is successfully attached, but it is not authorized to link to the distributed program in the server region, the NOTAUTH condition is raised. The NOTAUTH condition is also raised if the server program fails to access any CICS resources for security reasons.

The server program is restricted to a DPL-subset of the CICS API commands when running in a server region. The commands that are not supported include some that return security-related information. For programming information about which commands are restricted, see the *CICS/ESA Application Programming Reference*. For further information about DPL, refer to the *CICS/ESA Intercommunication Guide*.

Security checking done in AOR with LU6.2

This section summarizes how security checking is done in the AOR depending on how SECURITYNAME is specified in the AOR and TOR.

The link userid referred to in Table 23 on page 159 and Table 24 on page 160 is the one specified in the SECURITYNAME on the CONNECTION definition, or the USERID on the SESSIONS definition.

If a USERID is specified on the SESSIONS definition, and a link check is done, the userid used is the one on the SESSIONS definition.

If no userid is specified in SECURITYNAME, then the default userid of the AOR is used instead. However, if the SECURITYNAME userid is the same as the region

userid for the AOR, then the link is deemed to have the same security as the AOR, and ***link security is omitted altogether***. The effect of omitted link security depends on whether LOCAL or non-LOCAL attach security is specified for the link:

- For LOCAL attach security, the security specified in the USERID on the SESSIONS definition is used. If this too is omitted, then the default userid for the AOR is used.
- For non-LOCAL attach security, the security specified in the USERID on the sessions definition is *not* used. Only the userid received from the TOR is used to determine security.

Note: Neither the region userid for the TOR, nor the SECURITYNAME in the TOR's CONNECTION definition for the AOR, are relevant to security checking in the AOR.

Table 23 shows how checking is done when ATTACHSEC(LOCAL) is specified.

Table 23. LU6.2 and ATTACHSEC(LOCAL)

Region userid for AOR	SECURITYNAME in connection definition	USERID in SESSION definition	Checking in AOR
USERIDA	Not specified	Not specified	Check against AOR DFLTUSER
USERIDA	Not specified	USERIDA	Check against AOR DFLUTSER
USERIDA	Not specified	USERIDB	Check against USERIDB
USERIDA	USERIDA	Not specified	Check against AOR DFLTUSER
USERIDA	USERIDB	Not specified	Check against USERIDB
USERIDA	USERIDA	USERIDA	Check against AOR DFLTUSER
USERIDA	USERIDA	USERIDB	Check against USERIDB
USERIDA	USERIDB	USERIDA	Check against DFLTUSER
USERIDA	USERIDB	USERIDB	Check against USERIDB
USERIDA	USERIDB	USERIDC	Check against USERIDC

Table 24 on page 160 shows how checking is done when the ATTACHSEC parameter IDENTIFY (or PERSISTENT, or MIXIDPE) has been specified.

Table 24. LU6.2 and ATTACHSEC(IDENTIFY|PERSISTENT|MIXIDPE)			
Region userid for AOR	**SECURITYNAME in connection definition**	**USERID in SESSION definition**	**Checking in AOR**
USERIDA	Not specified	Not specified	Transmitted userid and AOR DFLTUSER
USERIDA	Not specified	USERIDA	Transmitted userid only
USERIDA	Not specified	USERIDB	Transmitted userid and USERIDB
USERIDA	USERIDA	Not specified	Transmitted userid only
USERIDA	USERIDA	USERIDA	Transmitted userid only
USERIDA	USERIDA	USERIDB	Transmitted userid and USERIDB
USERIDA	USERIDB	Not specified	Transmitted userid and USERIDB
USERIDA	USERIDB	USERIDC	Transmitted userid and USERIDC

Summary of resource definition options for LU6.2 security

The following is a summary of the resource definition options you need to define for LU6.2 security:

- On the CONNECTION definition:
  - ATTACHSEC, with any one of the following options:

    IDENTIFY
    LOCAL
    MIXIDPE
    PERSISTENT
    VERIFY

  - BINDPASSWORD
  - BINDSECURITY
  - SECURITYNAME
- On the SESSIONS definition:
  - USERID

For guidance on specifying CONNECTION and SESSION definitions, see the *CICS/ESA Resource Definition Guide*.

Chapter 13. APPC password expiration management

This chapter contains information on advanced program-to-program communications (APPC) password expiration management (PEM).

To use PEM you should understand APPC conversation-level security. To code the requester signon transaction, you also need to have basic APPC programming skills.

To find out what APPC PEM offers, read "Introduction to APPC password expiration management." System programmers responsible for coding the PEM requester should also read "APPC PEM processing" on page 164, which explains the requirements of the PEM requester and CICS PEM server.

Introduction to APPC password expiration management

This chapter introduces, and describes the benefits of, APPC password expiration management.

What APPC PEM does

APAR PQ07559

13/10/97. MJO

APPC PEM with CICS provides receive support for an APPC architected signon
transaction that verifies userid, password pairs, and processes requests for a password change by:

- Identifying a user and authenticating that user's identification
- Notifying specific users during the authentication process that their passwords have expired
- Letting users change their passwords when (or before) the passwords expire
- Telling users how long their current passwords will remain valid
- Providing information about unauthorized attempts to access the system using a particular user identifier.

Benefits of APPC PEM

APPC PEM has the following benefits:

- It enables users to update passwords on APPC links.

  This can be particularly useful in the case of expired passwords. On APPC links that do *not* support APPC PEM, when users' passwords expire on remote systems, they are unable to update them from their own systems. The only alternative on a non-APPC PEM system is to log on directly to the remote system via a non-APPC link, such as an LU2 3270-emulation session, to update the password.

- It provides APPC users with additional information regarding their signon status; for example, the date and time at which they last signed-on.

- It informs users whether their userid is revoked, or the password has expired, when they provide the correct password or PassTicket.

What is required to use APPC PEM

To use APPC PEM, you need a **PEM requester** and a **PEM server** linked by an APPC session. An external security manager, such as RACF, or an equivalent ESM, must also be available to the PEM server. Figure 11 shows an example configuration.

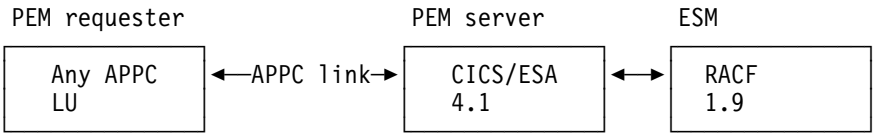

Figure 11. Example APPC PEM configuration

The PEM requester can be any APPC logical unit (LU) or node that is capable of initiating a conversation with the architected signon transaction. However, the benefits of using APPC PEM are increased when using an LU or node that does not have its own ESM; for example, a programmable workstation, such as an IBM Personal System/2 (PS/2). APPC PEM enables users of such LUs or nodes to manage their password values within the ESM used by CICS.

The PEM server can be any APPC LU that supports APPC PEM. In the context of this book, we assume that the PEM server is the one provided by CICS/ESA 4.1. It is referred to in the rest of this book as the CICS PEM server.

Roles of PEM requester and CICS PEM server

CICS/ESA does not send passwords on APPC conversations. This means that it can *attach*, but not *initiate* the signon transaction, and must always act as the PEM server. Therefore, the configuration must always include an LU that is capable of initiating the signon transaction to act as the PEM requester.

The PEM requester collects signon information and sends it to the CICS PEM server via a signon transaction program. The signon transaction program is a SNA service transaction program, as described in *SNA LU 6.2 Peer Protocols* manual.

APAR PQ07559

13/10/97 MJO

Note that a PEM signon is not to be confused with a CICS signon. In CICS, PEM signon is a way for the APPC LU to verify and manage passwords. Following verification or updating, the userid or password is intended to be included as the ASIS part of the FMH5 attach header. When this FMH5 is sent into CICS through the APPC link (provided ATTACHSEC is non-local) the userid is signed on to CICS. Therefore, a PEM signon does not result in the ESM last-connect, last-access information being updated. For more information, see “APPC password expiry management” on page 233.

The CICS PEM server then processes the signon request, updates the user’s password (if necessary), and returns a reply to the PEM requester containing responses and other data, such as a password expiry and information about

unauthorized attempts to sign on. The PEM requester then processes the data, as appropriate.

Example APPC PEM signon

Figure 12 shows an example signon for APPC PEM.

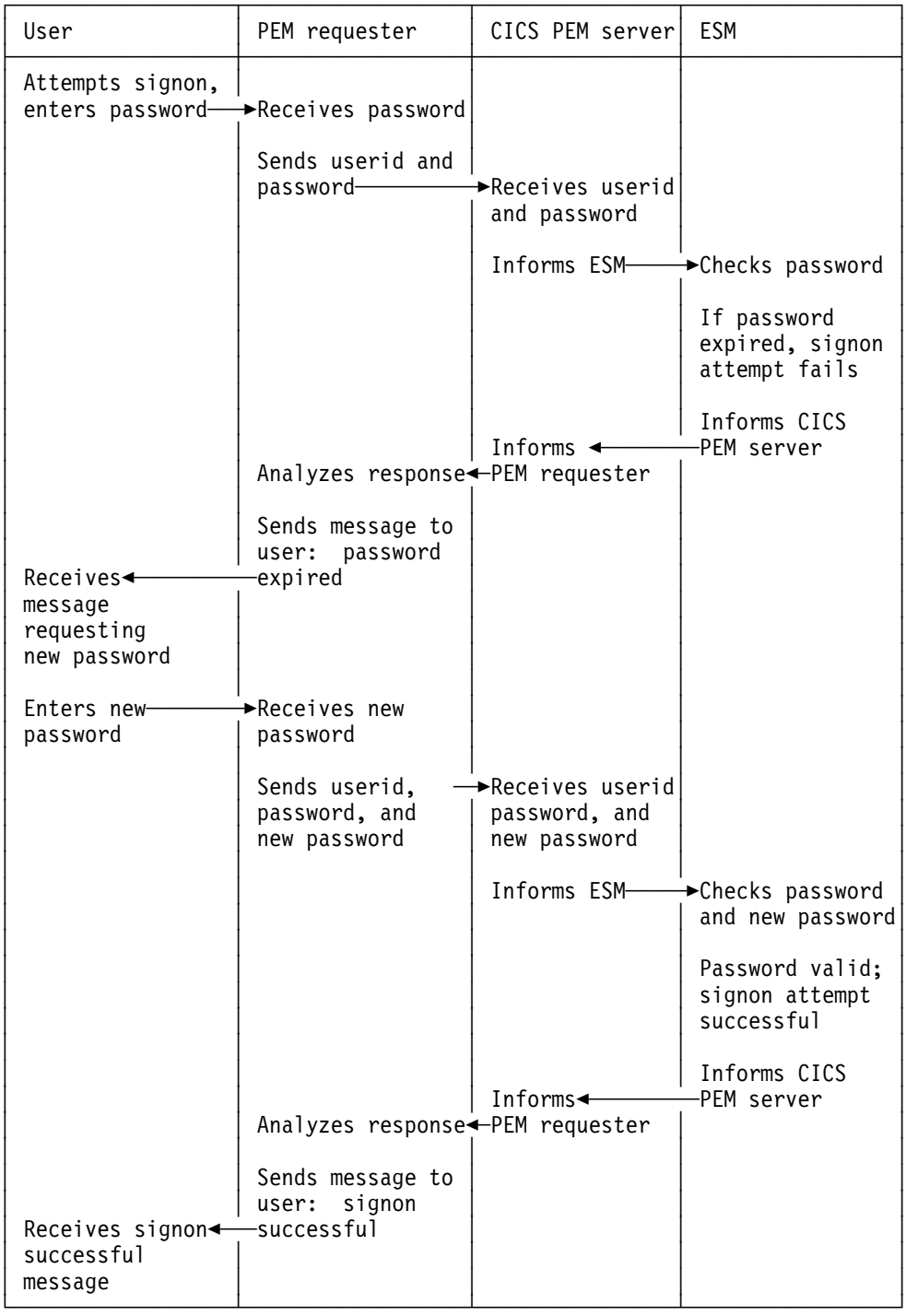

Figure 12. Example of signon with APPC PEM

APPC PEM processing

In order to code the signon transaction program for the PEM requester to send the signon details to the CICS PEM server, you need to know the following:

- What happens on each side of the link — see "Overview of APPC PEM processing."
- How to code the PEM requester — see "Setting up the PEM requester" on page 169, "Format of user data" on page 170, and "Examples of PEM requester and CICS PEM server user data" on page 178.

  See also Appendix A, "APPC PEM requester example program" on page 265, which contains an example of a PEM requester signon transaction program.
- The data the CICS PEM server requires from the PEM requester — see "Signon input data sent by PEM requester" on page 173
- The data the CICS PEM server sends in response to the PEM requester — see "Signon output data returned by CICS PEM server" on page 174.

Overview of APPC PEM processing

CICS provides the PEM server, the receive side of APPC PEM, by means of a CICS transaction that is started when an ATTACH for the signon transaction program is received from the PEM requester.

CICS retrieves the signon data associated with the request, calls the ESM to perform a signon, and retrieves signon details for the userid. If the signon data includes a new password value, CICS includes this value when it calls the ESM to request a signon.

If PV is being used, and signon completes correctly, the user is added to the PV "signed-on-*from* list" in CICS, and the PV "signed-on-*to* list" in the PEM requester.

APAR PQ07559

12/10/97 MJO

The "signed-on-" lists keep track of verified userids.

The CICS PEM server builds a reply and returns it to the PEM requester, after which the CICS PEM server transaction terminates normally.

PEM requester processing

The PEM requester signon transaction program:

1. Obtains signon information, for example by:
   - Displaying a message to the user requesting signon information; that is, userid, password, and, if required, new password; or
   - Accessing sign on information from a user who has already been authenticated locally.
2. Sends the signon information to the CICS PEM server via an APPC conversation.
3. Receives replies from the CICS PEM server on the same APPC conversation.

4. If PV is being used (either ATTACHSEC=PERSISTENT or ATTACHSEC=MIXIDPE is specified on the CONNECTION definition), and signon is successful, it adds the user's name to the PV signed-on-to list.
5. Processes the reply information from the CICS PEM server; for example, by:
   - Displaying the information to the user
   - Processing the data and saving it in a file to which only the user has access.

CICS PEM server processing

The CICS PEM server performs the following processing:

1. Accepts the userid and password, with optional new password, from the signon PEM requester.

   APAR PQ07559

   13/10/97 MJO

2. Tries to validate the user with its ESM.

   If the userid and password are valid and the password has not expired, the CICS PEM server extracts the following information from its ESM:
   - Date and time of the last successful signon
   - Data and time the password will expire (calculated by data extracted from the ESM by the CICS PEM server itself)
   - Number of unsuccessful signon attempts since the last successful signon.
3. Returns a response to the PEM requester (described in Table 26 on page 174, and illustrated in both Figure 18 on page 178 and Figure 21 on page 182), indicating whether the signon was successful or failed, and the reason for any failure:

```
Status           = (X'00') OK
Date-Time        = Current date and time
Last-Date-Time   = Date and time of previous successful signon
Expiry-Date-Time = Date and time password will expire
Revoke-Count     = Number of unsuccessful signon attempts made with
                   this userid since the previous successful signon
```

   Note: The revoke count is incremented by the ESM whenever an invalid signon attempt is processed. The signon request may originate from a non-CICS system (for example, a TSO user).

   If signon is unsuccessful, CICS returns to the PEM requester: a signon completion status value (as described in Table 28 on page 176) and, if appropriate, a formatting error value (as described in Table 29 on page 177).
4. If PV is being used (either ATTACHSEC=PERSISTENT or ATTACHSEC=MIXIDPE is specified on the CONNECTION definition), and signon is successful, it adds the user's name to the PV signed-on-from list.

Expected flows between PEM requester and CICS PEM server

Figure 13 through Figure 16 on page 169 show expected flows for successful and unsuccessful signon attempts with and without PV. These examples do not include information on setting up the connection. For more information on doing this, see the *CICS/ESA Intercommunication Guide*.

Note: CICS support for the PEM requester signon transaction assumes that the request for signon (or signon and change password) is for a single user. Batching of signon requests for different userids within a single signon transaction is not supported. If multiple sign on requests are passed in the input data, the CICS PEM server processes only the first one.

Successful signon — non-PV connection: Figure 13 shows the expected flows between the PEM requester and CICS PEM server during a successful signon when PV is not being used.

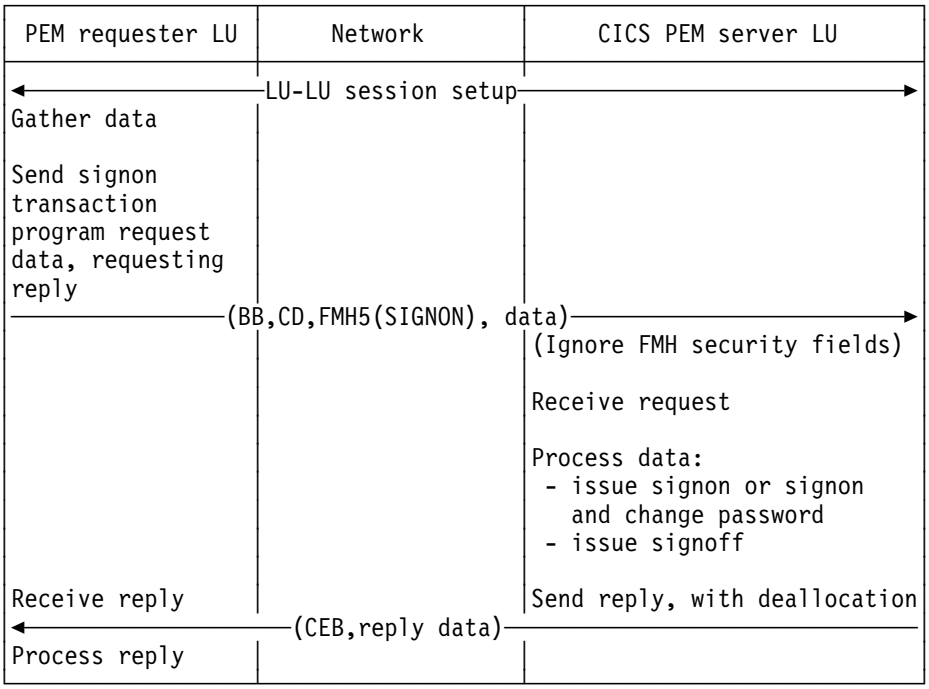

Figure 13. Successful signon — non-PV connection

Note: All security fields in the FMH-5 (userid, password and UP, AV, PV1 and PV2 bits) are ignored by the CICS PEM server when it attaches the signon transaction.

Unsuccessful signon — non-PV connection: Figure 14 shows the expected flows for an unsuccessful signon between a PEM requester and CICS PEM server when PV is not being used.

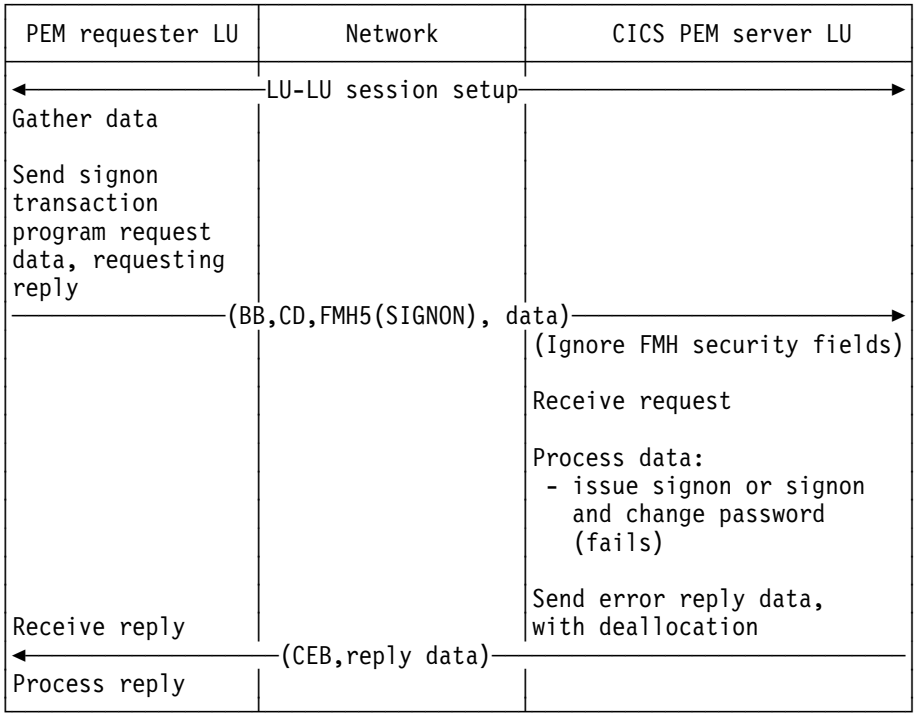

Figure 14. Unsuccessful signon — non-PV connection

Note: The CICS PEM server schedules signoff against the PEM requester if a sign on request for a userid fails.

Successful signon — PV connection: Figure 15 shows the expected flows between the PEM requester and CICS PEM server during a successful signon on a PV connection.

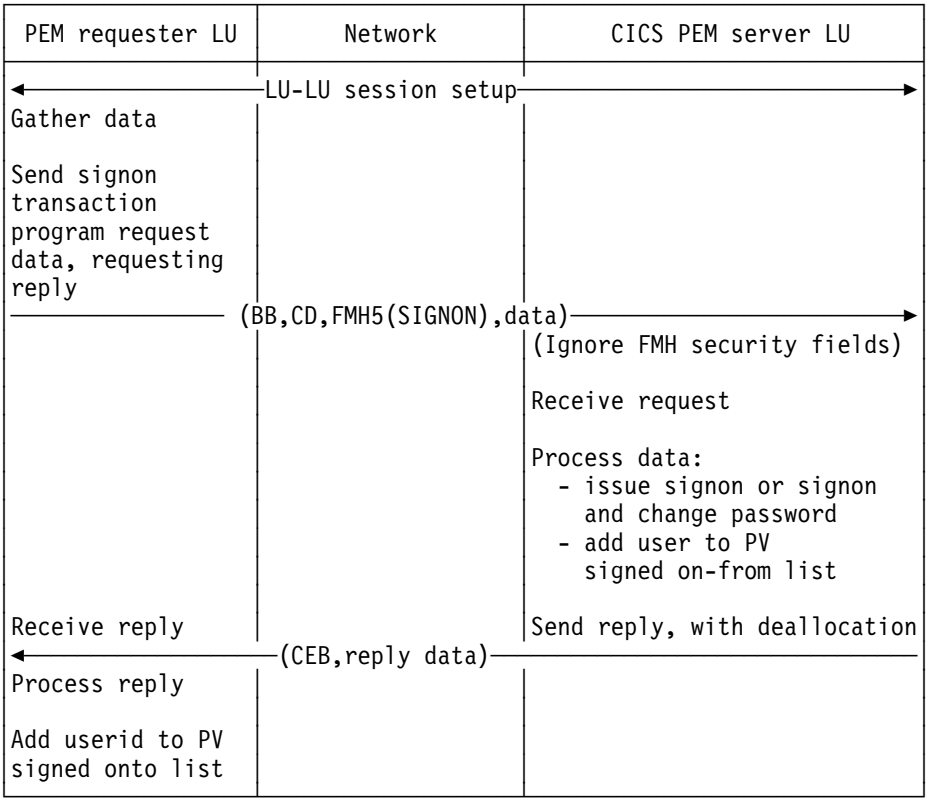

Figure 15. Successful signon — PV connection

Notes:

1. All security fields in the FMH-5 (userid, password and UP, AV, PV1 and PV2 bits) are ignored by the CICS PEM server when it attaches the signon transaction.
2. The CICS PEM server adds the userid to its PV signed-on-from list only if the signon and change password request is successful and either ATTACHSEC=MIXIDPE or ATTACHSEC=PERSISTENT is specified in the CONNECTION definition.
3. The PEM requester must add the userid to its PV signed on-to list only if a successful signon reply is received from the CICS PEM server. The userid has been added to the PV signed on from list in the CICS PEM server, so all subsequent attach requests from this userid can flow as signed on.

Unsuccessful signon — PV connection: Figure 16 shows the expected flows between a PEM requester and CICS PEM server during an unsuccessful signon on a PV connection.

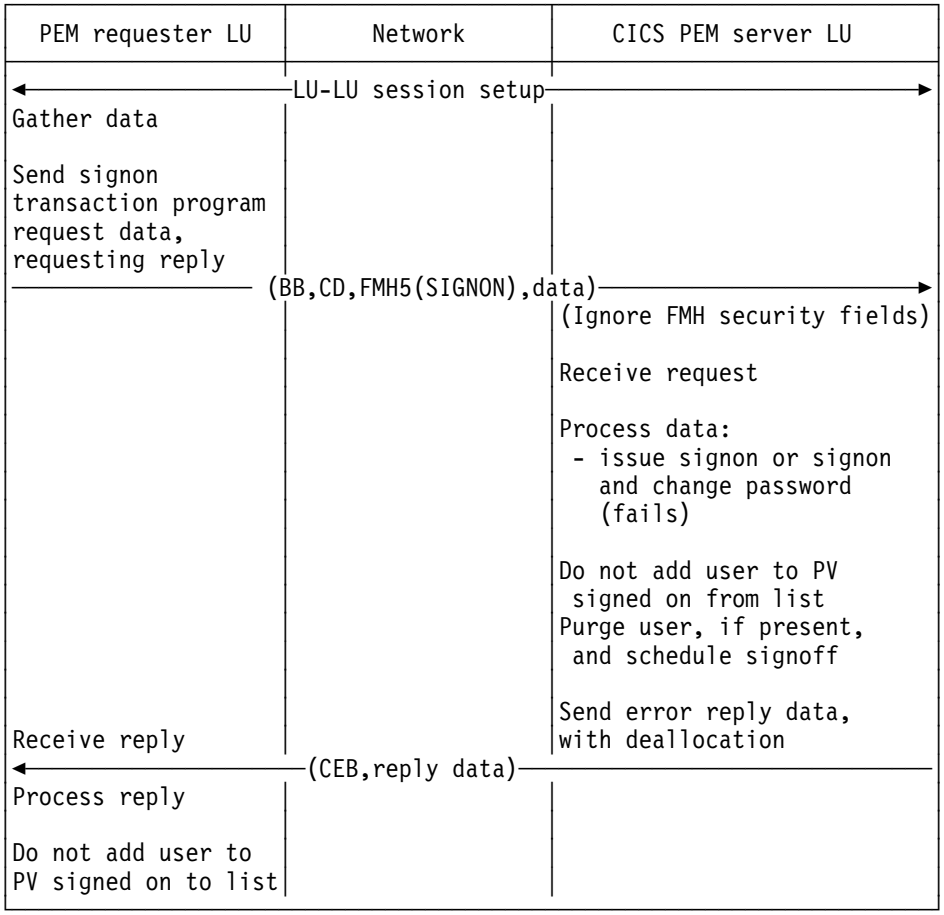

Figure 16. Unsuccessful signon — PV connection

Note: CICS schedules signoff against the PEM requester if a signon request for a userid fails, and that user is in the PV signed on from list. In this case, CICS sends the signoff transaction program output data to the PEM requester, where it is processed and the userid removed from the PV signed on to list.

Setting up the PEM requester

When setting up your PEM requester, you should be aware of the following:

- The conversation type used must be *basic* (also known as **unmapped**). In addition to sending the data you want the partner to receive, you must add control bytes (in assembler language or C/370) to convert the data into an SNA-defined format called a generalized data stream (GDS). You must also include the keyword GDS in any EXEC CICS commands used. See the *CICS/ESA Intercommunication Guide* for introductory information on basic conversations, and the *CICS/ESA Distributed Transaction Programming Guide* for information on using them.

- The SNA service transaction program name for the signon transaction program is **X'06F3F0F1'**, which is also the transaction id (XTRANID) that must be used for the CICS transaction CLS4. (XTRANID is specified in the CICS TRANSACTION definition.)
- The CICS PEM server signon transaction must run as a **sync level 0** transaction. If it is initiated with a sync level other than 0, it sends an ISSUE ABEND and frees the conversation.
- The userid and password must be translated into EBCDIC; if they are not, the ESM cannot recognize them and issues an error. See Figure 45 on page 266 for an example of converting userids and passwords to EBCDIC.
- The ATTACH request for the signon transaction program must specify SECURITY(NONE). CICS ignores any ATTACH security fields passed in the ATTACH function management header, FMH-5, for this transaction.
- CICS does not support the receipt of the PROFILE option in the signon transaction program. If data identifier (ID) X'00' is provided, status value X'06' (incorrect data format) is returned with formatting error X'0002' (precluded structure present), as described in Table 29 on page 177.
- The new password ID, X'06', is permitted and required only with the X'FF01' request data ID. If the new password is provided with a data ID other than X'FF01', status value X'06' (incorrect data format) is returned with formatting error X'0002' (precluded structure present), as described in Table 29 on page 177.
- CICS only supports userids and passwords up to 8 characters long. If the userid or password length (after stripping blanks and nulls) exceeds 8, status value X'06' (incorrect data format) is returned with formatting error X'000F' (data value out of range), as described in Table 29 on page 177.
- Program initialization parameter (PIP) data is optional on the ALLOCATE for the signon transaction, and is ignored if sent.
- If the signon transaction receives a GDS ISSUE SIGNAL command, it is ignored.
- If the CICS PEM server receives a GDS ISSUE ERROR command, it replies with ERROR and frees the conversation.
- If the CICS PEM server receives a GDS FREE command, it frees the conversation. (No diagnostic information about the type of conversation error is provided.)
- The CICS PEM server transaction does not support the receipt of data exceeding the maximum buffer size. If the concatenation bit in the initial LL is set, it is ignored; concatenated data is also ignored.

Format of user data

As part of the general rules for APPC basic conversations, the user data must be in LL-ID-data format (where LL and ID are both two bytes long), and must follow the attach FMH-5 header. As described in Table 25 on page 173, the CICS DFHCLS4 program requires the user input data stream to fit into the format shown in Figure 17 on page 171; if it does not, CICS rejects it.

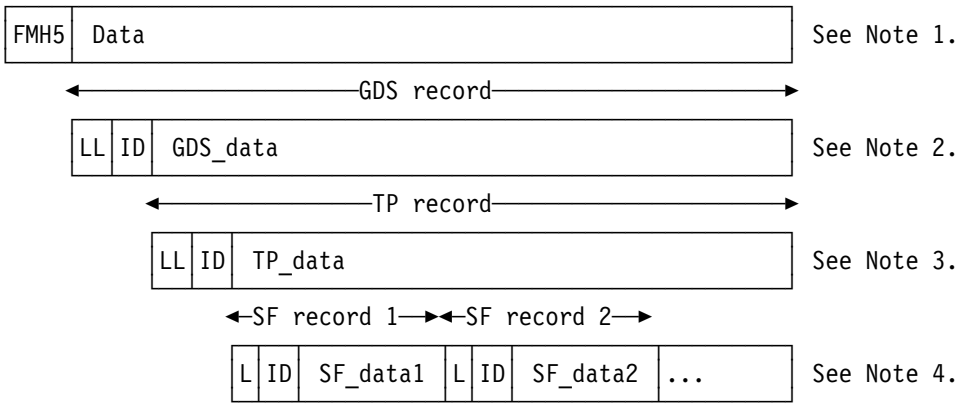

Figure 17. Format of user data

Notes:

1. An attach FMH-5 header with its data. Data is passed between the PEM requester and the CICS PEM server via GDS variables. (For information on GDS, see the *SNA LU 6.2 Peer Protocols* manual.)
2. GDS data in the format LL-ID-data. It is known as the **GDS record**, where:
   - `LL`, which is two bytes long, is the length of the GDS record, including the LL and ID lengths.
   - `ID`, which is two bytes long, indicates what the data record represents; for example, `X'1221'` (signon data).
3. The GDS data record is itself an LL-ID-data record; in this example, a transaction program record (or **TP record**) where:
   - `LL`, which is two bytes long, is the length of the TP record including the LL and ID lengths.
   - `ID` which is two bytes long, indicates the function the TP is to perform; for example, `X'FF00'` (signon) or `X'FF01'` (signon and change password).
4. The TP data record is divided up into L-ID-data records (where L and ID are each **one** byte long). These are known as subfield (or **SF records**) where:
   - `L` is the length of the SF record, including the `L` and `ID` lengths.
   - `ID` indicates the subfield being passed; for example, `X'01'` (userid), `X'02'` (password), and `X'06'` (new password).

PEM requester input and output data

To perform the functions described in “CICS PEM server processing” on page 165, the CICS PEM server needs input data from, and must send output data to, the PEM requester signon transaction program:

- The PEM requester sends data to the CICS PEM server, as described in Table 25 on page 173.
- The CICS PEM server sends data to the PEM requester, as described in Table 26 on page 174 through Table 29 on page 177.

The data must conform to the standards described in “Setting up the PEM requester” on page 169, and its format must be as described in “Format of user data” on page 170. See “Signon with correct userid and password” on page 178 and “Signon with new password” on page 179 for examples of signon output data in GDS flows.

Basic conversation information and data is contained in the attach FMH, as described in “Format of user data” on page 170. The sign on request attaches a transaction X'06F3F0F1', which is the SNA service transaction program name for the signon transaction program.

Signon input data sent by PEM requester

Table 25 shows the input data that the CICS PEM server needs from the PEM requester signon transaction program. See “Signon with correct userid and password” on page 178 and “Signon with new password” on page 179 for examples of signon input data in GDS flows.

Table 25. Signon request and data sent to CICS PEM server

Length (bytes)	Value (if constant)	Meaning
2	X'nnnn'	Length of entire GDS variable, including this 2-byte length value.
2	X'1221'	Data ID for signon data.
2	X'nnnn'	Length of this second (nested) data structure (length, data ID, and data), including this 2-byte length value.
2	X'FF00' or X'FF01'	Data ID for signon or signon and change password request data, respectively. (New password subfield is not permitted for X'FF00'.)
1	X'nn'	Length of subfield for userid, including this 1-byte length value.
1	X'01'	Data ID of subfield for userid.
n	C'xxxxxxxx'	Userid.
1	X'nn'	Length of subfield for password, including this 1-byte length value.
1	X'02'	Data ID of subfield for password.
n	C'xxxxxxxx'	Password.
1	X'nn'	Length of subfield for new password, including this 1-byte length value.
1	X'06'	Data ID of subfield for new password.
n	C'xxxxxxxx'	New password.

Signon output data returned by CICS PEM server

Table 26 lists the signon output data that the CICS PEM server returns to the PEM requester. See “Response to correct signon data” on page 180 and “Response to incorrect data format” on page 182 for examples of signon output data in GDS flows.

Table 26 (Page 1 of 2). Signon output data returned to PEM requester

Length (bytes)	Value (if constant)	Required or optional	Meaning
2	X'nnnn'	Required	Length of entire GDS variable, including this 2-byte length value.
2	X'1221'	Required	Data ID of subfield for signon data.
2	X'nnnn'	Required	Length of this second (nested) data structure (length, data ID, and data), including this 2-byte length value.
2	X'FF02'	Required	Data ID for signon reply data.
1	X'03'	Required	Length of subfield for signon completion status, including this 1-byte length value.
1	X'00'	Required	Data ID of subfield for signon completion status.
1	X'00' through X'06'	Required	Signon completion status — see Table 28 on page 176.
1	X'04'	Optional	Length of subfield for signon request formatting error, including this 1-byte length value.
1	X'01'	Optional	Data ID of subfield for signon request formatting error.
2	X'0000' through X'0003', X'0005' through X'0007', X'000F'.	Optional	Signon request formatting error — see Table 29 on page 177.
1	X'0A'	Optional	Length of subfield for date and time of current successful signon, including this 1-byte length value.
1	X'02'	Optional	Data ID of subfield for date and time of current successful signon.
8	See Table 27 on page 175 for format.	Optional	Date and time of current successful signon. The date and time returned are extracted by the ESM from the user profile.
1	X'0A'	Optional	Length of subfield for date and time of last successful signon, including this 1-byte length value.
1	X'03'	Optional	Data ID of subfield for date and time of last successful signon.

Table 26 (Page 2 of 2). Signon output data returned to PEM requester			
Length (bytes)	**Value (if constant)**	**Required or optional**	**Meaning**
8	See Table 27 on page 175 for format.	Optional	Date and time of last successful signon. The date and time returned are extracted by the ESM from the user profile.
1	X'0A'	Optional	Length of subfield for date and time password will expire, including this 1-byte length value.
1	X'04'	Optional	Data ID of subfield for date and time password will expire.
8	See Table 27 on page 175 for format.	Optional	Date and time password will expire. (The date and time returned are calculated from data obtained from the ESM.)
1	X'04'	Optional	Length of subfield for revoke count, including this 1-byte length value.
1	X'05'	Optional	Data ID of subfield for revoke count.
2	X'nnnn'	Optional	Revoke count.

Format of date and time subfields: Table 27 lists the format of the date and time subfields that the CICS PEM server can return to the PEM requester, as referred to in Table 26 on page 174. See “Response to correct signon data” on page 180 for an example of date and time subfields in a GDS flow.

Table 27. Format of date and time subfields using 24-hour clock	
Position	**Meaning**
00	Two-byte year value; for example, 1994=X'07C8'.
02	One-byte month value; January=X'01', December=X'0C'.
03	One-byte day value; first day=X'01', thirty-first day=X'1F'.
04	One-byte hour value; midnight=X'00', 23rd hour=X'17'.
05	One-byte minute value; on the hour=X'00', 59th minute=X'3B'.
06	One-byte second value; on the minute=X'00', 59th second=X'3B'.
07	One-byte 100ths of a second value; on the second=X'00', maximum=X'63'.

Note: The maximum time value for a given day is 23 hours, 59 minutes, and 59.99 seconds (decimal). Midnight is 0 hours, 0 minutes, and 0 seconds on the following day.

Signon completion status values returned to PEM requester: Table 28 describes the signon completion status values (referred to in Table 26 on page 174) that the CICS PEM server can return to the PEM requester in the status completion subfield in the signon reply data. See "Response to correct signon data" on page 180 for an example of signon completion status values in a GDS flow.

Table 28. Signon completion status values returned to PEM requester

Status value	Meaning
X'00'	All of the following conditions apply: • Userid valid • Password valid • Password not expired or new password specified • Regardless of whether the original password is expired, if a new password is specified, it must be valid. When this status value is returned, the new password is set if specified, and PV processing (if used) is complete.
X'01'	Userid not known to the receiver.
X'02'	Userid valid, password incorrect.
X'03'	Userid valid, password correct but expired. New password must be set.
X'04'	Userid valid, password correct, new password not acceptable to receiving security system.
X'05'	Security function failure. Function not performed.
X'06'	Incorrect data format. Specific error is contained in the signon request formatting error subfield described in Table 29 on page 177.

Note: The CICS PEM server never returns either of the following signon status values to the PEM requester:

- X'07' — general security error
- X'08' — password change completed, but signon failed.

Signon request formatting errors returned to PEM requester: Table 29 lists the signon request formatting error values (referred to in Table 26 on page 174) that the CICS PEM server can return to the PEM requester. Each is a 2-byte binary value. See “Response to incorrect data format” on page 182 for an example of signon request formatting errors in a GDS flow.

Table 29. Signon request formatting error values returned to PEM requester

Error value	Description
X'0000'	Undefined error not described below.
X'0001'	Required structure absent.
X'0002'	Precluded structure present.
X'0003'	Multiple occurrences of a nonrepeatable structure.
X'0005'	Unrecognized structure present where precluded.
X'0006'	Length outside specified range. This value assumes that the overall length arithmetic balances and that the sender intended to send the structure at that length.
X'0007'	Length exception. Length arithmetic is out of balance.
X'000F'	Data value out of range.

Application design

Applications should be designed so that the signon transaction program is run before any transactions that are run by the applications. This means that any password check and any password changing are kept separate from the application's own functions, helping make the application more “user-friendly.” In multitasking systems, it is possible for more than one signon transaction to start on parallel sessions. It then becomes more important for the code that deals with application-level ALLOCATE requests to serialize the signon process to completion, and then both flow as signed-on.

To record the date and time of a previous successful signon, the CICS PEM server signon program extracts password data from the ESM *before* it performs signon. If your system uses shared userids, and two users attempt to sign on at the same time, or if a user is multitasking, the time values returned to the PEM requester for the current signon may not be the same as the timestamp recorded on the ESM database. You should be aware of this if you are writing an application that is to run on multiple systems, and depends on the signon time returned to the PEM requester. (This situation should not apply on a single system, provided the signon process is serialized as suggested.)

If PV is being used, and the interval specified in PVDELAY is exceeded, and the userid is removed from the PV sign on from list, applications must allow for the signon process to be serialized again.

Examples of PEM requester and CICS PEM server user data

Data is passed between the PEM requester and the CICS PEM server via GDS variables. To help you check the data being sent by your PEM requester, the examples that follow show:

- "Signon with correct userid and password"
- "Signon with new password" on page 179
- "Response to correct signon data" on page 180
- "Response to incorrect data format" on page 182.

These examples are produced by the example PEM requester program shown in Figure 45 on page 266. The example program in Figure 45 on page 266 uses a **partner_LU_alias** of `hostcics`, an **LU_alias** of `ps2lua`, and a **mode_name** of `lu62ss`. When writing your own PEM requester program, you will need to use the values defined in your communications manager configuration.

Signon with correct userid and password: Figure 18 shows an example flow for a successful signon.

```
PEM hostcics ps2lua lu62ss sec2r01 drtnnom
```

Figure 18. Correct userid and password, no new password

A valid userid (`sec2r01`) and password (`drtnnom`) are correctly entered. No new password is entered.

The following hexadecimal user data is sent to the CICS PEM server:

`001A12210016FF000901E2C5C3F2D9F0F10902C4D9E3D5D5D6D4`

This contains the following values, as described in Table 25 on page 173:

`001A` Length of the entire GDS variable, including this 2-byte length value

`1221` Data ID for sign on data

`0016` Length of this second (nested) data structure (length, data ID, and data), including this 2-byte length value

`FF00` Data ID for **signon** request data

`09` Length of subfield for userid, including this 1-byte length value

`01` Data ID of subfield for userid

`E2C5C3F2D9F0F1` Userid (`SEC2R01`) in EBCDIC

`09` Length of subfield for password, including this 1-byte length value

`02` Data ID of subfield for password

`C4D9E3D5D5D6D4` Password (`DRTNNOM`) in EBCDIC

Signon with new password: Figure 19 shows an example flow for a successful signon using a new password.

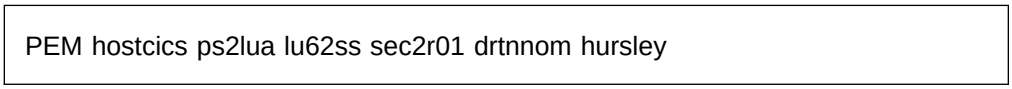

Figure 19. Signon with new password

A userid, password, and new password are correctly entered.

The following hexadecimal user data is sent to the CICS PEM server:

`00231221001FFF010901E2C5C3F2D9F0F10902C4D9E3D5D5D6D40906C8E4D9E2D3C5E8`

This contains the following values, as described in Table 25 on page 173:

`0023` Length of entire GDS variable, including this 2-byte length value

`1221` Data ID for sign

`001f` Length of this second (nested) data structure (length, data ID, and data), including this 2-byte length value

`FF01` Data ID for signon and change password request data

`09` Length of subfield for userid, including this 1-byte length value

`01` ID of subfield for userid

`E2C5C3F2D9F0F1` Userid (`SEC2R01`) in EBCDIC

`09` Length of subfield for password, including this 1-byte length value

`02` ID of subfield for password

`C4D9E3D5D5D6D4` Password (`DRTNNOM`) in EBCDIC

`09` Length of subfield for new password, including this 1-byte length value

`06` ID of subfield for new password

`C8E4D9E2D3C5E8` New password (`HURSLEY`) in EBCDIC

Response to correct signon data: Figure 20 shows an example of the response to the correct signon data being entered.

```
PEM_OK
GDS LLID
00 2d 12 21
Signon Reply LLID
00 29 ff 02
Signon Completion Status Subfield
03 00 00
Date & Time of Current Successful Signon Subfield
0a 02 07 ca 01 14 0d 24 31 62
Date & Time of Last Successful Signon Subfield
0a 03 07 ca 01 11 16 1b 23 3e
Date & Time Password Will Expire Subfield
0a 04 07 ca 02 03 00 00 00 00
Revoke Count Subfield
04 05 00 00
```

Figure 20. Response to correct signon data

The first three lines of hexadecimal user data returned to the PEM requester show the following *required* values, as described in Table 26 on page 174.

`002d` Total length of the GDS variable, including this 2-byte length value

`1221` Data ID for signon data

`0029` Length of this second (nested) data structure (length, data ID, and data), including this 2-byte length value

`FF02` Data ID for signon reply data

`03` Length of subfield for signon completion status, including this 1-byte length value

`00` Data ID for signon completion status

`00` Signon completion status. 00 indicates that the userid and password were valid, and the password had not expired. (See Table 28 on page 176 for a list of signon completion status values.)

In Figure 20 on page 180, the last four lines of hexadecimal user data returned to the PEM requester show the following optional values, as described in Table 26 on page 174. (Note that the formatting error subfields shown in Table 26 on page 174 are not included, indicating that there are no errors.)

`0A` Length of subfield for date and time of current successful signon including this 1-byte length value

`02` Data ID for date and time of current successful signon

Date and time of current successful signon, as described in Table 27 on page 175:

- `07CA` Year (1994)
- `01` Month (July)
- `14` Day (20)
- `0D` Hour (13)
- `24` Minutes (36)
- `31` Seconds (49)
- `62` Hundredths of a second (98)

`0A` Length of subfield for date and time of previous successful signon,

`03` Data ID for date and time of previous successful signon

Date and time of previous successful signon, as described in Table 27 on page 175:

- `07CA` Year (1994)
- `01` Month (January)
- `11` Day (17)
- `16` Hour (22)
- `1B` Minutes (27)
- `23` Seconds (35)
- `3E` Hundredths of a second (62)

`0a` Date and time password will expire (including this 1-byte length value)

`04` Length of subfield for data ID for date and time password will expire

Date and time password will expire, as described in Table 27 on page 175:

- `07ca` Year (1994)
- `02` Month (February)
- `0e` Day (14)
- `00` Hour (00)
- `00` Minutes (00)
- `00` Seconds 00)
- `00` Hundredths of a second (00)

`04` Length of subfield for revoke count, including this 1-byte length value

05 Data ID of subfield for revoke count

0000 Revoke count. (0000 means that there have been no unsuccessful signon attempts since the last successful signon with this userid.)

Response to incorrect data format: Figure 21 shows an example flow in response to incorrect data being entered.

```
PEM_OK
GDS LLID
00 0F 12 21
Signon Reply LLID
00 0B FF 02
Signon Completion Status Subfield
03 00 06
Signon Request Formatting Error Subfield
04 01 00 0F
```

Figure 21. Response to incorrect data format

The first three lines of hexadecimal user data returned to the PEM requester show the following required values, as described in Table 26 on page 174:

000F Length of entire GDS variable, including this 2-byte length value

1221 Data ID for signon data

000B Length of this second (nested) data structure (length, data ID, and data), including this 2-byte length value

FF02 Data ID for signon reply data

03 Length of subfield for signon completion status, including this 1-byte length value

00 Data ID of subfield for signon completion status

06 Signon completion status 06 indicates incorrect data format. (See Table 28 on page 176 for a list of signon completion status values.)

The last line of hexadecimal user data returned to the PEM requester shows the following **optional** values, which are returned only if there is an error. (The optional values are described in Table 26 on page 174.)

04 Length of subfield for signon request formatting error, including this 1-byte length value

01 Data ID of subfield for signon request formatting error

000F Signon request formatting error, indicating “data value out of range” (See Table 29 on page 177 for a description of other possible formatting errors.)

End of General-use programming interface

Chapter 14. Implementing LU6.1 security

This chapter tells you how to implement link security for LU6.1, and covers the following topics:

For LU6.1 links, CICS cannot check the identity of the requesting system, and the bind request is never rejected on security grounds. You are advised to use the intersystem security offered by LU6.2 links whenever possible. Note that no bind-time or user security can be applied to LU6.1 links.

Link security with LU6.1

Link security restricts the resources that a user can access, depending on the remote system from which they are accessed. The practical effect of link security is to prevent a remote user from attaching a transaction or accessing a resource for which the link userid has no authority.

Each link between systems is given an access authority defined by a link userid. A link userid for LU6.1 is a userid defined on your sessions definition for this connection. If not defined there, the link userid is taken to be the SECURITYNAME userid specified on the connection definition. If there is no SECURITYNAME, the link userid is the local region's default userid.

You cannot function ship to CICS without having a security check. However, the security check is minimized if the two regions involved are **equivalent systems**. This term means the same for LU6.1, LU6.2 and MRO: that the link userid matches the local region's userid.

If you have equivalent systems the resource check is made against the local region's default user. If you do not have equivalent systems, the resource check is carried out against the link userid.

If a failure occurs in establishing link security, the link is given security of the local region's default user. This would happen if, for example, the preset session userid had been revoked.

Specifying ATTACHSEC with LU6.1

With LU6.1 links, information about the remote user is not available for security purposes. In this case, the authority of the user is taken to be that of the link itself, and you must rely on link security alone to protect your resources.

With LU6.1, you can only specify ATTACHSEC(LOCAL) in the CONNECTION definition. Figure 22 shows an example of doing this using CEDA.

```
CEDA DEFINE CONNECTION(name)
  GROUP(groupname)
  .
  ATTACHSEC(LOCAL)
```

Figure 22. Defining signon level for user security with LU6.1

LOCAL is the default value. It specifies that a user identifier is not required from the remote system, and if one is received, it is ignored. Here, CICS makes the user security profile equivalent to the link security profile. You do not need to specify RACF profiles for the remote users.

Transaction, resource, and command security with LU6.1

As in a single-system environment, links must be authorized to:

- Attach a transaction
- Access all the resources that the transaction is programmed to use.

This results in security levels called **transaction security**, **resource security**, and **command security**.

Transaction security

As in a single-system environment, the security requirements of a transaction are specified when the transaction is defined, as described in Chapter 5, “Transaction security” on page 77.

In an LU6.1 environment, a transaction can be initiated, only if the link has sufficient authority.

Resource and command security

Resource and command security in an intercommunication environment are handled in much the same way as in a single-system environment.

Resource and command security checking are performed only if the installed TRANSACTION definition specifies that they are required; for example, on the CEDA DEFINE TRANSACTION command, as shown in Figure 23 on page 185.

```
CEDA DEFINE TRANSACTION
  .
  RESSEC(YES)
  CMDSEC(YES)
  .
```

Figure 23. Specifying resource and command security for transactions

If a transaction definition specifies resource security checking, using RESSEC(YES), the link must have sufficient authority for the resources that the attached transaction accesses.

If a transaction definition specifies command security checking, using CMDSEC(YES), the link must have sufficient authority for the commands (COLLECT, DISCARD, INQUIRE, PERFORM, and SET) that the attached transaction issues.

For further guidance on specifying resource and command security, see Chapter 6, "Resource security" on page 83 and Chapter 8, "CICS command security" on page 109.

NOTAUTH exceptional condition

If a transaction tries to access a resource, but fails the resource security checks, the NOTAUTH condition is raised.

When the transaction is the CICS mirror transaction, the NOTAUTH condition is returned to the requesting transaction, where it can be handled in the usual way.

Function shipping security with LU6.1

When CICS receives a function-shipped request, the transaction that is invoked is the **mirror transaction**. The CICS-supplied definitions of the mirror transactions all specify resource security checking, but not command security checking. This means that you are prevented from accessing the remote resources if the link does not have the necessary authority.

| Note that **transaction routing** across LU6.1 links is not supported.

If the CICS-supplied definitions of the mirror transactions are not what your security strategy needs, you can change them by copying the definitions in group DFHISC into your own group, changing them and then reinstalling them. For more information, see "Category 2 transactions" on page 127.

If you include a remote resource in your resource definitions, you can arrange for security checking to be done locally, just as if the resource were a local one. Also, the system that owns the resource can be made to apply an independent check, if it is able to receive the user identifier. You can therefore choose to apply security restrictions on both sides, on either side, or not at all.

Take care when you specify the SYSID option on a function-shipped request. Security checking is done in the remote system but is *bypassed in the local system*. Figure 24 on page 186 summarizes what happens.

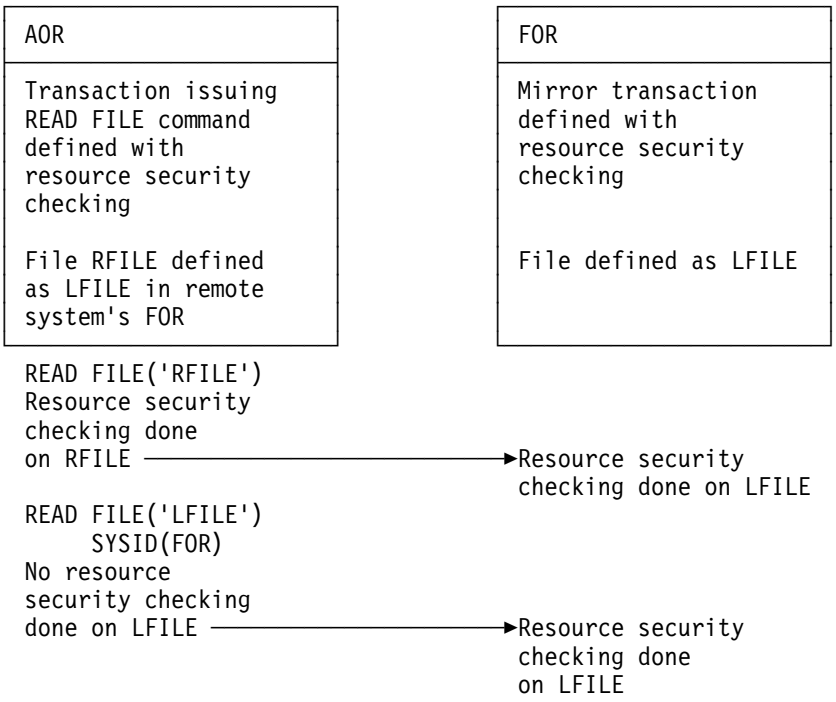

Figure 24. Security checking done with and without SYSID

For programming information on specifying the SYSID option, see the *CICS/ESA Application Programming Reference* manual.

Security checking done in AOR with LU6.1

This section summarizes how security checking is done in the AOR according to how SECURITYNAME is specified in the AOR and TOR, in an LU6.1 environment.

The link userid referred to in Table 30 on page 187 is the one specified in the SECURITYNAME on the CONNECTION definition, or the USERID on the SESSIONS definition.

If a USERID is specified on the SESSIONS definition, and a link check is done, the userid used is the one on the SESSIONS definition.

Table 30 on page 187 shows how checking is done when ATTACHSEC(LOCAL) is specified.

Neither the region userid for the TOR, nor the SECURITYNAME in the TOR's CONNECTION definition for the AOR, are relevant to security checking in the AOR.

Table 30. Security checking done in AOR			
Region userid for AOR	**SECURITYNAME in CONNECTION definition**	**USERID in SESSION definition**	**Checking in AOR**
USERIDA	Not specified	Not specified	Check against AOR DFLTUSER
USERIDA	Not specified	USERIDA	Check against AOR DFLTUSER
USERIDA	Not specified	USERIDB	Check against USERIDB
USERIDA	USERIDA	Not specified	Check against AOR DFLTUSER
USERIDA	USERIDB	Not specified	Check against USERIDB
USERIDA	USERIDA	USERIDA	Check against AOR DFLTUSER
USERIDA	USERIDA	USERIDB	Check against USERIDB
USERIDA	USERIDB	USERIDA	Check against AOR DFLTUSER
USERIDA	USERIDB	USERIDB	Check against USERIDB
USERIDA	USERIDB	USERIDC	Check against USERIDC

Summary of resource definition options for LU6.1 security

The following is a summary of the resource definition options you need to define for LU6.1 security.

- On the CONNECTION definition:
  - ATTACHSEC, with the LOCAL option specified or allowed to default
  - SECURITYNAME
- On the SESSIONS definition:
  - USERID

For guidance on specifying CONNECTION and SESSION definitions, see the *CICS/ESA Resource Definition Guide*.

Chapter 15. Implementing MRO security

This chapter tells you how to implement MRO security, and is organised as follows:

Security implications of choice of MRO access method

Either MVS cross-memory services or the CICS Type 3 SVC can be used for interregion communication (function shipping, transaction routing, distributed transaction processing, and asynchronous processing).

If you use cross-memory services, you lose the total separation between systems that is normally provided by separate address spaces.

The risk of accidental interference between two CICS address spaces connected by a cross-memory link is small. However, an application program in either system could access the other system's storage (subject to key-controlled protection) by using a sequence of cross-memory instructions.

If this situation would create a security exposure in your installation, use the CICS type 3 SVC for interregion communication, rather than MVS cross-memory services.

For information about how to specify the access method for MRO, see the *CICS/ESA Intercommunication Guide*.

Bind-time security with MRO

The CICS interregion communication (IRC) facility supports CICS multiregion operation (MRO) through the use of DFHAPPL.*applid* profiles in the FACILITY class.

There are two phases to bind security checking in DFHIRP, and these occur at:

- Logon time
- Connect time.

These security checks, via RACROUTE calls to the SAF interface, are always performed, regardless of whether the MRO partner regions are running with

external security active for CICS resource security checking (that is, for both SEC=YES and SEC=NO). In order for an MRO connection to be established between two regions, both the logon and connect security checks in both systems must be completed successfully. This security is applied to earlier releases of CICS using the CICS/ESA 4.1 version of DFHIRP, the CICS interregion communication program.

It is important to consider that *all* of the implications of bind-time security are considered on installation of CICS/ESA 4.1 (with DFHIRP 4.1). For further information see "Connect security."

Logon security checking

Logon security checking is performed whenever interregion communication is opened by a CICS region, which causes the region to log on to DFHIRP.

CICS interregion communication uses the external security manager to check that CICS regions logging on to IRC are the regions they claim to be.

Each region that logs on to DFHIRP must be authorized to RACF in a DFHAPPL.*applid* profile in the RACF FACILITY class. This requires the definition of a DFHAPPL.*applid* profile for each region that logs on to DFHIRP, and that each CICS region userid has UPDATE access to its own DFHAPPL.*applid* profile.

See Figure 25 for an illustration of logon checking.

Connect security

To perform MRO connect security checking, DFHIRP checks that each CICS region in the connection has read access to its partner's DFHAPPL.*applid* profile.

When CICS/ESA 4.1 DFHIRP is installed, all regions using earlier CICS releases in the MVS image use the DFHAPPL.*applid* form of MRO connect security. In addition, the SECURITYNAME parameter on the CONNECTION definition is obsolete and is ignored.

To authorize the MRO partner regions for bind security purposes, you must define the appropriate DFHAPPL profiles in the RACF FACILITY class. This means that each CICS region in an MRO interregion communication link must be given access to its partner's DFHAPPL.*applid* profile with READ access authority. For example, for the CICS TOR running under userid CICSRTOR (with APPLID CICSATOR), that connects to the AOR running under userid CICSRAOR (with APPLID CICSAAOR), the RACF commands to authorize the connections are shown in Figure 25 on page 191.

You cannot specify to CICS whether or not you want connect security checking for MRO connections—CICS always issues the RACROUTE calls.

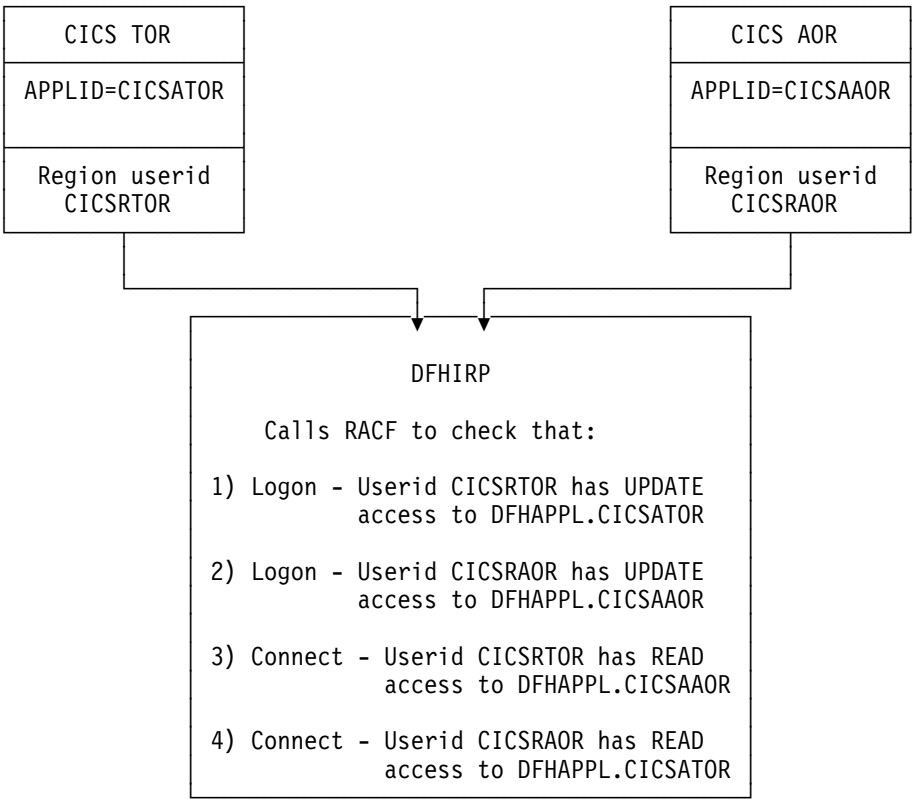

Figure 25. Illustration of the DFHIRP logon and connect security checks

The TOR and AOR shown in Figure 25, running under region userids CICSRTOR and CICSRAOR respectively, with APPLIDs CICSATOR and CICSAAOR, require the following RACF definitions to authorize their logon to DFHIRP:

An example of the MRO logon and connect process:

```
RDEFINE FACILITY (DFHAPPL.CICSATOR) UACC(NONE)
RDEFINE FACILITY (DFHAPPL.CICSAAOR) UACC(NONE)

PERMIT DFHAPPL.CICSATOR  CLASS(FACILITY) ID(CICSRTOR) ACCESS(UPDATE)
PERMIT DFHAPPL.CICSAAOR  CLASS(FACILITY) ID(CICSRAOR) ACCESS(UPDATE)
```

An example to show connection:

```
PERMIT DFHAPPL.CICSAAOR  CLASS(FACILITY) ID(CICSRTOR) ACCESS(READ)
PERMIT DFHAPPL.CICSATOR  CLASS(FACILITY) ID(CICSRAOR) ACCESS(READ)
```

Responses from the system authorization facility (SAF)

If the security profile for a specified resource is not retrieved, SAF neither grants nor refuses the access request. In this situation:

DFHIRP rejects the logon or connect request if:

- A security manager is installed, but is either temporarily inactive or inoperative for the duration of the MVS image. This is a fail-safe action, on the grounds

that, if the security manager was active, it might retrieve a profile that does not permit access.

DFHIRP allows the logon or connect request if:

- There is no security manager installed, or
- There is an active security manager, but the FACILITY class is inactive, or there is no profile in the FACILITY class. The logon is allowed in this case because there is no evidence that you want to control access to the CICS APPLID.

Any CICS region without a specific DFHAPPL.*applid* profile, or applicable generic profile, permits all logon and connect requests. No messages are be issued to indicate this. To avoid any potential security exposures, you can use generic profiles to protect all, or specific groups of, regions before, or in parallel with, security measures for specific regions. For example, specifying

```
RDEFINE FACILITY (DFHAPPL.*) UACC(NONE)
```

would ensure that any region without a more specific profile is prevented from binding.

Link security with MRO

Link security restricts the resources that a user can access, depending on the remote system from which they are accessed. The practical effect of link security is to prevent a remote user from attaching a transaction or accessing a resource for which the link userid has no authority.

Each link between systems is given an access authority defined by a link userid. A link userid for MRO is a userid defined on your sessions definition for this connection. Note that for MRO, unlike LU6.2, you can only have one sessions definition per connection, and there can only be one link userid per connection. If there is no preset session userid, the link userid is taken to be the region userid of the TOR region. The SECURITYNAME field on the connection definition is ignored for MRO.

You can never transaction route or function ship to CICS without having at least one security check, but the security checks done are minimized if the two regions involved are *equivalent systems*. This term means the same thing for LU6.1, LU6.2 and MRO: that the link userid matches the local region's userid.

If you have equivalent systems you will always only have one security check. This will either be made against the local region's default user (for ATTACHSEC=LOCAL) or against the userid in the received FMH-5 attach request (ATTACHSEC=IDENTIFY).

If you do not have equivalent systems for ATTACHSEC=LOCAL, resource checks are done only against the link userid. For ATTACHSEC=IDENTIFY you will always have two resource checks. One check is against the link userid and the other is against the userid received from the remote user in the attach request.

If a failure occurs in establishing link security, the link is given the same security authorization as defined for the local region's default user. This would happen, for example, if the preset session userid had been revoked.

The SESSIONS definition must be associated with a RACF user profile. This user profile must have access to any protected resource to which the inbound transaction needs access. See Chapter 2, “RACF facilities” on page 9 for guidance on defining profiles.

If the signon fails, a signon failure message is sent to the CSCS security destination, and the link is given the security of the DFLTUSER in the receiving system; that is, it is able to access only those resources to which the default user has access.

Obtaining the CICS region userid

For the purposes of MRO logon and connect security checks, DFHIRP needs to know the CICS region userid under which the CICS job, or task, is running. DFHIRP obtains the CICS region's userid by issuing a RACROUTE REQUEST=EXTRACT macro.

If you are not using RACF as your external security manager, you must use the MVS security router exit, ICHRTX00, to customize the response from the RACROUTE REQUEST=EXTRACT macro.

CICS determines whether a security manager is present or not by examining the SAF response codes.

User security with MRO

User security causes a second check to be made against a user signed on to a terminal in addition to the link security check described in “Link security with MRO” on page 192. You should consider whether you want the extra level of security checking that user security provides.

You can specify either LOCAL, in which the user is not checked, or IDENTIFY, in which a userid is required, but no password is sent.

You specify the signon support for each connection using the ATTACHSEC operand of CONNECTION definition, as described in “User security in link definitions.”

User security in link definitions

The level of user security you require for a remote system is specified in the ATTACHSEC operand of the CONNECTION definition. Figure 26 shows an example of defining ATTACHSEC using CEDA.

CICS interprets the parameters of the ATTACHSEC operand as described here. However, special rules apply for CICS transaction routing using CRTE, as described in “CICS routing transaction, CRTE” on page 198.

```
CEDA DEFINE CONNECTION(name)
  GROUP(groupname)
  .
  ATTACHSEC(LOCAL|IDENTIFY)
```

Figure 26. Defining sign-on level for user security

The ATTACHSEC operand specifies the signon requirements for incoming requests. It has no effect on requests that are issued by your system to a remote system; these are dealt with by the remote system.

The following ATTACHSEC operands are valid with MRO:

LOCAL
specifies that a user identifier is not required from the remote system, and if one is received, it is ignored. Here, CICS makes the user security profile equivalent to the link security profile. You do not need to specify RACF profiles for the remote users. (LOCAL is the default value.)

Specify ATTACHSEC(LOCAL) if you think that the link security profile alone provides sufficient security for your system.

IDENTIFY
specifies that a user identifier is expected on every attach request. All remote users of a system must be identified to RACF.

If an attach request with both a user identifier and a password is received on a link with ATTACHSEC(IDENTIFY), CICS rejects the attach request and unbinds the session.

If a null (X'00') character user identifier or an unknown user identifier is
received, CICS rejects the attach request. If no user identifier is received, the
attach is rejected unless USEDFLTUSER(YES) is specified on the connection.
In this case CICS applies the security capabilities of the default user, as
specified in the DFLTUSER system initialization parameter. For more
information, see “CICS default user” on page 17, and “Attach-time security and
the USEDFLTUSER option” on page 153.

Note: In the case of distributed transaction processing (DTP) transactions, you must issue a BUILD ATTACH request before the MRO SEND or CONVERSE command to include the userid of the terminal user in an attach request.

Signon status

With ATTACHSEC(IDENTIFY), the remote user remains signed-on after the conversation associated with the first attach request is complete. CICS then accepts attach requests from the same user without a new signon until either of the following occurs:

- | The period specified in the system initialization parameter USRDELAY elapses
  | after completion of the last transaction associated with the attach request for this user.

  | When you are running remote transactions, over ISC and IRC links,
  | USRDELAY defines the length of time for which entries can remain signed onto
  | the remote CICS region. For information on specifying USRDELAY, see the
  | *CICS/ESA System Definition Guide*. For information on tuning, see the *CICS/ESA Performance Guide*
- The CICS system is terminated.

If you alter the RACF profile of a signed-on remote user (for example, by revoking the user), CICS continues to use the authorization established at the first attach request until the user is signed off by one of the events just described.

Information about remote users

With MRO links, information about the user can be transmitted with the attach request from the remote system. This means that you can protect your resources not only on the basis of which remote system is making the request, but also on the basis of which actual user at the remote system is making the request.

This section describes some of the concepts associated with remote-user security, and how CICS sends and receives user information.

You will have to define your users to RACF. If a remote user is not defined to RACF, any attach requests from that remote user are rejected.

APAR PQ23001

MJO 1/7/99

User profiles can be transmitted instead of, or in addition to, user identifiers. The profile name, if supplied, is treated as the groupid.

If the user has been added to the front-end system with a groupid explicitly specified; for example, in EXEC CICS SIGNON, or by filling in the GROUPID parameter on the CESN panel, this will be propagated by CICS in outbound attach FMHs for MRO links when ATTACHSEC(IDENTIFY) has been specified in the CONNECTION definition. If the groupid has been allowed to default at the time the user was originally added to the front-end system, the profile field will not be included in the outbound FMH5. If the groupid is passed to the back-end system, the groupid will be used as part of ADD_USER processing on the back-end, (meaning that the userid must be defined as a member of the group passed in the ESM on the back-end for the ADD_USER to be successful).'

CICS sends userids on ATTACHSEC(IDENTIFY) conversations. Table 31 on page 196 shows how CICS decides what userid to send.

Table 31. MRO attach-time user identifiers	
Characteristics of the local task	**User identifier sent by the TOR to the AOR**
Task with associated terminal — user identifier	Terminal user identifier
Task with associated terminal — no user signed on and no USERID specified in the terminal definition	Default user identifier from the TOR
Task with no associated terminal or USERID started by interval control START command (if using function shipping or DTP)	User identifier for the task that issued the START command
Task started with USERID option	User identifier specified on the START command
CICS internal system task	CICS region userid
Task with no associated terminal started by transient data trigger	User identifier specified on the DCT that defines the queue
Task with associated terminal started by transient data trigger	Terminal user identifier
Task started from PLTPI	User identifier specified by the PLTPIUSR system initialization parameter

Transaction, resource, and command security with MRO

As in a single-system environment, users must be authorized to:

- Attach a transaction.
- Access all the resources that the transaction is programmed to use. This results in security levels called transaction security, resource security, surrogate user security, and command security.

Transaction security

As in a single-system environment, the security requirements of a transaction are specified when the transaction is defined, as described in Chapter 5, “Transaction security” on page 77.

In an MRO environment, two basic security requirements must be met before a transaction can be initiated:

- The link must have sufficient authority to initiate the transaction.
- The “user” who is making the request must have sufficient authority to access the system and to initiate the transaction.

Resource and command security

Resource and command security in an intercommunication environment are handled in much the same way as in a single-system environment.

When resource and command security checking are performed.

Resource and command security checking are performed only if the installed transaction definition specifies that they are required; for example, on the CEDA DEFINE TRANSACTION command, as shown in Figure 27.

```
CEDA DEFINE TRANSACTION
  .
  RESSEC(YES)
  CMDSEC(YES)
  .
```

Figure 27. Specifying resource and command security for transactions

If a transaction specifies resource security checking, using RESSEC(YES), both the link and the user must also have sufficient authority for the resources that the attached transaction accesses.

If a transaction specifies command security checking, using CMDSEC(YES), both the link and the user must also have sufficient authority for the commands (shown in Table 11 on page 109) that the attached transaction issues.

For further guidance on specifying resource and command security, see Chapter 6, “Resource security” on page 83 and Chapter 8, “CICS command security” on page 109.

NOTAUTH exceptional condition

If a transaction tries to access a resource, but fails the resource security checks, the NOTAUTH condition is raised.

When the transaction is the CICS mirror transaction, the NOTAUTH condition is returned to the requesting transaction, where it can be handled in the usual way.

Transaction routing security with MRO

In transaction routing, the authority of a user to access a transaction can be tested in both the TOR and the AOR.

In the TOR, a normal test is made to ensure that the user has authority to access the transaction defined as remote, just as if it were a local transaction. This test determines whether the user is allowed to run the relay program.

In the AOR, the transaction has as its principal facility a remote terminal (the “surrogate” terminal) that represents the “real” terminal in the TOR. The way in which the remote terminal is defined (see the *CICS/ESA Intercommunication Guide*) affects the way in which user security is applied.

- If the definition of the remote terminal does not specify the USERID parameter:
  - For links with ATTACHSEC(IDENTIFY), the transaction security and resource security of the user are established when the remote user is signed on. The userid under which the user is signed on, whether explicitly or implicitly (in the DFLTUSER system initialization parameter), has this security capability assigned in the remote system.

– For links with ATTACHSEC(LOCAL), transaction security, command security, and resource security are limited by the authority of the link.

In both cases, tests against the link security are made as described in “Link security with MRO” on page 192.

Note: During transaction routing, the 3-character operator identifier from the TOR is transferred to the surrogate terminal entry in the AOR. This identifier is not used for security purposes, but it may be referred to in messages and audit trails.

When transaction routing a PSB request, the following conditions must both be satisfied:

- ATTACHSEC on the connection definition must not be LOCAL (that is, it can be IDENTIFY, PERSISTENT, MIXIDPE, or VERIFY)
- PSBCHK=YES must be specified as a system initialization parameter in the remote system.

Preset security terminals and transaction routing

Preset security for a terminal is determined by the specification of the USERID parameter.

When considering the security aspects of transaction routing from a preset security terminal, you must remember that preset security is an attribute of the terminal rather than of the user who is performing the transaction routing request.

During transaction routing, a surrogate terminal is created in the AOR to represent the terminal at which the transaction routing request was issued. Whether the surrogate terminal has preset security or not depends upon a number of factors:

- If a remote terminal definition exists in the AOR for the terminal at the TOR, and specifies the USERID parameter, the surrogate terminal is preset with this userid. If the USERID parameter is not coded, the surrogate terminal does not have preset security.
- If a remote terminal definition does not exist in the AOR, the preset security characteristics of the surrogate terminal are determined from the terminal definition shipped from the TOR. If the shipped terminal definition has preset security, the surrogate also has preset security, unless the connection to the AOR is defined with ATTACHSEC=LOCAL, in which case any preset security information shipped to the AOR is ignored.

CICS routing transaction, CRTE

You can use the CICS routing transaction, CRTE, with MRO to run transactions that reside on a connected remote system, instead of defining these transactions as remote in the local system. CRTE is particularly useful for infrequently used transactions, or for transactions such as CEMT that reside on all systems.

The terminal through which CRTE is invoked must be defined on the remote system (or defined as “shippable” in the local system) and the terminal operator needs RACF authority if the remote system is protected.

Security checking done in the AOR for CRTE does not depend on what is specified on ATTACHSEC. Instead, security checking depends on whether the user signs on while using CRTE:

- *If the user does not sign on*, the surrogate terminal created is associated with the AOR default user. When a transaction is run, the security checks are carried out against this default user. A check is also done against the link userid to see whether the routing application itself has authority to access the resource.
- *If the user does sign on*, using the CESN transaction while running CRTE, the surrogate points to the userid of the signed-on user. For transactions attempting to access resources, security checking is done against the signed-on user's userid in the surrogate and the link userid.

For more information on CRTE, see the *CICS/ESA CICS-Supplied Transactions* manual and the *CICS/ESA Intercommunication Guide*.

Function shipping security with MRO

When CICS receives a function-shipped request, the transaction that is invoked is the **mirror transaction**. The CICS-supplied definitions of the mirror transactions all specify resource security checking, but not command security checking. This means that you are prevented from accessing the remote resources if either the link or your user profile on the other system do not have the necessary authority.

If the CICS-supplied definitions of the mirror transactions are not what your security strategy needs, you can change them by copying the definitions in group DFHISC into your own group, changing them and then reinstalling them. For more information, see "Category 2 transactions" on page 127.

If you include a remote resource in your resource definitions, you can arrange for security checking to be done locally, just as if the resource were a local one. Also, the system that owns the resource can be made to apply an independent check, if it is able to receive the user identifier. You can therefore choose to apply security restrictions on both sides, on either side, or not at all.

Note: If you specify the SYSID option on a function-shipped request, security checking is done in the remote system but is *bypassed in the local system*. Figure 28 on page 200 summarizes what happens.

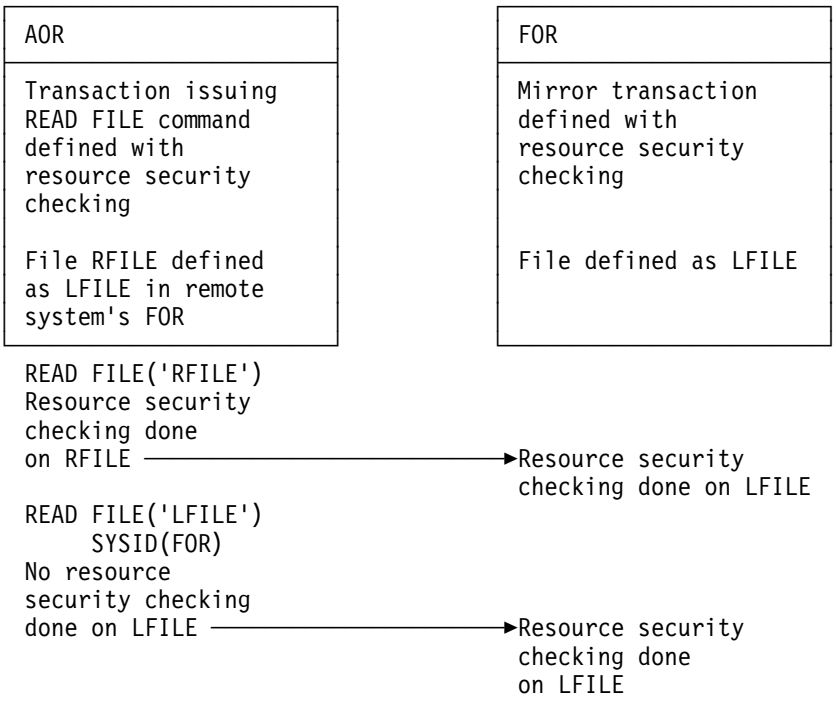

Figure 28. Security checking done with and without SYSID

For programming information on specifying the SYSID option, see the *CICS/ESA Application Programming Reference* manual.

Distributed program link security with MRO

The CICS distributed program link (DPL) facility enables a program (the client program) to call a CICS program (the server program) in a remote CICS region. The client program may be a CICS program or a non-CICS program.

A CICS client program uses DPL by specifying the SYSID option on the EXEC CICS LINK PROGRAM command, or omitting the SYSID option if the REMOTESYSTEM option of the program resource definition already specifies a remote CICS region. When the SYSID option on the EXEC CICS LINK command specifies a remote CICS system, the client region does not perform any resource security checking, but leaves the resource check to be performed in the server region.

A non-CICS client program uses calls to DFHXCIS to open a line to the CICS system, and then to link to a CICS program. This is called the external CICS interface (EXCI). One of the parameters of the link call is the transaction identifier under which the server program is to run. This transaction must be defined to CICS as running program DFHMIRS and as using profile DFHCICSA. Another parameter of the link call is the client's userid, which is validated if the MRO connection has been defined with ATTACHSEC(IDENTIFY).

The client program receives a USER_ERROR error if the external CICS interface command fails the security check. However, this error can have other causes; each reason code value for a USER_ERROR response indicates whether the command can be reissued directly, or whether the pipe being used has to be closed and reopened first.

The server program is executed by a mirror transaction, in a similar way to other function-shipped CICS requests. However, the transaction name associated with the mirror depends on how the program link is invoked in the client region. You must be aware of the transaction name because normal attach security applies to the mirror transaction:

- If a transaction identifier is specified on the link request, the specified transaction name is used for the mirror.
- If the transaction is omitted from the link request, but the TRANSID option is used in the program resource definition in the client region, the name for the mirror is taken from the program's TRANSID specification.
- Otherwise, the default name of CSMI is used for the mirror transaction.

You must authorize your users to access the transaction name that the mirror runs under. The userids to be authorized depend on whether LOCAL or IDENTIFY attach security is being used, and are described in "Security checking done in AOR with MRO." If the mirror transaction is defined with RESSEC(YES) in the server region, these userids must also be authorized to access the server program that is being linked to by the mirror. If the server program accesses any CICS resources, the same userids must be authorized to access them. If the server program invokes any SP-type commands, and the mirror transaction is defined with CMDSEC(YES) in the server region, the same userids must be authorized to access the commands.

If the mirror transaction cannot be attached because of security reasons, the NOTAUTH condition is not raised, but the TERMERR condition is returned to the issuing application in the client region. If the mirror transaction is successfully attached, but it is not authorized to link to the distributed program in the server region, the NOTAUTH condition is raised. The NOTAUTH condition is also raised if the server program fails to access any CICS resources for security reasons.

The server program is restricted to a DPL subset of the CICS API commands when running in a server region. The commands that are not supported include some that return security-related information. For programming information about which commands are restricted, see the *CICS/ESA Application Programming Reference* manual. For further information about DPL, refer to the *CICS/ESA Intercommunication Guide*.

Security checking done in AOR with MRO

This section summarizes how security checking is done in the AOR.

The userid of the front-end CICS region is assigned as the default. However, if a USERID is specified on the SESSIONS definition, and a link check is done, the userid actually used is the one on the SESSIONS definition.

The region userid referred to in Table 32 through Table 33 is the USERID on the SESSIONS definition. The userid referred to in this case is the one under which the job is running. This userid is the one normally returned by the security manager domain.

With ATTACHSEC(LOCAL) specified

Table 32 shows how checking is done in the AOR when ATTACHSEC(LOCAL) has been specified.

Table 32. Security checking done in AOR—ATTACHSEC(LOCAL) specified

Region userid for AOR	Userid in session definition	Region userid for TOR	Checking in AOR
USERIDA	Not specified	USERIDA	Check against AOR DFLTUSER
USERIDA	USERIDA	Anything	Check against AOR DFLTUSER
USERIDA	Not specified	USERIDB	Check against USERIDB
USERIDA	USERIDB	Anything	Check against USERIDB

With ATTACHSEC(IDENTIFY) specified

Table 33 shows how checking is done in the AOR when the ATTACHSEC(IDENTIFY) has been specified.

Table 33. Security checking done in AOR — ATTACHSEC(IDENTIFY) specified

Region userid for AOR	Userid in session definition	Region userid for TOR	Checking in AOR
USERIDA	Not specified	USERIDA	FMH-5 ATTACH check only
USERIDA	USERIDA	Anything	FMH-5 ATTACH check only
USERIDA	Not specified	USERIDB	FMH-5 ATTACH check and USERIDB
USERIDA	USERIDB	Anything	FMH-5 ATTACH check and USERIDB

Summary of resource definition options for MRO security

The following is a summary of the resource definition options you need to define for MRO security.

- On the CONNECTION definition:
  - ATTACHSEC, with either of the following options:
    - IDENTIFY
    - LOCAL
- On the SESSIONS definition:
  - USERID

For guidance on specifying CONNECTION and SESSION definitions, see the *CICS/ESA Resource Definition Guide*.

Chapter 16. Security for shared data tables

This chapter describes how to provide security for CICS shared data tables. It covers the following topics:

- "Overview"
- "Security checking" on page 206
- "LOGON security check" on page 206
- "CONNECT security checks" on page 206, (including "Bind security," and "File security").
- "RACF security-definition examples" on page 209

Overview

To provide security for a shared data table when ***cross-memory services*** are used, you must ensure that:

- The file-owning region (FOR) cannot be impersonated. You can prevent this by checking at LOGON time that the FOR is allowed to log on with the specified generic APPLID of the CICS system.
- An application-owning region (AOR) cannot gain access to data that it is not intended to access. You can prevent this by checking at CONNECT time that the AOR is allowed access to the FOR and, if file security is in force, that the AOR is allowed access to the requested file.

These security checks are performed by using the system authorization facility (SAF) to invoke RACF or an equivalent security manager.

Note: A region is still able to use data tables locally even if it does not have authority to act as a shared data table server.

The CICS shared data tables (SDT) facility reproduces the main characteristics of function-shipping security that operate at the region level, but you should note the following differences:

- SDT does not provide any mechanism for the FOR to perform security checks at the transaction level (there is no equivalent of ATTACHSEC(IDENTIFY) or ATTACHSEC(VERIFY)). Therefore, if you consider that the transaction-level checks performed by the AOR are inadequate for some files, you must ensure that those files are not associated with data tables in the FOR.
- SDT does not support any equivalent of preset security on SESSIONS, because no sessions are used.
- SDT does not pass any installation parameter list (INSTLN) information to the security user exits.

Security checking

You should consider the implications of the security checks before sharing a file that is associated with a data table.

SDT security makes use of existing CICS file security definitions, but it also relies on treating server application names (generic APPLIDs) as protected resources. An SDT server's generic APPLID is represented by a DFHAPPL.*applid* profile in the FACILITY resource class.

LOGON security check

To minimize the risk that an AOR might accept **counterfeit data records** from an FOR which is in fact an **imposter**, LOGON processing includes a security check to verify that the FOR is authorized to act as a server with the specified application name. This check is never bypassed, even when SEC=NO is specified at system initialization.

To act as a server for a protected APPLID named *applid*, a CICS region's userid must have UPDATE (or higher) access to DFHAPPL.*applid* in the FACILITY class. See Table 11 on page 109.

When a region attempts to logon as a server, SDT calls the system authorization facility (SAF) to check whether its userid has the required access authority.

If SAF neither grants nor refuses an access request: If a security profile for a specified resource is not retrieved, SAF neither grants nor refuses the access request. In this situation:

- SDT rejects the LOGON request if a security manager is installed but is either temporarily inactive or inoperative for the duration of this MVS IPL. This decision is made on the grounds that had the security manager been active it might have retrieved a profile that refuses access.
- SDT allows the LOGON to proceed if:
  - There is no security manager at all.
  - There is an active security manager but the facility class is undefined or inactive.
  - There is no profile covering the APPLID in question.

  The LOGON is allowed in these cases because there is no evidence that you want to control access to this particular APPLID.

CONNECT security checks

The security checks performed at CONNECT time provide two levels of security:

- **Bind security** allows an FOR that runs without CICS file security to be able to restrict shared access to selected AORs. (Running without file security minimizes runtime overheads and the number of security definitions.)
- **File security** can be activated in the FOR if you want SDT to implement those checks that apply to the AOR as a whole.

But note that SDT provides no way of implementing those security checks that an FOR makes at the transaction level when ATTACHSEC(IDENTIFY) or ATTACHSEC(VERIFY) is used with function shipping.

Bind security

To be allowed shared access to any of an FOR's data tables, an AOR's userid needs READ (or higher) access to the FOR's DFHAPPL.*applid* in the FACILITY class. This check is never bypassed, even when SEC=NO is specified at system initialization. Cases when SAF neither grants nor refuses access are resolved in the same way as for server LOGON (see "If SAF neither grants nor refuses an access request" on page 206). If the result is a refusal, shared access by the AOR to this APPLID is not permitted.

Note that controlling LOGON and bind by using different (but hierarchical) levels of access to the same resource has the following consequences:

- Any region with the same userid as a server can always bind to that server.
- It is impossible to control which userids can bind to a given APPLID without also controlling which userids can log on as a server for that APPLID.

SDT bind-time security uses different definitions from those employed by ISC and (if using preset sessions) MRO. So, unless you make them consistent, SDT access might be granted when function shipping attempts are rejected, or vice versa. Both MRO and SDT use the same class and so, with ISC only, SDT CONNECT security might react to changes in security definitions either earlier or later than function shipping.

If file security is not in force in the FOR (that is, if SEC=NO or XFCT=NO was specified at system initialization), an AOR that is allowed to bind to an FOR is also allowed to access all that FOR's shared data tables.

If file security is in force, an AOR that is allowed to bind is still allowed free access if the userids of the AOR and FOR are the same (undefined userids are not considered to be the same).

File security

When file security is in force in an FOR, and the userid of the AOR is undefined, a CONNECT request fails unless the FOR's default userid (specified by the DFLTUSER system initialization parameter) is allowed to read the specified file.

When file security is in force in an FOR, and the userid of the AOR is known but is different from that of the FOR, SDT first checks whether the AOR's userid is allowed to sign on to the FOR's application (that is, whether its userid has READ access to entity APPLID in the APPL class). Cases when SAF neither grants nor refuses the request are resolved in the same way as for server LOGON (see "If SAF neither grants nor refuses an access request" on page 206).

If the userid is allowed to sign on to the FOR's application, the CONNECT request succeeds unless the AOR's userid is not allowed to read the specified file. Otherwise, the CONNECT request is treated in the same way as when the AOR's userid is undefined.

Function shipping detects that an AOR's access to a file has been revoked when a rebuild of the file control resource class is completed in the FOR. However, if a valid connection already exists, SDT continues to allow access until something causes the connection to be broken. See "Refreshing resource profiles in main storage" on page 29, and the information appropriate to RACF 1.9 and RACF 2.1.

Warning: If you use ISC instead of MRO for function shipping, you must ensure that the value of the SECURITYNAME parameter in the FOR is the same as the userid of the AOR. Otherwise, the SDT CONNECT and function shipping security checks will be inconsistent.

Use of RACF group classes for file security definitions

If you use RACF group classes for any of your file security definitions, you might need to take some additional steps to ensure that SDT enforces your security policy correctly. This is because RACF security checks ignore profiles defined in a group class unless a RACLIST operation has been performed for the corresponding member class.

When function shipping is used, profiles in group classes are always examined because the FOR carries out the necessary RACLIST processing during initialization. However, these in-storage profiles are not accessible during SDT CONNECT processing because they are stored in the FOR's address space. Therefore, you must use the TSO command:

```
SETROPTS RACLIST(resource class)
```

to ensure that each FOR's file security definitions are accessible to all address spaces in the MVS system. (This might not be necessary if you know that every AOR is enforcing file security using the same resource class as its FORs.)

The SETROPTS RACLIST command for a specific resource class is valid only if the definition in the class descriptor table allows it. Therefore, if you use installation-defined resource classes for CICS file security, you must change the definition of each *member* class (not that of its related group class) by specifying RACLIST=ALLOWED on the ICHERCDE macro.

If you are using RACF 1.9, you can change the class descriptor table for the default classes FCICSFCT and HCICSFCT by installing PTF UY77225. Also for RACF 1.9, you are recommended to install the PTF associated with APAR OY51791 to ensure that SETROPTS RACLIST does not cause more profiles than necessary to be brought into storage.

If you share the RACF database between MVS systems, you should consider whether you need to install PTF UY75680.

To activate any updates to RACF information that affects CICS file security, you need to issue the TSO command:

```
SETROPTS RACLIST(resource-class) REFRESH
```

as well as issuing the CEMT PERFORM SECURITY REBUILD command in all CICS regions that use the resource class in question.

RACF remembers SETROPTS RACLIST commands and reinstates the previous environment automatically following an IPL. Also, when two or more MVS systems share the RACF database, the effects of a SETROPTS RACLIST command that is issued on one system are propagated to the others when they are IPLed.

RACF security-definition examples

For RACF, you protect an application by using RDEFINE and PERMIT commands to define suitable profiles and access lists for the application (DFHAPPL.*applid*) in the FACILITY class.

Example 1: The following RACF definitions specify that:

- Only USERA can act as the SDT FOR for application CICSA
- Only USERA, USERB, and USERC can have shared access to CICSA's data tables.

```
RDEFINE FACILITY (DFHAPPL.CICSA) UACC(NONE)

PERMIT DFHAPPL.CICSA CLASS(FACILITY) ID(USERA) ACCESS(UPDATE)

PERMIT DFHAPPL.CICSA CLASS(FACILITY) ID(USERB USERC) ACCESS(READ)
```

Example 2: The following RACF definitions specify that:

- Only users in group GRPTEST can act as SDT FORs for applications that have names beginning with CICSTST
- Any user can have shared access to the FOR's data tables (subject to FOR file security).

```
RDEFINE FACILITY (DFHAPPL.CICSTST*) UACC(READ)

PERMIT DFHAPPL.CICSTST* CLASS(FACILITY) ID(GRPTEST) ACCESS(UPDATE)
```

Part 4. Customization

Part 4, “Customization” gives an overview of customizing the CICS-ESM interface, including:

Chapter 17. Customizing security processing

This chapter contains Product-sensitive Programming Interface and Associated Guidance information.

This chapter introduces you to the CICS-RACF interface, and describes how the MVS router passes control to RACF. It describes how RACF exit programs can access CICS-related information. Finally, it lists the control points at which CICS invokes the ESM. The chapter is organized as follows:

For programming information on customizing the CICS-ESM interface (using either RACF or a compatible user-written or vendor-supplied ESM), see the *CICS/ESA Customization Guide*.

Overview of the CICS-RACF interface

In CICS/ESA 4.1, the only form of security CICS supports is that provided by an external security manager, such as RACF. CICS uses, by means of the RACROUTE macro, the MVS system authorization facility (SAF) interface to route authorization requests to RACF.

As shown in Figure 29, the RACROUTE macro invokes the MVS router, which invokes the RACF router, which calls the ESM (in this case RACF).

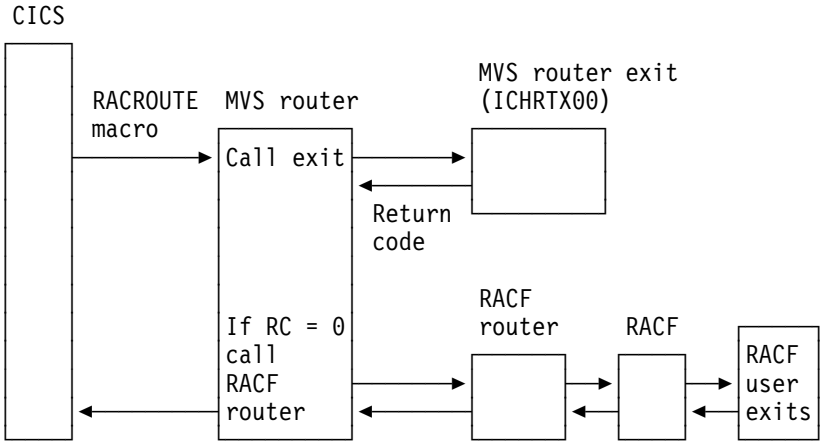

Figure 29. MVS router exit

| See the *RACROUTE Macro Reference* manual for information on how the RACROUTE macro is coded.

The control points at which CICS issues a RACROUTE macro to route authorization requests are described in “CICS security control points” on page 216.

MVS router

The system authorization facility (SAF) provides your installation with centralized control over security processing by using a system service called the ***MVS router***. The MVS router provides a common system interface for all products providing and requesting resource control. The resource-managing components and subsystems (such as CICS) call the MVS router as part of certain decision-making functions in their processing, such as access control checking and authorization-related checking. These functions are called **control points**. This single SAF interface encourages the use of common control functions shared across products and across systems.

If RACF is available in the system, the MVS router may pass control to the RACF router, which in turn invokes the appropriate RACF function. (The parameter information and the RACF router table, which associates router invocations with RACF functions, determine the appropriate function.) However, before calling the RACF router, the MVS router calls an optional, installation-supplied security-processing exit, if one has been installed.

The system authorization facility and the SAF router are present on all MVS systems, even if RACF is not installed. Although the SAF router is not part of RACF, many system components and programs, such as CICS, invoke RACF through the RACROUTE macro and SAF. Therefore, installations can modify RACF parameter lists and do customized security processing within the SAF router. For information about how to code a SAF router exit, see the *External Security Interface (RACROUTE) Macro Reference for MVS and VM* manual.

How ESM exit programs access CICS-related information

When CICS invokes the ESM, it passes information about the current CICS environment, for use by an ESM exit program, in an **installation data parameter list**. How your exit programs access the installation data parameter list depends on the ESM you are using. The ICHxxxxx interfaces defined in Table 34 on page 215 below apply only to RACF. For programming information on non-RACF interfaces, see the *CICS/ESA Customization Guide*.

RACF user exit parameter list

If you write RACF user exits, you can find the address of the installation data parameter list directly from the RACF user exit parameter list. The name of the relevant field in the user exit parameter list varies according to the RACROUTE REQUEST type and the RACF user exit that is invoked. The relationships between REQUEST type, exit name, and field name are shown in Table 34 on page 215.

Table 34. Obtaining the address of the installation data parameter list			
RACROUTE REQUEST type	**RACF exit**	**Exit list mapping macro**	**Parameter list field name. (See Notes 1 and 2.)**
VERIFY	ICHRIX01	ICHRIXP	RIXINSTL
	ICHRIX02	ICHRIXP	RIXINSTL
AUTH	ICHRCX01	ICHRCXP	RCXINSTL
	ICHRCX02	ICHRCXP	RCXINSTL
FASTAUTH	ICHRFX01	ICHRFXP	RFXANSTL
	ICHRFX02	ICHRFXP	RFXANSTL
LIST	ICHRLX01	ICHRLX1P	RLX1INST
	ICHRLX02	ICHRLX2P	RLX2PRPA

Notes:

1. The 'xxxINSTL' field points to the installation parameter list only if you code ESMEXITS=INSTLN in the CICS system initialization parameters. The default value for this parameter is NOINSTLN, which means that no installation data is passed. (Note that ESMEXITS cannot be coded as a SIT override.)
2. RLX2PRPA contains the address of the ICHRLX01 user exit parameter list (RLX1P). Field RLX1INST of RLX1P points to the installation data parameter list.
3. There is no RACF user exit for REQUEST=EXTRACT, and no installation parameter data is passed. Any customization must be done using the MVS router exit, ICHRTX00.

For brief descriptions of RACF exits and their functions, see the *RACF Security Administrator's Guide*. For full descriptions of the RACF exit parameter lists, see the *System Programming Library: RACF* manual.

Installation data parameter list

The installation data parameter list gives your ESM exit programs access to the following information:

- CICS security event being processed
- Details of the current CICS environment, as available. That is:
  - APPLID of the CICS region
  - Common work area
  - Transaction being invoked
  - Program being executed
  - CICS terminal identifier
  - VTAM LUname
  - Terminal user area
  - An 8-byte communication area, whose usage is described in the *CICS/ESA Customization Guide*.

For programming information about user-written ESMs, see the *CICS/ESA Customization Guide*.

CICS security control points

The following list summarizes the RACROUTE macros used by CICS to invoke the ESM, and the control points at which they are issued.

Some of these calls may not always be issued, because CICS reuses entries for users already signed on.

RACROUTE
This is the “front end” to the macros described below, it invokes the MVS router.

RACROUTE REQUEST=VERIFY
This macro is issued at operator signon (with the parameter ENVIR=CREATE), and at signoff (with the parameter ENVIR=DELETE). This macro creates or destroys an ACEE (access control environment element). It is issued at the following CICS control points. It is also issued (with the parameter ENVIR=VERIFY) early in normal signon through EXEC CICS SIGNON, but this call is ignored by RACF.

Each of the following control points relates to ENVIR=CREATE.

- Normal signon through EXEC CICS SIGNON
- Signon of the default userid DFLTUSER
- Signon of preset security terminals
- Signon of MRO sessions
- Signon of LU6.1 sessions
- Signon of LU6.2 sessions
- Signon for XRF tracking of any of the above
- Signon associated with the userid on attach requests (for all operands of ATTACHSEC except LOCAL).

Each of the following control points relates to ENVIR=DELETE:

- Normal sign-off through EXEC CICS SIGNOFF
- Signoff when deleting a terminal
- Signoff when TIMEOUT expires
- Signoff when USRDELAY expires
- Signoff of MRO sessions
- Signoff of LU6.1 sessions
- Signoff of LU6.2 sessions
- Signoff for XRF tracking of any of the above.
- Sign-off associated with the userid on attach requests (for all operands of ATTACHSEC except LOCAL)

RACROUTE REQUEST=VERIFYX
This macro creates and deletes an ACEE in a single call. It is issued at the following control points:

- Signon, as an alternative to VERIFY, when an optimized signon is performed for subsequent attach sign-ons across an LU6.2 link with ATTACHSEC(VERIFY) or ATTACHSEC(PERSISTENT).
- When an invalid password or PassTicket is presented, or EXEC CICS VERIFY PASSWORD is issued.

RACROUTE REQUEST=FASTAUTH

This macro is issued during resource checking, on behalf of a user who is identified by an ACEE. It is the high-performance form of REQUEST=AUTH, using in-storage resource profiles, which does not cause auditing to be performed. It is issued at the following CICS control points:

- When attaching local transactions
- When checking link security for transaction attach
- Transaction validation for MRO tasks
- CICS resource checking
- Link security check for a CICS resource
- Transaction validation for EDF
- Transaction validation for the transaction being tested (by EDF)
- DBCTL PSB scheduling resource security check
- DBCTL PSB scheduling link security check
- Local DL/I PSB scheduling resource check
- Remote DL/I PSB scheduling resource check
- When checking surrogate user authorities
- QUERY SECURITY with the RESTYPE option.

RACROUTE REQUEST=AUTH

This macro provides a form of resource checking with a larger pathlength, and causes auditing to be performed. It is used as follows:

- After a call to FASTAUTH indicates an access failure that requires logging.
- When a QUERY SECURITY request with the RESCLASS option is used. This indicates a request for a resource for which CICS has not built in-storage profiles.

RACROUTE REQUEST=LIST

This macro is issued to create and delete the in-storage profile lists needed by REQUEST=FASTAUTH. (One REQUEST=LIST macro is required for each resource class.) It is issued at the following CICS control points:

- When CICS security is being initialized
- When an EXEC CICS PERFORM SECURITY REBUILD command is issued
- When XRF tracks either of these events.

RACROUTE REQUEST=EXTRACT

This macros is issued (with the parameters SEGMENT=CICS,CLASS=USER, with the parameters and with the SEGMENT=BASE,CLASS=USER to obtain the national language and user name) at all the following control points:

- Normal signon through EXEC CICS SIGNON
- Signon of the default userid DFLTUSER
- Signon of preset security terminals
- Signon of MRO sessions
- Signon of LU6.1 sessions
- Signon of LU6.2 sessions
- Signon for XRF tracking of any of the above
- Signon associated with the userid on attach requests (for all operands of ATTACHSEC except LOCAL).

It can be used to verify the user's password when an entry in the user table is reused within the USRDELAY period.

It is also issued (with the parameters SEGMENT=SESSION,CLASS=APPCLU) during verification of LU6.2 bind security, at the following CICS control point:

- Bind of LU6.2 sessions.

Note: There is no RACF user exit for REQUEST=EXTRACT and no installation parameter data is passed. Any customization must be done using the MVS router exit, ICHRTX00. For a detailed description of these macros, see the *RACROUTE Macro Reference* manual.

Determining the userid of the CICS region

CICS makes use of the userid of the region in which it runs for the following purposes:

- To prefix resource names if SECPRFX=YES is specified. For more information about the SECPRFX system initialization parameter see "SECPRFX" on page 54.
- As the user to be checked for category 1 transactions. For more information see "Category 1 transactions" on page 126.
- As the default PLTPI user for PLTPI non-terminal security, if a PLTPIUSR is not specified in the system initialization parameter.
- For SURROGAT checking (for example, authority to use the PLTPI and default userids).

CICS obtains the region userid by invoking the external security manager, which extracts it from the relevant RACF control blocks for the job. The security domain obtains the region userid for the above functions by issuing a RACROUTE REQUEST=EXTRACT macro. To customize the response from this macro, and thus the security identification of a CICS region, you must use the MVS security router exit, ICHRTX00.

For MRO logon and bind security, the region userid is obtained directly from the appropriate accessor environment element (ACEE), and cannot be customized. For more information see Chapter 15, "Implementing MRO security" on page 189.

Specifying user-defined resources to RACF

If you want to use the QUERY SECURITY command with the RESCLASS option, you may need to create user-defined resources within user-defined classes to represent the non-CICS resources that you want to query. To do this, you must add entries to the RACF class descriptor table (CDT) and to the RACF router table. Then, you must activate the new classes, define your resources in the new classes, and finally grant your users access to the resources. To improve the performance of QUERY SECURITY, you should also consider loading the new resource profiles into virtual storage.

Adding new resource classes to the class descriptor table

The RACF class descriptor table has a system-defined part, and an installation-defined part named ICHRRCDE. You add new resource classes to ICHRRCDE by coding the ICHERCDE macro. For example, to add the new class $FILEREC to the CDT, and optionally a corresponding group class $GILEREC, add the following macros to ICHRRCDE:

```
$FILEREC ICHERCDE CLASS=$FILEREC,      Entity or Member class          *
               GROUP=$GILEREC,                                         *
               ID=192,                                                 *
               MAXLNTH=17,                                             *
               RACLIST=ALLOWED,                                        *
               FIRST=ALPHANUM,                                         *
               OTHER=ANY,                                              *
               POSIT=42,                                               *
               OPER=NO,                                                *
               DFTUACC=NONE
$GILEREC ICHERCDE CLASS=$GILEREC,      Group class                     *
               MEMBER=$FILEREC,                                        *
               ID=191,                                                 *
               MAXLNTH=17,                                             *
               FIRST=ALPHANUM,                                         *
               OTHER=ANY,                                              *
               POSIT=42,                                               *
               OPER=NO,                                                *
               DFTUACC=NONE
```

The same classes must also be added to the RACF router table, ICHRFR01, by coding the ICHRFRTB macro:

```
ICHRFRTB CLASS=$FILEREC,ACTION=RACF
ICHRFRTB CLASS=$GILEREC,ACTION=RACF
```

Both the ICHERCDE and ICHRFRTB macros are described in the *RACF Macros and Interfaces* manual.

When you have recreated the two modules ICHRRCDE and ICHRFR01, you must re-IPL your MVS system to bring them into use.

Activating the user-defined resource classes

Once the new classes have been installed in the system, it is necessary to activate them in RACF before they can be used. This has to be done by a user with system-SPECIAL authority, who enters the following commands under TSO:

```
SETROPTS CLASSACT($FILEREC)
SETROPTS GENERIC($FILEREC)
```

To improve the performance of QUERY SECURITY, you should load the new resource profiles into virtual storage by using the RACLIST option. The RACLIST option is *required* if you are using the group class, because the connection between the group class and the entity class is resolved by RACLIST:

```
SETROPTS RACLIST($FILEREC)
```

The SETROPTS commands only need to be issued for the entity class $FILEREC, because the group class $GILEREC has the same POSIT number.

Defining resources within the new class

Resources within the new classes have to be defined by a user with system-SPECIAL authority, or with CLAUTH authority in the new class. CLAUTH authority is granted by issuing the following TSO command:

```
ALTUSER userid CLAUTH($FILEREC)
```

If you have the required authority, you can create the new resources by issuing the following TSO commands:

```
RDEFINE $FILEREC PAYFILE.SALARY   UACC(NONE)
RDEFINE $FILEREC PAYFILE.TAXBAND  UACC(NONE)
RDEFINE $GILEREC PERSONAL.DETAILS ADDMEM( PERSONAL.DEPT,     +
                                          PERSONAL.MANAGER,  +
                                          PERSONAL.PHONE)    +
                                  UACC(READ)
```

Now you are ready to authorize users to use the new resources. Assume that PAYROLL is the name of a group of users who are to be permitted to update all the pay and personal details fields in an employee record. The following TSO commands grant UPDATE access to all users in the group:

```
PERMIT PAYFILE.SALARY   CLASS($FILEREC) ID(PAYROLL) ACCESS(UPDATE)
PERMIT PAYFILE.TAXBAND  CLASS($FILEREC) ID(PAYROLL) ACCESS(UPDATE)
PERMIT PERSONAL.DETAILS CLASS($FILEREC) ID(PAYROLL) ACCESS(UPDATE)
```

If you had previously loaded the profiles by using the RACLIST option, you should now refresh the profiles in virtual storage by issuing the command:

```
SETROPTS RACLIST($FILEREC) REFRESH
```

Designing applications to use the user-defined resources

This section describes an example of how you might design applications to make use of the user-defined resources.

Your applications use CICS file control in the normal way to read records from the pay and personal details file. Because you are controlling individual fields within each record, you may not need to apply resource security at the file level, so your transactions can be defined with RESSEC(NO). After reading the file record, but before displaying the results, you use QUERY SECURITY to determine whether the user has the authority to access the particular field within the record. For instance, before displaying the salary amount, you issue:

```
EXEC CICS QUERY SECURITY RESCLASS('$FILEREC')
                         RESID('PAYFILE.SALARY')
                         RESIDLENGTH(14)
                         READ(read_cvda)
```

Then, depending on the value returned in `read_cvda`, your application either displays the salary or a message stating that the user is not authorized to display it. Likewise, as part of a transaction that updates a person's telephone number, you issue:

```
EXEC CICS QUERY SECURITY RESCLASS('$FILEREC')
                         RESID('PERSONAL.PHONE')
                         RESIDLENGTH(14)
                         UPDATE(update_cvda)
```

If the value returned in `update_cvda` indicates that the user has UPDATE access, the transaction can continue and update the telephone number in the file. Otherwise, it should indicate that the user is not authorized to update the telephone number.

How to bypass attach checks for non-terminal transactions

CICS always performs a transaction-attach security check for each transaction attach, even when the transaction has no associated terminal. Although this generally gives greater control over who can initiate transactions, it is different from the behavior of releases of CICS before CICS/ESA 4.1. The following suggests how you can bypass transaction-attach security checks for non-terminal transactions while continuing to keep full transaction-attach security for terminal-attached transactions.

CICS always performs the transaction-attach resource check using `RACROUTE REQUEST=FASTAUTH`, so you only need to provide an ICHRFX01 user exit. The ICHRFX01 routine must issue a zero return code to indicate that the resource check processing is to continue, or a return code of 8 to indicate that the check is to be regarded as successful.

So that the ICHRFX01 exit can determine the circumstances under which it is called, you must specify ESMEXITS=INSTLN in the SIT for the CICS regions for which you want to control transaction-attach security. Then your ICHRFX01 routine should do the following:

1. Obtain the address of the CICS installation data parameter list, as described in “How ESM exit programs access CICS-related information” on page 214. If this address is zero, either the caller of the RACROUTE macro is not CICS or it is a CICS region whose behavior you do not wish to modify, so exit with a return code of zero.
2. Use the DFHXSUXP macro to map the fields in the installation data parameter list.
3. Confirm that the installation data was created by CICS by checking that UXPDFHXS is equal to ‘DFHXS’. If it is not equal to ‘DFHXS’, exit with a return code of zero.
4. Examine field UXPPHASE in the installation data. If it is not equal to USER_ATTACH_CHECK (X'40'), this is not a transaction attach, so exit with a return code of zero.
5. Examine field UXPTERM in the installation data. If it is nonzero, this is a terminal-related transaction attach, so exit with a return code of zero.
6. If UXPPHASE is USER_ATTACH_CHECK and UXPTERM is zero, then a non-terminal transaction is being attached. Exit with a return code of 8 to indicate to RACF that this check is successful. The function RACROUTE REQUEST=FASTAUTH then completes with a return code of zero, and CICS continues with the attach of the non-terminal transaction.

Global user exits in signon and signoff

CICS provides the XSNON global user exit in EXEC CICS SIGNON processing and the XSNOFF global user exit in EXEC CICS SIGNOFF processing. These exits do not allow you to affect the result of the signon or signoff, but notify you when the userid associated with a terminal changes. The exits are further described in the *CICS/ESA Customization Guide*.

Part 5. Migration and coexistence

This part describes the security implications in migrating from earlier releases of CICS/ESA to CICS/ESA 4.1. It includes two chapters:

- Chapter 18, “Migration considerations” on page 225 describes the migration implications of certain security-related features introduced in CICS/ESA 4.1 and earlier releases. It also covers the mixing of internal and external security in an MRO environment, the use of preset security terminals, and CESN.
- Chapter 19, “Coexistence with previous CICS releases” on page 235 describes various aspects of coexistence from a security viewpoint, including MRO, SIT parameters, transaction resource definitions, and timeout values.

Chapter 18. Migration considerations

This chapter describes several considerations for migrating from earlier releases of CICS to CICS/ESA 4.1. Note that CICS/ESA 3.3 security is essentially the same as CICS/ESA 3.2.1 security.

Introduction of UPDATE access authority in CICS/ESA 3.1.1

In CICS releases before CICS/ESA Version 3, the only access authority recognized by CICS is READ. These CICS releases do not distinguish levels of access authority (for example, between READ and UPDATE authority).

CICS/ESA Version 4 always differentiates between requests to read data from those that attempt to update data. CICS/ESA Version 4 conveys the application program's intent when issuing RACROUTE authorization requests, so that RACF can provide the required response to either type of access intent for any particular terminal user. In CICS releases earlier than 3.1.1. a user with READ access to a CICS filename is allowed to perform update as well as read activity to the designated file. The authorization is essentially only a "use" or "not use" distinction.

You can alter existing PERMIT commands to specify ACCESS(UPDATE) as appropriate. When these altered profiles are used with earlier CICS releases, access is granted when either READ or UPDATE authority is in effect for access to the resource described by the profile. This enhancement, introduced in CICS/ESA 3.1.1, to correctly map the access intent within the CICS application to the RACROUTE REQUEST=FASTAUTH request issued by CICS, is not dependent on any RACF release. The enhancement is in effect when CICS/ESA 3.1.1 is running with EXTSEC=(YES,,UPDATE) specified as a system initialization parameter.

In CICS/ESA 3.2.1 and CICS/ESA 3.3, specifying SEC=MIGRATE causes CICS to ignore any distinction between READ and UPDATE access intent. In CICS/ESA 4.1, SEC=MIGRATE is no longer supported. For more information, see "SEC" on page 54.

For releases before CICS/ESA 3.1.1, there are alternatives that can provide the level of application control required even though CICS does not distinguish READ from UPDATE. If an update function within an application is performed either by a program or under a transaction code that is not required for the read-only function,

program security or transaction security can be used to determine whether a given terminal user can perform an update function in the earlier CICS releases.

One other possibility that involves modification of the application, but one that is potentially very simple, is to define an alias transaction code to be used when the update function is to be performed (assuming that a single program performs both the inquiry and update processing). The only change required within the application program itself is to add a test before performing an update to ensure that EIBTRNID contains the transaction code intended to permit updating. When the applications are driven through menu panels it is frequently possible to make the introduction of this alias transaction code completely transparent to the users of the application.

Removal of internal security in CICS/ESA 3.2.1

In CICS releases before CICS/ESA 3.2.1, for any resource type for which RACF authorization was not requested (for example, XFCT=NO in the system initialization parameters), CICS reverted to the CICS internal resource security level checking mechanism, as though RSLC=YES had been specified for the transaction.

This mechanism is based on the RSLKEY values defined for the terminal user in the sign-on table and the RSL value specified in the relevant resource control table entry. Unless the CICS resource definition, for a filename for example, has RSL(PUBLIC) specified or the RSL(nn) matches one of the RSLKEYs associated with the signed-on terminal user, then the NOTAUTH condition is raised following the execution of an EXEC CICS READ FILE(filename) ... command. For all resource definitions for which the X*name* class is NO, and which may be accessed by any transaction for which RSLC=EXTERNAL is specified, you must specify either RSL=PUBLIC or RSL=*n*, where *n* is a security key other than 0.

Removal of internal LU6.2 bind time security

The BINDPASSWORD in a CSD CONNECTION definition is not used for LU6.2 bind time security validation. Instead, you should create RACF APPCLU profiles, and specify XAPPC=YES on the SIT to maintain validated links.

Use of CICS segment in RACF user profiles in CICS/ESA 4.1

If you are migrating to CICS/ESA 4.1 from a release of CICS before CICS/ESA 3.2.1, you can use the DFHSNMIG migrate utility to migrate your existing signon table (SNT) to the CICS segment of RACF user profiles. See “Signon table migration utility” on page 227 and the *CICS/ESA Operations and Utilities Guide* for information about the DFHSNMIG utility.

The CICS segment of the user profile contains data for CICS terminal users. In earlier releases of CICS, you provided this information in a CICS SNT. For information on the order in which CICS searches for the operator information, see “Obtaining CICS-related data for the default user” on page 72.

Signon table migration utility

The sign-on table migration utility, DFHSNMIG, is provided to help you migrate CICS terminal-user data from an SNT to the CICS segment of a RACF user's profile. The utility reads the SNT, and for each user entry in the SNT it creates a CLIST of RACF commands, generating either an ADDUSER or ALTUSER command as appropriate for each SNT user entry. Because the DFHSNT macro is no longer supplied, the SNTs must be assembled using the pre-CICS 4.1 DFHSNT macro.

DFHSNMIG can be found as an APF-authorized program in CICS410.SDFHAUTH, and must be run from an APF-authorized library. If you invoke the program from TSO, add its name to the list of authorized program names in the AUTHPGM NAMES section in the IKJTSOxx member of SYS1.PARMLIB.

The DFHSNMIG utility creates a CLIST of ADDUSER and ALTUSER commands to define CICS users to RACF. These commands do not specify the default RACF group each user should belong to. You might want to edit the CLIST created by DFHSNMIG to add DFLTGRP information. See "Defining terminal users and user groups to RACF" on page 73 for an example of specifying DFLTGRP on the ADDUSER command.

Figure 30 shows an example signon table entry. In this example, OLDUSER is an existing RACF-defined userid, and NEWUSER is a userid that has not previously been defined to RACF. DFHSNT TYPE=(ENTRY,DEFAULT) is a default entry, for which DFHSNMIG will *not* create an entry.

```
SNT     DFHSNT TYPE=INITIAL
*
        DFHSNT TYPE=ENTRY,                                              *
               USERID=OLDUSER,                                          *
               OPIDENT=OLD,                                             *
               OPPRTY=255,                                              *
               OPCLASS=(1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,   *
               19,20,21,22,23,24),                                      *
               NATLANG=K,                                               *
               XRFSOFF=FORCE
*
        DFHSNT TYPE=ENTRY,                                              *
               USERID=NEWUSER,                                          *
               OPIDENT=NEW,                                             *
               OPPRTY=100,                                              *
               TIMEOUT=20,                                              *
               OPCLASS=(10)
*
        DFHSNT TYPE=(ENTRY,DEFAULT),                                    *
               OPIDENT=XXX,                                             *
               TIMEOUT=10
*
        DFHSNT TYPE=FINAL
         END
```

Figure 30. Example sign-on table entry

Figure 31 shows an example of output from DFHSNMIG, which has changed the SNT shown in Figure 30 on page 228 into entries for the RACF database. For more information about running DFHSNMIG, see the *CICS/ESA Operations and Utilities Guide*.

```
/*-------------------------------------------------------*/
/*                                                       */
/* Migration of DFHSNT. (Created by DFHSNMIG utility.)   */
/* This CLIST will add CICS attributes into your RACF    */
/* database.  Please note that keywords are for RACF 1.9 */
/* and will not work against earlier versions of RACF.   */
/*                                                       */
/* You may need to edit this file before executing the   */
/* CLIST under a TSO userid that has SPECIAL authority.  */
/*                                                       */
/* ADDUSER: Asks RACF to create a new entry for the user.*/
/* ALTUSER: Adds CICS attributes to an existing RACF user.*/
/*                                                       */
/* Userid   - Identifier for user.                       */
/* LANGUAGE - Preferred language:  ENU = English (US)    */
/*                                 JPN = Japanese        */
/* The CICS attributes are:                              */
/* OPCLASS  - Operator Class                             */
/* OPIDENT  - Operator Identifier                        */
/* OPPRTY   - Operator Priority                          */
/* TIMEOUT  - Timeout Value                              */
/* XRFSOFF  - FORCE or NOFORCE                           */
/*                                                       */
/*-------------------------------------------------------*/
/*-------------------------------------------------------*/
/* Details for                                 OLDUSER   */
/*-------------------------------------------------------*/
ALTUSER OLDUSER                                          +
  LANGUAGE(PRIMARY(JPN))                                 +
  CICS(                                                  +
     OPCLASS(1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,  +
      19,20,21,22,23,24)                                 +
     OPIDENT(OLD)                                        +
     OPPRTY(255)                                         +
     TIMEOUT(0)                                          +
     XRFSOFF(FORCE)                                      +
  )
/*-------------------------------------------------------*/
/* Details for                                 NEWUSER   */
/*-------------------------------------------------------*/
ADDUSER NEWUSER                                          +
  CICS(                                                  +
     OPCLASS(1,10)                                       +
     OPIDENT(NEW)                                        +
     OPPRTY(100)                                         +
     TIMEOUT(20)                                         +
     XRFSOFF(NOFORCE)                                    +
  )
/*-------------------------------------------------------*/
/* 00000002 entries successfully processed.              */
/*-------------------------------------------------------*/
```

Figure 31. Example of output from DFHSNMIG

Goodnight transaction

By specifying your own GNTRAN transaction you can use the CICS API to control the TIMEOUT operation. For example, your transaction could display a screen that prompts for the password. Specifying the EXEC CICS ASSIGN USERID request obtains the userid, and EXEC CICS VERIFY PASSWORD( ) USERID ( ) would validate the input. Based on the response, the user could remain signed on or be signed off.

By default CICS uses the CESF transaction to signoff a user terminal. The goodnight transaction is not available for a surrogate terminal that is timed out during a CRTE session. Signoff occurs with a loss of the security capabilities the terminal previously had, leaving a DFHSN1200 message in the log.

APAR PQ08546

13/10/97 MJO

GNTRAN=NO must be used if TCAM terminal users have a non-zero TIMEOUT value specified in the CICS segment of their RACF profiles.
GNTRAN=<*transaction_id*> is not supported for TCAM terminal users subject to TIMEOUT.

Migrating to RACF on releases pre-CICS/ESA 3.2.1

In preparation for migrating to RACF on CICS/ESA 3.2.1 or later releases, which do not support the old CICS internal security mechanism, you should consider migrating to RACF on your existing release. If you do this, there are some factors you should consider. For example:

- Can you convert a region in its entirety to RACF or because of the size of the conversion task, will you have to convert regions progressively, and run with some internal and some external security?
- Are you using MRO? If so, you can mix internal and external security while you gradually complete the migration process.

Internal and external security can be mixed in the same region, but there are some pitfalls to be avoided. One “mix” you must avoid is trying to use the CICS-supplied CEDF transaction specified with a different form of security from the security of a transaction running under CEDF. For example, if you try and run CEDF with external security to debug transactions protected with CICS internal security, CEDF abends with abend code AED3.

Mixing internal and external security in an MRO environment

If you are migrating to external (RACF) security gradually, perhaps on an application-by-application basis, and you are operating CICS with MRO, you can mix internal and external security. However, you must ensure that you make the changes in a way that allows the old internal security to work as before, as well as enabling your new RACF security to work as planned. Consider the example illustrated in the figures that follow.

- In Figure 32 on page 231, CICS uses internal security in both the TOR and the AOR regions.

- In Figure 33 on page 232, CICS uses RACF in both the TOR and the AOR regions.

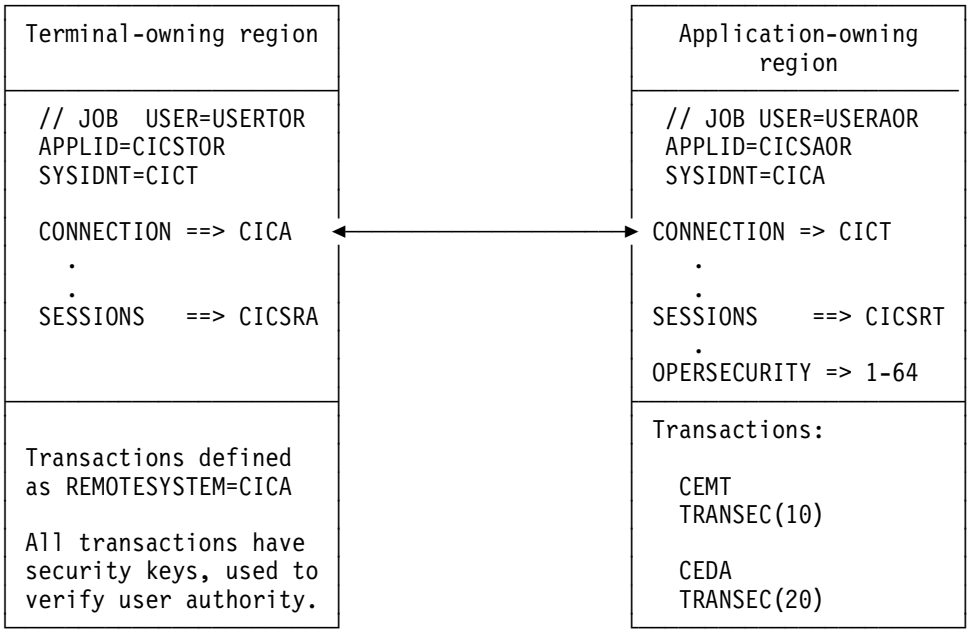

Figure 32. Simple MRO configuration with TOR and AOR using CICS internal security

Notes:

1. The SESSIONS definition installed in CICA specifies operator security keys, indicating preset security on the link between the TOR and AOR.
2. With OPERSECURITY specified, CICS does not sign on the link for the purpose of checking the TOR's authorization to route and attach transactions in the AOR. With preset security, CICS checks the transaction security keys of transactions routed from the TOR to determine whether they can be attached. In this example, all transactions can be attached because the full range of security keys is specified.
3. In this example, security checking on terminal users is done in the TOR.

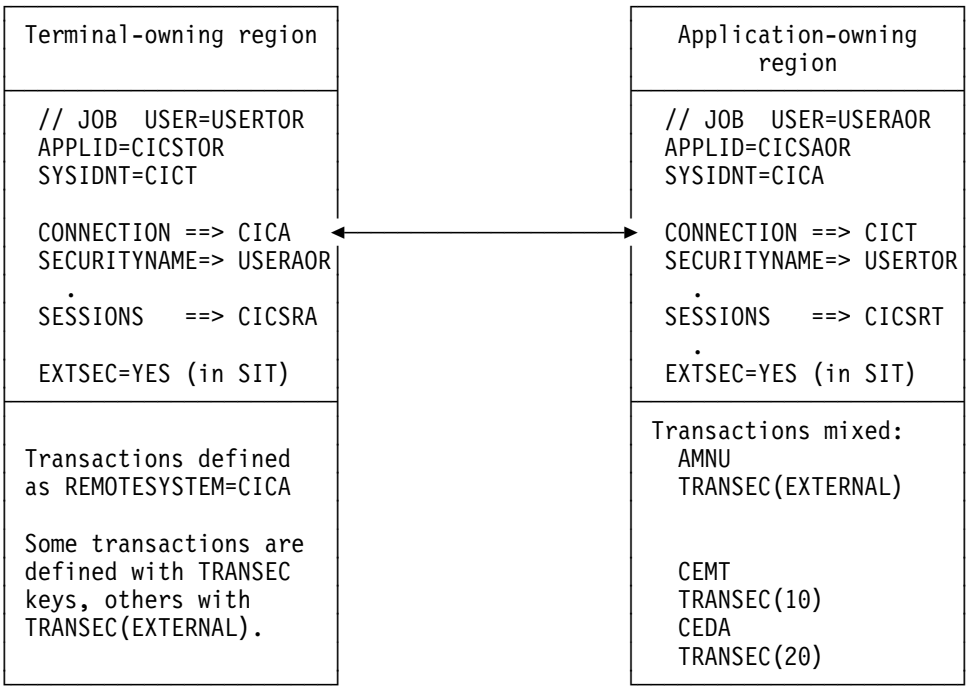

Figure 33. The MRO regions now using both internal and external security

Note: The SESSIONS definition installed in CICA no longer specifies security keys, so there is no preset security on the link between the TOR and AOR. (Preset security with an external security manager is not supported in releases of CICS earlier than CICS/ESA 3.2.1.)

Installing preset security terminals

In releases before CICS/ESA 4.1, the authority to install terminals with preset userids was ALTER authority to the TERMINAL resource in the CCICSCMD resource class. In Release 4.1, you must change these RACF definitions to use READ authority to the *userid*.DFHINSTL resource in the SURROGAT resource class. To migrate any definitions before Release 4.1, see the *CICS/ESA Migration Guide*.

To define a surrogate user to RACF:

1. Define a resource in the SURROGAT resource class for each transaction user. The name of the resource is:
   - *userid*.DFHSTART for authority to start a transaction with the START command.
   - *userid*.DFHINSTL for authority to:
     - Install terminals with a preset userid
     - Start TD trigger-level transactions
     - Run PLTPI transactions at CICS startup.
   - Adding the appropriate surrogate users to the access lists for those resources, with READ authority.

For more information about surrogate users defined to RACF see Chapter 7, “Surrogate user security” on page 103.

Signing off with CESN

In CICS/ESA 4.1 there is a change to the way the CESN transaction works. Users who are accustomed to using CESN to sign off should be advised that CICS/ESA 4.1 does not always sign off a user when CESN is entered.

In releases before CICS/ESA 4.1, CESN signs off any signed-on user as soon as the transaction identifier is entered, and then presents a signon panel. For example, a user is signed off by first entering CESN, and then pressing F3 when the signon panel is displayed. Doing the same thing under CICS/ESA 4.1 does not cause the user to be signed off.

In CICS/ESA 4.1, CESN signs off any signed-on user only when a new signon attempt is made. Alternatively, the signed-on user is signed off if the CESN transaction identifier is entered with operands (for example USERID=userid).

APPC password expiry management

When you successfully verify your password with CICS/ESA 4.1, you are *not* signed on in the target CICS system, as you were with previous releases of CICS/ESA.

If your CICS connection specifies persistent verification, a successful password verification will cause you to be added to the LUIT table. If no other attaches are received, you will receive a CLS3 transaction flow after the PVDELAY interval.

Transaction-attach security for non-terminal transactions

CICS/ESA 4.1 introduces transaction-attach security for non-terminal transactions, which was never required in earlier releases. This means that when transaction security is active (SEC=YES and XTRAN is not NO) CICS *always* checks the authority of *any* userid to attach a transaction.

In particular, when transactions are started by the EXEC CICS START TRANSID command (without either a TERMID or USERID operand) they inherit the userid associated with the starting transaction, and the inherited userid must be authorized to attach the started transaction.

Therefore you may need to authorize your users to attach transactions that they previously only had authority to start. Alternatively, you can customize RACF so that the transaction-attach check is always successful for non-terminal transactions, as described in “How to bypass attach checks for non-terminal transactions” on page 221.

Chapter 19. Coexistence with previous CICS releases

This chapter covers the following topics:

This chapter discusses the considerations and differences in how you implement RACF security on releases of CICS earlier that CICS/ESA 4.1. You can use RACF with the following releases:

- **CICS/OS/VS Version 1 Release 7 in an MVS/370 or MVS/XA environment**: This release of CICS supports RACF 1.6 and later, upward compatible, releases. See the *CICS/OS/VS Installation and Operations Guide* for information about using RACF with CICS OS/VS 1.7.
- **CICS/MVS 2.1 in an MVS/XA or MVS/ESA environment**: This release of CICS supports RACF 1.6 and later, upward compatible, releases. See the *CICS/MVS Operations Guide* for information about using RACF with CICS/MVS 2.1.
- **CICS/ESA Version 3 Release 1 in an MVS/ESA environment**: This release of CICS supports RACF 1.8.1 and later, upward compatible, releases. See the *CICS/ESA System Definition Guide* for information about using RACF with CICS/ESA 3.1.1.
- **CICS/ESA 3.2.1 in an MVS/ESA environment**: This release of CICS supports RACF 1.8.1 and later, upward compatible, releases, but exploits some functions specific to RACF 1.9.
- **CICS/ESA 3.3 in an MVS/ESA environment**: This release of CICS supports RACF 1.8.1 and later, upward compatible, releases, but exploits some functions specific to RACF 1.9.
- **CICS/ESA 4.1 in an MVS/ESA environment**: This release of CICS supports RACF 1.9 and later, upward compatible, releases, but exploits some functions specific to RACF 2.1.

The pre-4.1 CICS releases listed, together with the stated releases of RACF, support all the security functions described in the earlier chapters of this book for CICS/ESA 4.1, except as shown in Table 35 on page 236.

Table 35. CICS releases in which some security-related functions are not available

Function	Releases in which function is not available
Default userid	**CICS OS/VS 1.7, CICS/MVS 2.1, and CICS/ESA 3.1.1** You cannot use RACF to define a CICS default userid. The security attributes for unsigned-on users are predetermined by CICS.
CICS segment	**CICS OS/VS 1.7, CICS/MVS 2.1, and CICS/ESA 3.1.1** You cannot use RACF to define CICS terminal operator data. To specify operator characteristics, you must define appropriate entries in the CICS sign-on table (SNT).
Command security	**CICS OS/VS 1.7 and CICS/MVS 2.1** You cannot use RACF to perform CICS command security checking in these releases of CICS.
LU 6.2 session security	**CICS OS/VS 1.7, CICS/MVS 2.1, and CICS/ESA 3.1.1** You cannot use RACF to control LU 6.2 session security. You must use the CICS internal mechanisms by specifying a bind password on the appropriate connection definition.
Preset security	**CICS OS/VS 1.7, CICS/MVS 2.1, and CICS/ESA 3.1.1** You cannot use RACF for preset security on terminals and sessions. You must use the CICS internal mechanisms by specifying the appropriate security keys for the terminals and sessions.
QUERY SECURITY	**CICS OS/VS 1.7 and CICS/MVS 2.1** You cannot query the user's security access to RACF-protected resources using this CICS API command. This command, introduced in CICS/ESA 3.1.1 for CICS-managed resources, was extended in CICS/ESA 3.2.1 to permit user applications to query user-defined resources.
SECURITYNAME	**CICS/ESA 4.1** You cannot use this in MRO CONNECTION definitions.
BINDPASSWORD	**CICS/ESA 4.1** You cannot use this in CONNECTION definitions.
Automatic resolution of resource profiles refreshed by SETROPTS RACLIST	**Releases earlier than CICS/ESA 4.1**.
PassTicket support	You cannot generate PassTickets with releases earlier than CICS/ESA 4.1, but you can **use** them in signon in any release.
Reuse of RACF user profiles in VLF between MRO regions.	**Releases earlier than CICS/ESA 4.1**. Only available with RACF 2.1.
Surrogate user support	**Releases earlier than CICS/ESA 4.1**.
VERIFY CHANGE PASSWORD	**Releases earlier than CICS/ESA 4.1**.
USEDFLTUSER	**Releases earlier than CICS/ESA 4.1**.

In addition to the functions not supported, there are differences in the way you specify RACF security on CICS resource definitions, and on system initialization parameters. This chapter explains these differences, and how your CICS/ESA 4.1 regions can coexist in the RACF security environment with earlier releases.

System initialization parameters

If you are using a pre-CICS/ESA 3.2.1 version of CICS, the system initialization parameter for specifying that you want to use RACF is EXTSEC. This parameter is described in the following sections for the earlier releases. The security parameters introduced in CICS/ESA 3.2.1 (SEC, SECPRFX, DFLTUSER, ESMEXITS, XAPPC), described in Chapter 3, "CICS data set and system security" on page 39, should not present any coexistence difficulties.

EXTSEC parameter in CICS OS/VS 1.7 and CICS/MVS 2.1

The options available on the EXTSEC parameter for CICS OS/VS 1.7 and CICS/MVS 2.1 are as follows:

EXTSEC=({NO|YES}[,{NOPREFIX|PREFIX}])

Code this parameter to indicate whether you want CICS to use RACF to control access to resources in the CICS region.

NO

Specify NO if you do not want to use RACF for security. (In pre-CICS/ESA 3.2.1 releases, security defaults to the CICS internal security mechanisms if you specify EXTSEC=NO.)

YES

Specify YES if you do want to use RACF for security checking. In addition to specifying EXTSEC=YES, you must also specify which type of CICS-managed resources you want CICS to control access to through calls to RACF. You do this on the appropriate resource class system initialization parameters, such as XTRAN, XDCT, XFCT and so on. See "CICS resource class system initialization parameters" on page 56 for details of these. (Note that one of the resource class parameters, XAPPC, is supported by CICS/ESA 3.2.1 and CICS/ESA 4.1 only and another, XCMD, is supported by all releases of CICS/ESA Version 4.)

NOPREFIX

Specify NOPREFIX if you do not want the CICS region userid to be used as a prefix to the CICS resource name to form the RACF profile name of the resource.

PREFIX

Specify PREFIX if you want the CICS region userid to be used as a prefix to the CICS resource name to form the RACF profile name of the resource. For example, if the CICS region userid is CICSREG1, and the resource is the CICS-supplied transaction CEMT, CICS passes CICSREG1.CEMT to RACF as the resource profile name.

EXTSEC parameter in CICS/ESA 3.1.1

The options available on the EXTSEC parameter for CICS/ESA 3.1.1 are as follows:

EXTSEC=({NO|YES}[,{NOPREFIX|PREFIX}][,{READ|UPDATE}])

Code this parameter to indicate whether you want CICS to use RACF to control access to resources in the CICS region. The NO|YES and NOPREFIX|PREFIX options are the same as for CICS OS/VS 1.7 and CICS/MVS 2.1.

Use the READ and UPDATE options as follows:

READ

Specify READ if you do not want CICS to distinguish between READ and UPDATE access intent in transactions. This is the default in the system initialization table.

When you specify READ, CICS requires only that the user has at least READ authority, regardless of the access intent.

If command security checking is specified for transactions that contain EXEC CICS system programming commands, then specifying READ means that:

- CICS does not make any check on INQUIRE and COLLECT commands
- CICS checks that the user has at least READ authority for SET, PERFORM, and DISCARD commands.

UPDATE

Specify UPDATE if you want CICS to distinguish between READ and UPDATE access intent in transactions.

When you specify UPDATE, CICS requires the appropriate level of authorization for the access intent: a minimum of READ permission for read intent, and a minimum of UPDATE permission for update intent.

If command security checking is specified for transactions that contain EXEC CICS system programming commands, then specifying UPDATE means that the appropriate level of authority is checked for as follows:

- CICS checks that the user has at least READ authority for INQUIRE and COLLECT commands.
- CICS checks that the user has at least UPDATE authority for SET, DISCARD, and PERFORM commands.

| SEC=MIGRATE for CICS/ESA 3.2 and 3.3 is equivalent to
| EXTSEC=(YES,,READ) for CICS/ESA 3.1.1. For more information about
| this, see “Introduction of UPDATE access authority in CICS/ESA 3.1.1” on page 225.

For the results of the interaction between the READ or UPDATE parameter definitions, the access intent of the user application, and the permission defined to RACF, see Table 36 on page 239.

For more information on defining command security checking for EXEC CICS SP-type commands, see Chapter 8, “CICS command security” on page 109.

Table 36. Determining results of RACF authorization requests			
Access permission defined to RACF for CICS user	**EXTSEC= (YES,,READ) and access intent in application is READ or UPDATE**	**EXTSEC=(YES,,UPDATE)**	
		Access intent in application is READ	**Access intent in application is UPDATE**
NONE	Refused	Refused	Refused
READ	Permitted	Permitted	Refused
UPDATE	Permitted	Permitted	Permitted

Transaction attach security using RACF

Terminal users must sign on if they want to invoke transactions that are subject to transaction-attach security checking. Unlike CICS/ESA 3.2.1, CICS/ESA 3.3, and CICS/ESA 4.1, in which transaction-attach security checking by an external security manager is mandatory, in earlier releases you can control whether you want CICS to call RACF at transaction-attach time. You do this at the individual transaction level by specifying EXTSEC(YES) on the transaction resource definitions (or TRANSEC(EXTERNAL) for CICS/ESA 3.1.1– see the note in Table 37). The effect of the EXTSEC or TRANSEC parameters specified on resource definitions is summarized in Table 37.

Table 37 (Page 1 of 2). Transaction attach security		
Sign-on status	**Resource definition security option**	**Effect at transaction attach**
User is not signed on	EXTSEC(YES) or TRANSEC (EXTERNAL)	The transaction fails with a security violation message. CICS cannot call RACF to check on the terminal user's authority to invoke the transaction without a userid.
User is not signed on	EXTSEC(NO) or TRANSEC(YES)	CICS uses its own internal security mechanism, based on the transaction's CICS resource security keys, to determine whether the terminal user is allowed to invoke the transaction. In this case, the only transactions the user can invoke are those with a security key of 1, the special built-in default security value assigned to all terminal users not signed on.
User is signed on	EXTSEC(YES) or TRANSEC (EXTERNAL)	CICS calls RACF to check on the user's authority to invoke the transaction.

Table 37 (Page 2 of 2). Transaction attach security

Sign-on status	Resource definition security option	Effect at transaction attach
User is signed on	EXTSEC(NO) or TRANSEC (YES)	CICS does not call RACF to check on the user's authority to invoke the transaction.

Note: The transaction-attach security parameter changed in CICS/ESA 3.1.1. EXTSEC(YES|NO) became TRANSEC(YES|EXTERNAL):

```
TRANSEC(EXTERNAL) is the equivalent of EXTSEC(YES)
                   used before CICS Version 3.
TRANSEC(YES)      is the equivalent of EXTSEC(NO)
                   used before CICS Version 3.
```

The TRANSEC parameter, for specifying the security key value on the resource definition in CICS OS/VS 1.7 and CICS/MVS 2.1, changed to TRANSECNUM in CICS/ESA 3.1.1. See "Transaction resource definitions" on page 240 for more information about the changes to resource definition options between the releases.

Transaction resource definitions

You might find it necessary to have earlier releases of CICS coexisting with CICS/ESA 4.1 in the same MVS image, possibly because you cannot migrate all of your CICS regions to CICS/ESA 4.1 in their entirety. If so, you might want to share resource definitions from the same CSD. How you can do this is discussed in the following section, which compares the various resource definition parameters. Some of the obvious differences are illustrated by the CEDA display panels shown in the figures that follow.

Note in Figure 34 that the internal security attributes of earlier releases are shown, even though internal security is not supported in CICS/ESA 3.2.1 and later releases. This is to enable you to share the CSD between different releases of CICS, and continue to maintain the resource definitions for the earlier releases. To update any resource definitions being shared between releases you must use the CICS/ESA 4.1 ALTER function.

```
 OBJECT CHARACTERISTICS                              CICS RELEASE = 0410
  CEDA  VIew


   SECURITY
    RESSec          : No                No | Yes
    Cmdsec          : No                No | Yes
    Extsec          : No                No | Yes
    TRANsec         : 01                1-64
    RSl             : 00                0-24 | Public
                                                          APPLID=CICSTOR
 PF 1 HELP 2 COM 3 END      6 CRSR 7 SBH 8 SFH 9 MSG 10 SB 11 SF 12 CNCL
```

Figure 34. View of part of a CICS/ESA 4.1 transaction resource definition

In the CEDA panel for CICS/ESA 3.1.1 shown in Figure 35 on page 241 the security keywords are changed from those used in CICS OS/VS 1.7 and CICS/MVS 2.1, but the supported function is the same. For details of the changes, compare Figure 35 with Figure 36.

```
 OBJECT CHARACTERISTICS
  CEDA  VIew


  SECURITY
   TRANSEC        : YES               Yes| External
   TRANSECNum     : 01                1-64
   RESSec         : No                No | Yes | External
   RESSECNum      : 00                0-24 | Public
   Cmdsec         : No                No | Yes | External
                                                    APPLID=CICSTOR
PF 1 HELP     3 END        6 CRSR 7 SBH 8 SFH 9 MSG 10 SB 11 SF 12 CNCL
```

Figure 35. View of part of a CICS/ESA 3.1.1 transaction resource definition

```
 OBJECT CHARACTERISTICS
  CEDA  VIew


  SECURITY
   Extsec         : No                No | Yes
   TRANsec        : 01                1-64
   RSL            : 00                0-24 | Public
   RSLC           : No                No | Yes | External
                                                    APPLID=CICSTOR
PF 1 HELP      3 END       6 CRSR 7 SBH 8 SFH 9 MSG 10 SB 11 SF 12 CNCL
```

Figure 36. View of part of a CICS/MVS 2.1 and CICS/OS/VS 1.7 transaction resource definition

Transaction-attach security coexistence

If you are sharing the CSD (for example, in an MRO environment where the AORs are at the CICS/ESA 3.2.1 or CICS/ESA 3.3 level, and the TOR is at an earlier release level), you are recommended to use external security for all connected regions. For attach-time transaction security, this means you should specify the security attributes shown in the first row in Table 38. If you cannot easily convert your old regions to use RACF, you can still share the same resource definitions, even though the earlier releases are using CICS internal security. To use RACF on CICS/ESA 3.2.1 or CICS/ESA 3.3, and internal security on an earlier release, define the transaction security attributes as shown in the second row in the table.

Table 38 (Page 1 of 2). Transaction-attach security definitions across releases

Security attribute on transaction resource definition			**Action taken by the CICS releases indicated in the column headings**
CICS/OS/VS 1.7 and CICS/MVS	**CICS/ESA 3.1.1**	**CICS/ESA 3.2.1, or later**	
EXTSEC(YES)	TRANSEC (EXTERNAL)	None	For all the releases, CICS calls RACF to verify whether the user is permitted to invoke the transaction.

Table 38 (Page 2 of 2). Transaction-attach security definitions across releases

Security attribute on transaction resource definition			**Action taken by the CICS releases indicated in the column headings**
CICS/OS/VS 1.7 and CICS/MVS	**CICS/ESA 3.1.1**	**CICS/ESA 3.2.1, or later**	
EXTSEC(NO) TRANSEC(nn)	TRANSEC (YES) TRANSECNUM (nn)	None	In CICS/ESA 3.2.1, CICS/ESA 3.3, and CICS/ESA 4.1, CICS calls RACF to verify whether the user is permitted to invoke the transaction. In all earlier releases, CICS uses its own internal security mechanisms, comparing the transaction security number from the resource definition (nn) with the terminal user's security keys, to determine whether the user is permitted to invoke the transaction.

Note: To alter resource definitions in a CSD that is being shared between CICS/ESA 4.1 and an earlier release, always make the changes using the CICS/ESA 4.1 ALTER command (in compatibility mode). See Figure 37 for an example of a CEDA panel after pressing PF2.

```
 OBJECT CHARACTERISTICS                                COMPATIBILITY MODE
  CEDA  VIew


  SECURITY
   RESSec          : No                 No | Yes | External
   Cmdsec          : No                 No | Yes | External
   Extsec          : No                 No | Yes
   TRANsec         : 01                 1-64
   RSl             : 00                 0-24 | Public
                                                          APPLID=CICSTOR
PF 1 HELP 2 COM 3 END      6 CRSR 7 SBH 8 SFH 9 MSG 10 SB 11 SF 12 CNCL
```

Figure 37. A CICS/ESA 4.1 transaction resource definition in compatibility mode

Resource security coexistence

If you are sharing the CSD (for example, in an MRO environment where the AORs are at the CICS/ESA 3.2.1 or later level and the TOR is at an earlier release level), you are recommended to use external security for all connected regions. For resource security, this means you should specify resource security attributes as shown in the first row in Table 39 on page 243. If you cannot easily convert your old regions to use RACF, you can still share the same resource definitions, even though the earlier releases are using CICS internal security. To use RACF on CICS/ESA 3.2.1 or CICS/ESA 4.1, and internal security on an earlier release, define the transaction's resource security attributes as shown in the second row in the table.

Table 39. Equivalent resource security definitions across releases

Resource security attribute keywords			Action taken by the CICS releases indicated in the column headings
CICS/OS/VS 1.7 and CICS/MVS	**CICS/ESA 3.1.1**	**CICS/ESA 3.2.1, or later**	
RSLC (EXTERNAL)	RESSEC (EXTERNAL)	RESSEC (YES)	For all releases, CICS calls RACF to verify that the user is permitted to use resources accessed by the transaction, and for which security is specified by the appropriate system initialization parameters.
RSLC(YES)	RESSEC(YES)	RESSEC (YES)	In CICS/ESA 3.2.1 and later, CICS calls RACF to verify whether the user is permitted to access the resource. In the earlier releases, CICS uses its own internal security mechanisms, comparing the security number from the relevant resource definition (RSL number) with the terminal user's security keys to determine whether the user is permitted to access the resource. (The RSL number of the resource being accessed must not be zero.)

Note: To alter resource definitions in a CSD that is being shared between CICS/ESA 4.1 and an earlier release, always make the changes using, in compatibility mode, the CICS/ESA 4.1 version of either CEDA or DFHCSDUP. See Figure 37 for an example of a CEDA panel.

Table 40. Equivalent command security definitions across releases

Command security attribute keywords		**Action taken by the CICS releases indicated in the column headings**
CICS/ESA 3.1.1	**CICS/ESA 3.2.1, or later**	
CMDSEC (EXTERNAL)	CMDSEC (YES)	For all releases, CICS calls RACF to verify that the user is permitted to use commands accessed by the transaction, and for which security is specified by the system initialization parameters XCMD, SEC, or EXTSEC.
CMDSEC(YES)	CMDSEC (YES)	In CICS/ESA 3.2.1 or later, CICS calls RACF to verify whether the user is permitted to use the command. In CICS/ESA 3.1.1, CICS uses its own internal security mechanisms, comparing the RESSECNUM of the relevant DFHEIQxx program name with the terminal user's security keys (RESSECKEYS) to determine whether the user is permitted to use the command. (The RESSECNUM of the program being accessed must not be zero.)
Note: To alter resource definitions in a CSD that is being shared between CICS/ESA 4.1 and an earlier release, always make the changes using, in compatibility mode, the CICS/ESA 4.1 version of either CEDA or DFHCSDUP. See Figure 37 on page 242 for an example of a CEDA panel.		

Extending timeout values

In any one MVS image, there can be many CICS regions, but only one RACF system in which the TIMEOUT values are stored. These values affect only CICS/ESA 3.2.1 and later systems.

If you are using RACF 1.9, which does not support a TIMEOUT value of 99 hours 59 minutes, there are no coexistence issues. Similarly, if you are using RACF 2.1, which does support a TIMEOUT value of 99 hours 59 minutes, and you run only CICS/ESA 4.1 regions, there are no coexistence issues.

However, if you are using RACF 2.1, and you run CICS regions with different levels of CICS, there are coexistence issues that you need to consider. For example, a user may have a TIMEOUT value defined as 0101 (one hour and one minute). In the CICS/ESA 4.1 region, the value will be treated correctly. But in the regions running CICS/ESA 3.3 or CICS/ESA 3.2.1, the hours will be truncated and the value will be treated as one minute.

The situation is even worse if the TIMEOUT value is set as 0200 (two hours). The user in the CICS/ESA 4.1 region will be timed out after two hours, while in the regions running earlier levels of CICS, the user will never be timed out. We recommend that in most cases you use the minutes value that you used on earlier levels of CICS. For upward compatibility, both CICS/ESA 4.1 and RACF accept a value of 0060. You can then add the number of hours, as required, in the CICS/ESA 4.1 region. If the hours value is greater than zero, then the minutes value cannot exceed 59 minutes.

See the *RACF Command Language Reference* manual for information about entering the extended timeout values, and the *RACF General User's Guide* for information about how to list them. In addition, the *RACF System Programmer's Guide* gives information on configuring your panels to use either the old values or the extended values.

MRO bind security with multiple CICS releases in the same MVS

With CICS/ESA 4.1, DFHIRP resides within MVS and outside the individual CICS regions. Therefore, when CICS/ESA 4.1 DFHIRP is installed, all CICS regions in the MVS image, regardless of their release level, use the DFHAPPL.*applid* form of MRO connect security. The CICS/ESA 4.1 implementation of MRO security is therefore forced upon any regions coexisting with a CICS/ESA 4.1 region.

The SECURITYNAME option on the MRO connection definition no longer has any effect on bind-time or link security. This means that there may be a reduced level of security on MRO links, unless you specify otherwise. Any CICS region without either a specific DFHAPPL.*applid* profile (or applicable generic profile) ceases to have MRO connect security.

CICS does not issue any messages to indicate this change.

You can use generic profiles to protect all regions or specific groups of regions against potential security exposures. They can be used before or in parallel with security measures for specific regions. Use the RDEFINE statement in RACF to establish profiles for the links to be protected. For example, specifying the following would ensure that all CICS regions were subject to connect-time security:

```
RDEFINE FACILITY (DFHAPPL.*) UACC(NONE)
```

Authority to use a link can then be specified using the PERMIT command in RACF.

It is still necessary to define regions to DFHIRP as described in "Logon security checking" on page 190.

Part 6. Problem determination

This part describes aspects of problem determination in a CICS-RACF security environment as follows:

Chapter 20. Problem determination in a CICS-RACF security environment

This chapter provides information to help you find the causes of access authority problems.

Resolving problems when access is denied incorrectly

When a user requires access to a protected resource (such as a CICS transaction) and RACF denies the requested access, you will often have to analyze the problem before deciding what action to take.

The basic points to ensure are that:

- CICS is using RACF for this particular kind of resource.
- You know which profile RACF is using to check the user's authority.
- You know which userid is used for the authorization check.

For each security violation, up to three messages are issued:

- CICS issues an authority message to the terminal user (or a "not authorized" return code is issued to an application).
- CICS sends a message DFHXS1111 to the CSCS transient data destination.
- RACF sends an ICH408I message to the CICS region's job log and to the security console.

  For a brief description of message ICH4081, see "RACF message ICH408I" on page 254. (For complete descriptions of this and all other RACF messages, see the *RACF Messages and Codes* manual.)

If message ICH408I is issued for an authorization failure, RACF is active. The message text itself indicates the userid for which the authorization check was done and the name of the RACF profile that was used for the check.

When issued because of a CICS-originated authorization check, the ICH408I message is sent to the CICS region's job log. Most CICS authorization messages also go to the CSCS transient data queue, except DFHIR and DFHZC messages, which go to the CSMT transient data queue.

Note: You can use the CICS-supplied message domain global user exit, XMEOUT, to reroute CICS-issued authorization messages. (For example, you can send them to the same console as the ICH408I messages.) For programming information about using XMEOUT, see the *CICS/ESA Customization Guide*.

If no profile exists for a particular resource, RACF returns a "profile not found" indication to CICS. CICS issues message DFHXS1111 with a SAF return code of X'00000004' and an ESM code of X'00000000'. *No ICH408I message is issued in this case.* The RLIST command issues a message stating that no profile was found.

Note: The RLIST command shows the profile as it exists in the RACF database, which might not necessarily be the same as the in-storage copy that CICS uses.

Note: When you have determined which RACF profile is denying access, see the *RACF Security Administrator's Guide* for a description of authorization checking for RACF-protected resources. The following sections describe some further steps to take in resolving "access denied" problems.

Is CICS using RACF for this particular kind of resource?

- Is CICS using an external security manager (ESM)?

  Make sure that CICS is using an ESM. If it is not using an ESM it issues message DFHXS1102.

- Is security checking done for the particular general resource class? Message DFHXS1105 tells you if the class named on an X*name* parameter has been initialized.

  Note: If message DFHXS1105 is not there, ensure that the SEC=YES system initialization parameter is specified for the region.

  Check the appropriate CICS initialization parameter for the resource. For example: for transactions, this is the XTRAN parameter.

Which profile is RACF using?

- Check the RACF message ICH408I for the name of the profile that RACF used.
- If CICS prefixing is in effect for the CICS region involved, the prefix specified is used as the first qualifier of RACF resource profiles (or member names).
  - Make sure that you have specified the correct prefix as part of resource profile names (on the RDEFINE command) and as member names on the ADDMEM operand.
  - Make sure the CICS job is running under the correct prefix if SECPRFX=YES is specified.
  - Make sure that an installation-written SAF exit is not changing the effective userid under which the CICS region is running.

  Note: The name of the resource in message ICH408I includes the prefix if SECPRFX=YES is specified in the system initialization parameters.
- Is CICS using current copies of the RACF resource profiles?

  If you are using RACF Version 1.9, and you have created, changed, or deleted a resource profile, the in-storage profile does not reflect the change until one of the following is completed:
  - A PERFORM SECURITY REBUILD command for the CICS region

    The EXEC CICS PERFORM SECURITY REBUILD command is complete when NORMAL is displayed on the CEMT user's terminal, or when the EXEC CICS command returns control with no errors.
  - A SETROPTS GENERIC(class-name) REFRESH command (a generic profile has been changed).
  - A restart of the CICS region.

  If you are using RACF Version 2.1, and you have created, changed or deleted a resource profile, the in-storage profile does not reflect the change until the following command completes:

```
SETROPTS RACLIST(classname) REFRESH
```

For more information about refreshing resource profiles, see “Refreshing resource profiles in main storage” on page 29.

Note: If RACF is unable to process the PERFORM SECURITY REBUILD command (for example, due to an abend), CICS may terminate, depending on the circumstances of the abend. For more information on this command, see “Refreshing resource profiles in main storage” on page 29.

- You can use the TSO RLIST command to determine which profile (or profiles) protect the resource. See “Which profile is used to protect the resource?”

Which userid did CICS supply for the authorization check?

Check to see if the user reporting the problem has signed on to CICS. If the user has not signed on to CICS, one of the following could be occurring:

- If you are using preset terminal security, the authorization could be related to that terminal's userid.
- The user could be trying to operate as the CICS default user (without signing on to CICS).
- If the transaction was initiated by a START command, the userid could be inherited from the transaction issuing the START, or specified on the START command itself.
- If the transaction was initiated as the trigger transaction associated with a transient data queue, the userid could have been specified in the DCT for the queue.
- If the program is running as a PLTPI program, the userid could be specified in the PLTPIUSR system initialization parameter.

Note: RACF message ICH408I identifies the userid, as supplied by CICS to RACF, for which the authorization failed.

For help in identifying the user, see Table 1 on page 13.

Which profile is used to protect the resource?

If you are using generic profiles (and you are *not* using resource group profiles), only the most specific profile is used. For example, if the following profiles exist:

```
**
C*
CE*
CEDA
```

CEDA is the profile that is used to control access to the CEDA transaction. If you delete profile CEDA and refresh the in-storage copies, CE⋆ is used.

Note: This assumes CICS prefixing is not used and generic profile checking is used. (That is, that the RACF command SETROPTS GENERIC(class_name) has been issued for the class.)

If resource group profiles have been defined in the relevant class (for example, profiles in the GCICSTRN class), it is possible that more than one profile is used in determining a user's access. To determine which profiles protect the resource,

enter the RLIST command with the RESGROUP operand. Be sure to specify the **member class** on the RLIST command. For example:

```
RLIST TCICSTRN transaction-name RESGROUP
```

If prefixing is in effect for this CICS region, include the prefix on the resource name specified on the RLIST command:

```
RLIST TCICSTRN prefix.transaction-name RESGROUP
```

Note that these examples use the member class TCICSTRN, not the resource group class GCICSTRN.

The AUDITOR attribute enables users to list all profiles that are defined, but does not authorize them to change those profiles. We recommend you give AUDITOR access to system programmers who need to see all profiles (for example, those who are doing problem determination) instead of system-SPECIAL.

As a result of issuing RLIST with RESGROUP, you might see:

- A brief listing of the resource group profile that protects the resource. See Figure 38.
- A slightly longer listing showing the member profile as well as the resource group profile. See Figure 39 on page 253.
- A “profile not found” message, if no profile is found that protects the resource. See Figure 40 on page 253.
- A “not authorized” message, if If a profile exists, but you are not authorized to list the profile. See Figure 41 on page 253.

```
rlist tcicstrn cemt resgroup
CLASS      NAME
-----      ----
TCICSTRN   CEMT
GROUP CLASS NAME
----- ----- ----
GCICSTRN
RESOURCE GROUPS
-------- ------
CAT2
```

Figure 38. Output of RLIST command with RESGROUP: resource group profile

```
rlist tcicstrn cemt resgroup
 CLASS      NAME
 -----      ----
 TCICSTRN   CEMT
 GROUP CLASS NAME
 ----- ----- ----
 GCICSTRN
 RESOURCE GROUPS
 -------- ------
 CICSCAT2A
 LEVEL  OWNER      UNIVERSAL ACCESS  YOUR ACCESS  WARNING
 -----  --------   ----------------  -----------  -------
  00    PUB01           NONE              ALTER    NO
 INSTALLATION DATA
 -----------------
 NONE
 APPLICATION DATA
 ----------------
 NONE
 AUDITING
 --------
 FAILURES(READ)
 NOTIFY
 ------
 NO USER TO BE NOTIFIED
```

Figure 39. Output of RLIST command with RESGROUP: several profiles

Note: When you are using resource group profiles, more than one profile might be used at a time. If the resource is protected by more than one profile, you are strongly urged to delete all other occurrences of the resource name. Use the DELMEM operand on the RALTER command to remove the resource name from existing resource group profiles. Use the RDELETE command — with care — to delete profiles from the member class.

```
rlist tcicstrn dfhcicsm.cemt resgroup
 ICH13003I DFHCICSM.CEMT NOT FOUND
```

Figure 40. Output of RLIST command with RESGROUP: profile not found

Note: If you get the “profile not found” message, make sure that generic profile processing is in effect for the specified class. (SETROPTS LIST will show this.) If, indeed, no profile exists, create a suitable profile and ensure that the appropriate users and groups have access.

```
rlist tcicstrn dfhcicsm.cemt resgroup
 ICH13002I NOT AUTHORIZED TO LIST DFH*
```

Figure 41. Output of RLIST command with RESGROUP: authorization message

The “not authorized” message identifies the name of the profile preventing you from having access. You can ask the RACF security administrator (who has the system-SPECIAL attribute and can therefore list the profile) to investigate the problem.

Some possible solutions are:

- The profile is not needed and can be deleted.
- You can be made OWNER of the profile.
- A more specific generic profile can be created, and you or your group can be made OWNER of the new profile.
- If the profile is a discrete profile, you can be given ALTER access to the profile.
- You can be assigned the AUDITOR attribute.

For a description of the authority needed to list a general resource profile, see the description of the RLIST command in the *RACF Command Language Reference* manual.

RACF message ICH408I

For a complete description of RACF message ICH408I, see the *RACF Messages and Codes* manual.

The first line of message ICH408I identifies a user that had an authorization problem. The other lines of the message describe the request the user was issuing and the reason for the failure.

See the following example:

```
ICH408I USER(JONES   ) GROUP(DEPT60  ) NAME(M.M.JONES           )
ICH408I   FLA32 CL(FCICSFCT)
ICH408I   INSUFFICIENT ACCESS AUTHORITY
ICH408I   FROM F%A* (G)
ICH408I   ACCESS INTENT(UPDATE )  ACCESS ALLOWED(READ   )
```

This message can be interpreted as follows:

User JONES, a member of group DEPT60, whose name is M.M.JONES, had INSUFFICIENT ACCESS AUTHORITY to resource FLA32, which is in class FCICSFCT.

Note: If the class shown is a resource group class, the profile might be in the class shown or the related member class. For example, if GCICSTRN appears, check TCICSTRN also. If HCICSFCT appears, check FCICSFCT also. For a list of all the IBM-supplied class names, see Table 3 on page 28. For a list of the installation-defined class names that are in use on your installation, see your RACF system programmer, or issue the SETROPTS LIST command.

The RACF profile protecting the resource is F%A*. "(G)" indicates that F%A* is a generic profile.

The access attempted by user JONES was UPDATE, but the access allowed by RACF was READ. Therefore, user JONES was denied access.

A DFHXS1111 message would also be issued for this access attempt:

```
DFHXS1111 26/07/94 15:34:01 CICSSYS1 Security violation by user JONES
          at netname D2D1 for resource FLA32 in class
          TCICSTRN. SAF codes are (X'00000008',X'00000000'). ESM codes
          are (X'00000008',X'00000000').
```

The SAF and ESM codes come from RACROUTE REQUEST=AUTH.

To find the cause of the violation, issue the RLIST command with AUTHUSER specified to list the profile indicated in the ICH408I message. The AUTHUSER operand requests display of the access list, as shown in Figure 42.

```
rlist fcicsfct f%a* authuser
CLASS      NAME
-----      ----
FCICSFCT   F%A* (G)
GROUP CLASS NAME
----- ----- ----
HCICSFCT
LEVEL  OWNER      UNIVERSAL ACCESS  YOUR ACCESS  WARNING
-----  --------   ----------------  -----------  -------
 00    CICSADM          NONE              ALTER    NO
                              .
                              .
                              .
                              .
USER      ACCESS
----      ------
DEPTA     UPDATE
JONES     READ
GROUPX    NONE
SYSPROG   ALTER
```

Figure 42. Requesting a display of the access list

In this profile, user JONES has an explicit entry in the access list. If the userid itself does not appear in the access list, check for one of JONES's connect groups. (Issue LISTUSER JONES to list the groups to which JONES is connected.) Other specifications in the profile (such as security level or security category) might cause access to be denied. For a complete description, see the *RACF Security Administrator's Guide*.

Note: If NOTIFY(CICSADM) were specified in this profile, TSO userid CICSADM would receive immediate notification of failed attempts to access resources protected by the profile.

Resolving problems when access is allowed incorrectly

There could be many reasons why a user might have access to a protected resource, when you think that the user should *not* have that access. The following is a list of checks that you can make to investigate this kind of situation:

- Confirm which userid the user is signed on as. (Make sure the user has actually signed on and is not acting as the CICS default user.) You can ask the user to sign off, then sign on again. You can also ask the user to issue EXEC CICS ASSIGN or EXEC CICS INQUIRE TERMINAL, which can be issued with the CECI transaction (or a user-written transaction).
- Make sure that the SEC system initialization parameter is SEC=YES for the CICS region the user is signed on to.

  If SEC=NO is specified, users can access any resource.

- If the user is running a transaction that communicates with other regions such as application-owning regions (AORs) or file-owning regions (FORs), make sure that the SEC system initialization parameter is SEC=YES for those regions.
- Is prefixing correct?
  - Has the CICS JOB been submitted by the correct USER?
  - Is SECPRFX set correctly?
  - Has an installation-written SAF exit been used to return a different CICS region userid when RACROUTE=EXTRACT has been specified?
- Depending on the resource, make sure that RESSEC(YES) is specified for each transaction that might access that resource.
- Is the appropriate X*name* CICS system initialization parameter correctly set?

  For example, if it is a file control request, is XFCT=YES or XFCT=value specified? If the X*name* parameter specifies a value other than YES or NO, does the value show the correct installation-defined class name?
- Is the transaction exempt from transaction security? (For information on transactions that may have been defined in this way, see "Category 3 transactions" on page 131.)
- Does the transaction have the correct RESSEC and CMDSEC options?
- Check that the RESSEC setting on the MIRROR transaction is correct.
- If the resource is temporary storage, are you using the correct TST? Check:
  - The DFHTST TYPE=SECURITY entry in the TST
  - That TST entries are in the correct order.
- If intersystem communication is involved, check the following:
  - Is a SECURITY REBUILD required (on this or on the remote system)?
  - If ATTACHSEC=LOCAL is specified, does the SECURITYNAME userid have access to the resource?
  - Is ATTACHSEC=IDENTIFY specified?
  - Check for 'equivalent systems' causing link security to be bypassed.
  - Is the remote system using the same RACF database?
- Do you have any RACF installation exits?
- To check the profile that you think protects the resource, use the checklist provided in the *RACF Security Administrator's Guide*.

CICS/ESA initialization failures related to security

From CICS/ESA 4.1, if SEC=YES or is specified, external security is *required*. If external security cannot be provided, CICS/ESA cannot be initialized.

Figure 43 shows an example of a failure to initialize.

```
DFHPA1927  DBDCCICS SEC=YES,SECPRFX=NO
DFHPA1927  DBDCCICS XUSER=YES
DFHPA1927  DBDCCICS DFLTUSER=SEC6D01
DFHPA1927  DBDCCICS PLTPISEC=NONE
DFHPA1927  DBDCCICS PLTPIUSR=SEC7P01
DFHPA1927  DBDCCICS XCMD=YES
DFHPA1927  DBDCCICS XDCT=YES
DFHPA1927  DBDCCICS XFCT=YES
DFHPA1927  DBDCCICS XJCT=YES
DFHPA1927  DBDCCICS XPCT=YES
DFHPA1927  DBDCCICS XPSB=YES
DFHPA1927  DBDCCICS XPPT=UNKNOWN
DFHPA1927  DBDCCICS XTST=YES
DFHPA1927  DBDCCICS XTRAN=YES
DFHPA1927  DBDCCICS XAPPC=YES
DFHPA1927  DBDCCICS SNSCOPE=NONE
DFHPA1927  DBDCCICS .END
DFHPA1101  DBDCCICS DFHSITDZ IS BEING LOADED.
DFHPA1108  DBDCCICS DFHSITDZ HAS BEEN LOADED.
                   (GENERATED AT: MM/DD= 07/26 HH:MM= 13:25).
+DFHTR0103 TRACE TABLE SIZE IS 2000K
+DFHSM0122I CICSGSA1 Limit of DSA storage below 16MB is 5,120K.
+DFHSM0123I CICSGSA1 Limit of DSA storage above 16MB is 20M.
+DFHSM0115I CICSGSA1 Storage protection is active.
+DFHSM0126I CICSGSA1 Transaction isolation is not active.
+DFHDM0101I CICSGSA1 CICS is initializing.
+DFHSI1500 CICSGSA1 CICS/ESA Version 4.1.0 Startup is in progress.
+DFHXS1100I CICSGSA1 Security initialization has started.
+DFHSI1501I CICSGSA1 Loading CICS nucleus.
+DFHDU0304I CICSGSA1 Transaction Dump Data set DFHDMPA  opened.
+DFHXS1105 CICSGSA1 Resource profiles for class ACICSPCT have been built.
+DFHXS1105 CICSGSA1 Resource profiles for class CCICSCMD have been built.
+DFHXS1105 CICSGSA1 Resource profiles for class DCICSDCT have been built.
+DFHXS1105 CICSGSA1 Resource profiles for class FCICSFCT have been built.
+DFHXS1105 CICSGSA1 Resource profiles for class JCICSJCT have been built.
+DFHXS1106 CICSGSA1
 Resource profiles could not be built for class MUNKNOWN. CICS is
 terminated. SAF codes are (X'00000004',X'00000000'). ESM codes are
 (X'00000000',X'00000000').
+DFHDU0303I CICSGSA1 Transaction Dump Data set DFHDMPA  closed.
+DFHKE1800 CICSGSA1 ABNORMAL TERMINATION OF CICS/ESA IS COMPLETE.
```

Figure 43. Security initialization failure

If security initialization fails:

1. Examine the DFHXS1106 message return codes. In the example shown in Figure 43 on page 257, SAF return code X'00000004' and reason code X'00000000' were issued:
   - A return code of X'00000004' indicates that an error occurred in the MVS security router (RACROUTE). See the RACROUTE macro reference in "CICS security control points" on page 216.
2. Check the CICS startup options, in particular the X*name* system initialization parameters. Make sure that:
   - The class is defined to RACF and is active (use the SETROPTS LIST command to check this).
   - The class is defined in the router table. To do this, examine the installation source for ICHRFR01 for any installation-defined classes. (The description of the ICHFRTB macro in the *RACF Macros and Interfaces* manual includes a listing of the IBM-supplied module, ICHRFR0X.)

Figure 43 on page 257 shows that XPPT=UNKNOWN has been specified. This causes CICS to try to use a class called MUNKNOWN. MUNKNOWN has not been defined to the MVS router or the RACF CDT.

RACF abends

If a RACF abend occurs, see the *RACF Messages and Codes* manual and the *RACF Diagnosis Guide* for further guidance.

SAF or RACF installation exits

Check if any SAF or RACF installation exits are causing initialization (RACLIST) requests to fail.

CICS default user fails to sign on

Figure 44 on page 259 shows an example of a CICS job log if the DFLTUSER fails to sign on. CICS is started with SEC=YES and DFLTUSER=AUSER1. User profile AUSER1 has not been defined to RACF.

```
DFHPA1927  DBDCCICS SEC=YES,SECPRFX=NO
DFHPA1927  DBDCCICS XUSER=YES
DFHPA1927  DBDCCICS DFLTUSER=UNKNOWN
DFHPA1927  DBDCCICS PLTPISEC=NONE
DFHPA1927  DBDCCICS PLTPIUSR=SEC7P01
DFHPA1927  DBDCCICS XCMD=YES
DFHPA1927  DBDCCICS XDCT=YES
DFHPA1927  DBDCCICS XFCT=YES
DFHPA1927  DBDCCICS XJCT=YES
DFHPA1927  DBDCCICS XPCT=YES
DFHPA1927  DBDCCICS XPSB=YES
DFHPA1927  DBDCCICS XPPT=YES
DFHPA1927  DBDCCICS XTST=YES
DFHPA1927  DBDCCICS XTRAN=YES
DFHPA1927  DBDCCICS XAPPC=YES
DFHPA1927  DBDCCICS .END
DFHPA1101  DBDCCICS DFHSITDZ IS BEING LOADED.
DFHPA1108  DBDCCICS DFHSITDZ HAS BEEN LOADED.
                    (GENERATED AT: MM/DD= 07/26 HH:MM= 13:25).
+DFHTR0103 TRACE TABLE SIZE IS 2000K
+DFHSM0122I CICSGSA1 Limit of DSA storage below 16MB is 5,120K.
+DFHSM0123I CICSGSA1 Limit of DSA storage above 16MB is 20M.
+DFHSM0115I CICSGSA1 Storage protection is active.
+DFHSM0126I CICSGSA1 Transaction isolation is not active.
+DFHDM0101I CICSGSA1 CICS is initializing.
+DFHXS1100I CICSGSA1 Security initialization has started.
+DFHSI1500 CICSGSA1 CICS/ESA Version 4.1.0 Startup is in progress.
+DFHXS1105 CICSGSA1 Resource profiles for class ACICSPCT have been built.
+DFHXS1105 CICSGSA1 Resource profiles for class CCICSCMD have been built.
+DFHXS1105 CICSGSA1 Resource profiles for class DCICSDCT have been built.
+DFHSI1501I CICSGSA1 Loading CICS nucleus.
+DFHXS1105 CICSGSA1 Resource profiles for class FCICSFCT have been built.
+DFHXS1105 CICSGSA1 Resource profiles for class JCICSJCT have been built.
+DFHXS1105 CICSGSA1 Resource profiles for class MCICSPPT have been built.
+DFHXS1105 CICSGSA1 Resource profiles for class PCICSPSB have been built.
+DFHXS1105 CICSGSA1 Resource profiles for class SCICSTST have been built.
+DFHXS1105 CICSGSA1 Resource profiles for class TCICSTRN have been built.
+DFHXS1105 CICSGSA1 Resource profiles for class SURROGAT have been built.
ICH408I USER(UNKNOWN ) GROUP(        ) NAME(???                 )
  LOGON/JOB INITIATION - USER AT TERMINAL          NOT RACF-DEFINED
IRR012I  VERIFICATION FAILED. USER PROFILE NOT FOUND.
+DFHXS1104 CICSGSA1
 Default security could not be established for userid UNKNOWN. The
 security domain cannot continue, so CICS is terminated. SAF codes are
 (X'00000004',X'00000000'). ESM codes are (X'00000004',X'00000000').
+DFHKE1800 CICSGSA1 ABNORMAL TERMINATION OF CICS/ESA IS COMPLETE.
```

Figure 44. Example of CICS job log if DFLTUSER fails to sign on

This CICS region cannot be initialized because, with SEC=YES specified, external security is required and the default user must be defined to RACF.

Revoked user attempting to sign on

The following example sequence illustrates what happens when a revoked user attempts to sign on:

1. User USR001 attempts to sign on using CESN. However, the user is revoked. The user sees the following on the terminal:

   ```
   DFHCE3546 Your signon userid has been revoked. Signon is terminated.
   ```

2. A RACF ICH408I message is sent to the CICS region's job log:

   ```
   ICH408I USER(USR001  ) GROUP(GRP001  ) NAME(AUSER           )
     LOGON/JOB INITIATION - REVOKED USER ACCESS ATTEMPT
   ```

   This message indicates that user USR001, whose name as recorded in the RACF user profile is AUSER, and whose current RACF connect group is GRP001, attempted to sign on.

3. A CICS message is sent to the CSCS transient data queue:

   ```
   DFHSN1120 26/07/94 12:20:24 CICSSYS1 Signon at netname D2D1
   with userid USR001 failed because the userid has been revoked.
   ```

User has insufficient authority to access a resource

Now let us consider user USR001, who has signed on successfully with current connect group GRP001. User USR001 attempts unsuccessfully to use transaction CEMT, which is protected by profile CAT2 in class GCICSTRN (the resource group class for CICS transactions), because XTRAN=YES is specified in the CICS system initialization parameters.

1. The terminal user received the following CICS message:

   ```
   DFHAC2033 26/07/94 15:18:44 CICSSYS1 You are not authorized to use
   transaction CEMT. Check that the transaction name is correct.
   ```

2. A RACF ICH408I message is sent to the CICS region's job log:

   ```
   ICH408I USER(USR001  ) GROUP(GRP001  ) NAME(AUSER         )
   ICH408I   CEMT CL(TCICSTRN)
   ICH408I   INSUFFICIENT ACCESS AUTHORITY
   ICH408I   ACCESS INTENT(READ   )  ACCESS ALLOWED(NONE   )
   ```

   This message indicates that user USR001, whose name as recorded in the RACF user profile is AUSER, and whose current RACF connect group is GRP001, attempted to use the CEMT transaction. To do this, AUSER needs to have at least READ access to the profile protecting the CEMT transaction. However, RACF determined that AUSER had *no* access authority.

3. A CICS message is sent to the CSCS transient data queue:

   ```
   DFHXS1111 26/07/94 13:30:41 CICSSYS1 CEMT Security violation
             by user USR001 at netname D2D1 for resource CEMT in class
             TCICSTRN. SAF codes are (X'00000008',X'00000000'). ESM codes
             are (X'00000008',X'00000000').
   ```

   The following message is also sent to the CSMT transient data queue:

   ```
   DFHAC2003 26/01/94 15:18:44 CICSSYS1 Security violation has been
   detected term id = D2D1, trans id = CEMT, userid = USR001.
   ```

4. Which profile protects CEMT?

   It appears from the ICH408I message that profile CEMT in class TCICSTRN protects CEMT. However, this is not necessarily the case. A resource group

profile (in class GCICSTRN) might protect CEMT. In fact, in this case, there is no profile named CEMT. If a system-SPECIAL or AUDITOR user issues the SEARCH command with CLASS(TCICSTRN) specified, no profile named CEMT would appear.

To determine which profile was actually used, you must issue the RLIST command with the RESGROUP operand as follows:

```
RLIST member-class resource-name RESGROUP
```

In this case, issue the following:

```
RLIST TCICSTRN CEMT RESGROUP
```

Note: If prefixing is used for this CICS region, specify the prefix on the resource-name in the RLIST command.

RACF displays the following:

```
CLASS      NAME
-----      ----
TCICSTRN   CEMT
GROUP CLASS NAME
----- ----- ----
GCICSTRN
RESOURCE GROUPS
-------- ------
CAT2
```

The profiles in class GCICSTRN that protect CEMT are shown under RESOURCE GROUPS in the command output. In this case, only one profile (CAT2) protects profile CEMT.

Note: If a profile in class TCICSTRN protected CEMT, that profile's contents would be added to the output of RLIST.

5. To determine how profile CAT2 protects CEMT, list that profile with the AUTHUSER operand specified on the RLIST command:

   ```
   RLIST GCICSTRN CAT2 AUTHUSER
   ```

   RACF displays the following:

   ```
   CLASS      NAME
   -----      ----
   GCICSTRN   CAT2
   MEMBER CLASS NAME
   ------ ----- ----
   TCICSTRN
   RESOURCES IN GROUP
   --------- -- -----
   CDBC
   CDBI
   CBRC
   CEDA
   CEMT
   CETR
   LEVEL  OWNER      UNIVERSAL ACCESS  YOUR ACCESS  WARNING
   -----  --------   ----------------  -----------  -------
   ⋮
   NOTIFY
   ------
   NO USER TO BE NOTIFIED
   ```

```
USER      ACCESS   ACCESS COUNT
----      ------   ------ -----
DEPTA     ALTER       000000
USR001    NONE        000000
-------- -------  ------------ -------- -------------------------
NO ENTRIES IN CONDITIONAL ACCESS LIST
```

Password expiry management problem determination

If you are running a CICS-APPC PEM environment, and are not receiving the expected responses, check the following possible sources of errors in the signon transaction program:

- The function management header (FMH) may be in error; check that:
  - The conversation type being used is **basic**.
  - The XTRANID in the CICS TRANSACTION definition for CLS4 is X'06F3F0F1'. (See “Setting up the PEM requester” on page 169.)
  - The CICS PEM server signon transaction is running as a **synclevel 0** transaction. (See “Setting up the PEM requester” on page 169.)
- The user data may be in error; check that:
  - Valid lengths are being sent; see Table 25 on page 173, Table 26 on page 174, and “Format of user data” on page 170.
  - Userids and passwords are sent in EBCDIC. (See “Setting up the PEM requester” on page 169.)
  - GDS variables (required in basic conversations) are being used: (See “Format of user data” on page 170.)

Note: If the CICS PEM server receives an error in the FMH or user data, it sends an ISSUE ERROR to the PEM requester, and terminates without an abend. If this happens, it is likely that there is an error in the flow. For examples of valid flows, see “Examples of PEM requester and CICS PEM server user data” on page 178.

Execution diagnostic facility (EDF): The execution diagnostic facility (EDF) *cannot* be used to check DFHCLS4, for security reasons, because user passwords would be displayed on the EDF screens.

Part 7. Appendixes

The three appendixes are:

Appendix A. APPC PEM requester example program

This appendix contains General-Use Programming Interface and Associated Guidance information.

This appendix contains an APPC PEM requester example program, written in C/2 for use on an IBM PS/2 computer running OS/2. ***Note that this example program is provided for guidance only***.

You may find it useful to copy and modify this example when coding your own PEM requester.

If you intend to write a program similar to this example, you may need to refer to the *OS/2 Extended Edition APPC Programming Reference* manual, which tells you how to write programs using the Operating System/2 (OS/2) Extended Edition APPC Interface.

For examples of PEM requester and CICS PEM server user data produced by this program, see:

- “Signon with correct userid and password” on page 178
- “Signon with new password” on page 179
- “Response to correct signon data” on page 180
- “Response to incorrect data format” on page 182.

```
/*****************************************************************************/
/*                                                                           */
/* MODULE NAME : PEM.C                                                       */
/*                                                                           */
/* DESCRIPTIVE : APPC Password Expiration Management Requester Sample Program*/
/*               for OS/2 Extended Edition                                   */
/*                                                                           */
/*****************************************************************************/

#define LINT_ARGS

#include <APPC_C.H>
#include <ACSSVCC.H>
#include <STDIO.H>
#include <STDLIB.H>
#include <STDDEF.H>
#include <STRING.H>
#include <DOS.H>
#include <DOSCALLS.H>

/* Macro clear_vbc sets the APPC verb control block to zero */
#define clear_vcb()   memset(&vcb,(int)'\0',sizeof(vcb))

typedef unsigned int BOOL;
#define FALSE                 0
#define TRUE                  1

/* Type of PEM function, and possible values */
typedef unsigned int resp_type;
#define PEM_OK                0
#define TP_STARTED_FAIL       1
#define ALLOCATE_FAIL         2
#define BUFF_ALLOC_FAIL       3
#define SEND_FAIL             4
#define RECEIVE_FAIL          5
#define RECEIVE_DEALLOC_FAIL  6
#define TP_ENDED_FAIL         7

/* PEM has two call functions: Sign On and Change Password */
typedef unsigned int func_type;
#define SIGN_ON               0
#define CHANGE                1

/* APPC return code: Primary and Secondary */
typedef struct ret_code_str {
        unsigned short p;
        unsigned long  s;
        } APPC_rc;
```

Figure 45 (Part 1 of 20). APPC PEM requester example program

```
/* Length Constants */
#define PEM_REQ_LEN          45
#define PEM_RESP_LEN         50
#define PARTNER_LU_ALIAS_LEN  8
#define TP_NAME_LEN          64
#define LU_ALIAS_LEN          8
#define MODE_NAME_LEN         8
#define TP_ID_LEN             8

/* Pointer to alphanumeric shared buffer */
unsigned char far *anbptr;


/* Verb Control Block (vcb) - See APPC.H for component declarations */
union  {
       struct allocate              alloc;
       struct deallocate            dealloc;
       struct receive_and_wait      rcv_wait;
       struct send_data             send;
       struct tp_started            tpstart;
       struct tp_ended              tpend;
       struct convert               cnvt;
       } vcb;

/* Pointer to the vcb */
unsigned far *vcbptr;

/* Function Prototypes */
#ifdef LINT_ARGS
void      main(int,char **);
resp_type pem(func_type,char *,char *,char *,char *,char *,char *,char *,
              char *);
void      build_request(func_type,char *,char *,char *,char *);
APPC_rc   TP_STARTED(char *,char *,char *);
APPC_rc   ALLOCATE(char *,char *,char *,char *,unsigned long *);
APPC_rc   SEND_REQUEST(char *,unsigned long,char *);
APPC_rc   RECEIVE_RESPONSE(char *,unsigned long,char *);
APPC_rc   RECEIVE_DEALLOC(char *,unsigned long);
APPC_rc   TP_ENDED(char *);
BOOL      get_shared_buffer(void);
void      free_shared_buffer(void);
void      convert_to_EBCDIC(char *, unsigned int);
void      display_PEM_response(char *);
#endif
```

Figure 45 (Part 2 of 20). APPC PEM requester example program

```
/*****************************************************************************/
/* MAIN function                                                             */
/*                                                                           */
/* This function is a skeleton driver for the PEM function,  It takes the    */
/* arguments specified on the program invocation, and passes them on to PEM. */
/*                                                                           */
/* Syntax of program invocation is:                                          */
/*                                                                           */
/* PEM partner_LU_alias LU_alias mode_name user_id password {<new_password>} */
/*                                                                           */
/* Input:                                                                    */
/*   ARGC                                                                    */
/*   ARGV                                                                    */
/*                                                                           */
/*****************************************************************************/

void main(argc,
          argv)

int       argc;
char      *argv[];
{
  resp_type resp;

  func_type function_required;

  unsigned short i, j;




  char      tps_tp_name[TP_NAME_LEN],            /* TP name for TP Started   */
            partner_LU_alias[PARTNER_LU_ALIAS_LEN],
            LU_alias[LU_ALIAS_LEN],
            mode_name[MODE_NAME_LEN],
            user_id[9],                          /* 8 Chars plus null        */
            password[9],
            new_password[9],
            PEM_response[PEM_RESP_LEN];          /* PEM Response Structure   */

  switch (argc) {                                /* If new password specified*/
  case  6 : function_required = SIGN_ON;         /* No, then Sign On         */
            break;
  case  7 : function_required = CHANGE;          /* Yes, then Change Password*/
            strcpy(new_password, strupr(argv[6]));
            break;
  default : printf("Invalid Syntax\n\n");
            printf("Command Format is:\n\n");
            printf("PEM <partner_LU_alias> <LU_alias> <mode_name> <User_Id> <Password> {<New_Password>}\n");
            exit(0);
            break;
  }
```

Figure 45 (Part 3 of 20). APPC PEM requester example program

```
/* Set APPC parameters to blanks */
memset(tps_tp_name,     (int) ' ', TP_NAME_LEN);
memset(partner_LU_alias,(int) ' ', PARTNER_LU_ALIAS_LEN);
memset(LU_alias,        (int) ' ', LU_ALIAS_LEN);
memset(mode_name,       (int) ' ', MODE_NAME_LEN);

/* Set APPC Parameters to values specified on the program invocation.  */
/* Note, as memcpy is being used, the resulting strings will not be    */
/* null terminated.                                                    */
memcpy(tps_tp_name,      strupr(argv[0]), strlen(argv[0]));
memcpy(partner_LU_alias, strupr(argv[1]), strlen(argv[1]));
memcpy(LU_alias,         strupr(argv[2]), strlen(argv[2]));
memcpy(mode_name,        strupr(argv[3]), strlen(argv[3]));

/* Set the User-id and password fields */
strcpy(user_id,  strupr(argv[4]));
strcpy(password, strupr(argv[5]));


vcbptr = (unsigned far *)&vcb;                /* Set pointer to vbc       */

resp = pem(function_required,                 /* Issue the PEM request    */
           tps_tp_name,
           partner_LU_alias,
           LU_alias,
           mode_name,
           user_id,
           password,
           new_password,
           PEM_response);




switch (resp) {                               /* Check outcome of request */
case PEM_OK               : printf("PEM_OK\n\n");
                            display_PEM_response(PEM_response);
                            break;
case TP_STARTED_FAIL      : printf("TP_STARTED_FAIL\n");
                            break;
case ALLOCATE_FAIL        : printf("ALLOCATE_FAIL\n");
                            break;
case BUFF_ALLOC_FAIL      : printf("BUFF_ALLOC_FAIL\n");
                            break;
case SEND_FAIL            : printf("SEND_FAIL\n");
                            break;
case RECEIVE_FAIL         : printf("RECEIVE_FAIL\n");
                            break;
case RECEIVE_DEALLOC_FAIL : printf("RECEIVE_DEALLOC_FAIL\n");
                            break;
case TP_ENDED_FAIL        : printf("TP_ENDED_FAIL\n");
                            break;
```

Figure 45 (Part 4 of 20). APPC PEM requester example program

```
  default                  : printf("Invalid Resp from PEM\n");
                             break;
  }
}
/*****************************************************************************/

/*****************************************************************************/
/* APPC Password Expiration Manager (PEM) Requester                          */
/*                                                                           */
/* Builds a PEM request structure, sends it to the CICS Server, and receives */
/* the response.  This response structure is passed back to the caller.      */
/*                                                                           */
/* Input:                                                                    */
/*   function_required                                                       */
/*   tps_tp_name                                                             */
/*   partner_LU_alias                                                        */
/*   LU_alias                                                                */
/*   mode_name                                                               */
/*   user_id                                                                 */
/*   password                                                                */
/*   new_password                                                            */
/*                                                                           */
/* Output:                                                                   */
/*   PEM_response                                                            */
/*                                                                           */
/* Return Value:                                                             */
/*   rerp_type = {PEM_OK, TP_STARTED_FAIL, ALLOCATE_FAIL, BUFF_ALLOC_FAIL,   */
/*                SEND_FAIL, RECEIVE_FAIL, RECEIVE_DEALLOC_FAIL,             */
/*                TP_ENDED_FAIL}                                             */
/*                                                                           */
/*****************************************************************************/

resp_type pem (function_required,
               tps_tp_name,
               partner_LU_alias,
               LU_alias,
               mode_name,
               user_id,
               password,
               new_password,
               PEM_response)

func_type      function_required;
char           tps_tp_name[],
               partner_LU_alias[],
               LU_alias[],
               mode_name[],
               user_id[],
               password[],
               new_password[],
               PEM_response[];
```

Figure 45 (Part 5 of 20). APPC PEM requester example program

```
{
  char         tp_name[TP_NAME_LEN],
               tp_id[TP_ID_LEN],
               PEM_request[PEM_REQ_LEN];

  BOOL buffer_allocated;
  APPC_rc ret_code;                 /* APPC Primary & Secondary return code */
  unsigned long conv_id;            /* APPC Conversation Id                 */

  /* Set TP Name of CICS PEM server transaction - 0x06F3F0F1 */
  memset(tp_name,     (int) ' ',TP_NAME_LEN);




  tp_name[0] = (char) 0x06;
  tp_name[1] = (char) 0xF3;
  tp_name[2] = (char) 0xF0;
  tp_name[3] = (char) 0xF1;


  /* DOSALLOCSEG for APPC data buffer */
  buffer_allocated = get_shared_buffer();

  if (!buffer_allocated) {
     printf("Buffer allocation failed\n");
     return(BUFF_ALLOC_FAIL);
  }

  /* Build a PEM request structure from the given parameters */
  build_request(function_required,
                user_id,
                password,
                new_password,
                PEM_request);

  /*Tell APPC a TP has Started */
  ret_code = TP_STARTED(LU_alias,
                        tps_tp_name,
                        tp_id);

  if (ret_code.p != AP_OK) {
     printf("TP_STARTED failed\n");
     printf("APPC Return Code - Primary  : %#010hx \n", ret_code.p);
     printf("                 - Secondary: %#010lx \n", ret_code.s);
     return(TP_STARTED_FAIL);
  }
```

Figure 45 (Part 6 of 20). APPC PEM requester example program

```
/* Allocate APPC session for conversation */
ret_code = ALLOCATE(tp_id,
                    partner_LU_alias,
                    tp_name,
                    mode_name,
                    &conv_id );

if (ret_code.p != AP_OK) {
   printf("ALLOCATE failed\n");
   printf("APPC Return Code - Primary  : %#010hx \n", ret_code.p);
   printf("                 - Secondary: %#010lx \n", ret_code.s);
   return(ALLOCATE_FAIL);
}

/* Send request to the CICS PEM Server, inviting a response */
ret_code = SEND_REQUEST(tp_id,
                        conv_id,
                        PEM_request);

if (ret_code.p != AP_OK) {
   printf("SEND_REQUEST failed\n");
   printf("APPC Return Code - Primary  : %#010hx \n", ret_code.p);
   printf("                 - Secondary: %#010lx \n", ret_code.s);
   return(SEND_FAIL);
}




/* Receive response from the CICS PEM Server */
ret_code = RECEIVE_RESPONSE(tp_id,
                            conv_id,
                            PEM_response);

if (ret_code.p != AP_OK) {
   printf("APPC RECEIVE failed\n");
   printf("APPC Return Code - Primary  : %#010hx \n", ret_code.p);
   printf("                 - Secondary: %#010lx \n", ret_code.s);
   return(RECEIVE_FAIL);
}

/* Receive Deallocation notification from the CICS PEM Server */
ret_code = RECEIVE_DEALLOC(tp_id,
                           conv_id);

free_shared_buffer();

if (ret_code.p != AP_DEALLOC_NORMAL) {
   printf("APPC RECEIVE_DEALLOCATE failed\n");
   printf("APPC Return Code - Primary  : %#010hx \n", ret_code.p);
   printf("                 - Secondary: %#010lx \n", ret_code.s);
   return(RECEIVE_DEALLOC_FAIL);
}
```

Figure 45 (Part 7 of 20). APPC PEM requester example program

```
  /* Tell APPC that transaction program is done */
  ret_code = TP_ENDED(tp_id);

  if (ret_code.p != AP_OK) {
     printf("APPC TP_ENDED failed\n");
     printf("APPC Return Code - Primary  : %#010hx \n", ret_code.p);
     printf("                 - Secondary: %#010lx \n", ret_code.s);
     return(TP_ENDED_FAIL);
  }

  return(PEM_OK);
}
/*****************************************************************************/


/*****************************************************************************/
/* build_request                                                             */
/*                                                                           */
/* Builds a PEM request structure as defined in the CICS-APPC Password       */
/* Expiration Management Guide.                                              */
/*                                                                           */
/* Input:                                                                    */
/*   function_required                                                       */
/*   user_id                                                                 */
/*   password                                                                */
/*   new_password                                                            */
/*                                                                           */
/* Output:                                                                   */
/*   PEM_request                                                             */
/*   User_id, password and new_password will now be in EBCDIC.               */
/*                                                                           */
/*****************************************************************************/

void build_request(function_required,
                   user_id,
                   password,
                   new_password,
                   PEM_request)

func_type          function_required;
char               user_id[],
                   password[],
                   new_password[],
                   PEM_request[];




{
char *sub_field_ptr;     /* Pointer used to build variable length sub fields */
unsigned int length;     /* Sub field length                                 */
```

Figure 45 (Part 8 of 20). APPC PEM requester example program

```
/* Set the GDS Variable ID to 0x1221 */
PEM_request[2] = (char) 0x12;
PEM_request[3] = (char) 0x21;

/* Set the Request field to 0xFF00 for Sign On, 0xFF01 for Change Password */
PEM_request[6] = (char) 0xFF;
if (function_required == SIGN_ON)
   PEM_request[7] = (char) 0x00;
else
   PEM_request[7] = (char) 0x01;

/* Set pointer to start of variable length sub fields */
sub_field_ptr = &PEM_request[8];

/* set user id sub field - length byte, id 0x01, and EBCDIC value */
length = strlen(user_id);
sub_field_ptr[0] = (char) length + 2;
sub_field_ptr[1] = (char) 0x01;
convert_to_EBCDIC(user_id, length);
memcpy(&sub_field_ptr[2], user_id, length);

/* Set pointer to start of next variable length sub field */
sub_field_ptr = &sub_field_ptr[2] + length;

/* set password sub field - length byte, id 0x02, and EBCDIC value */
length = strlen(password);
sub_field_ptr[0] = (char) length + 2;
sub_field_ptr[1] = (char) 0x02;
convert_to_EBCDIC(password, length);
memcpy(&sub_field_ptr[2], password, length);

/* Set pointer to start of next variable length sub field */
sub_field_ptr = &sub_field_ptr[2] + length;

/* If Change Password requested, then include the new password */
if (function_required == CHANGE) {
  /* set new password sub field - length byte, id 0x06, and EBCDIC value */
  length = strlen(new_password);
  sub_field_ptr[0] = (char) length + 2;
  sub_field_ptr[1] = (char) 0x06;
  convert_to_EBCDIC(new_password, length);
  memcpy(&sub_field_ptr[2], new_password, length);

  /* Set pointer to start of next variable length sub field */
  sub_field_ptr = &sub_field_ptr[2] + length;
}

/* Set GDS Variable LL - high order byte to 0x00, low order byte to the */
/* length of the built request, including the length of this LL.        */
PEM_request[0] = (char) 0x00;
PEM_request[1] = (char) (sub_field_ptr - &PEM_request[0]);
```

Figure 45 (Part 9 of 20). APPC PEM requester example program

```
  /* Set nested data structure LL - high order byte to 0x00, low order    */
  /* byte to the length of the GDS variable, minus the length of the GDS  */
  /* LLID (4 bytes)                                                       */
  PEM_request[4] = (char) 0x00;
  PEM_request[5] = (char) (sub_field_ptr - &PEM_request[0] - 4);
}
/*****************************************************************************/



/*****************************************************************************/
/* convert_to_EBCDIC                                                         */
/*                                                                           */
/* Converts the given ASCII string to EBCDIC.                                */
/*                                                                           */
/* input:                                                                    */
/*   ASCII_string                                                            */
/*   length                                                                  */
/*                                                                           */
/* output:                                                                   */
/*   ASCII_string                                                            */
/*                                                                           */
/*****************************************************************************/

void convert_to_EBCDIC(ASCII_string, length)

char ASCII_string[];
unsigned int length;
{
  clear_vcb();

  vcb.cnvt.opcode = SV_CONVERT;
  vcb.cnvt.direction = SV_ASCII_TO_EBCDIC;
  vcb.cnvt.char_set = SV_AE;
  vcb.cnvt.len = length;
  vcb.cnvt.source = vcb.cnvt.target = (unsigned char far *)ASCII_string;

  ACSSVC_C ((long) vcbptr);
}
/*****************************************************************************/


/*****************************************************************************/
/* get_shared_buffer                                                         */
/*                                                                           */
/* APPC requires a data buffer in a shared unnamed segment                   */
/*                                                                           */
/* return value                                                              */
/*   BOOL = {TRUE, FALSE}                                                    */
/*                                                                           */
/*****************************************************************************/
```

Figure 45 (Part 10 of 20). APPC PEM requester example program

```
BOOL get_shared_buffer()
{
  unsigned short dos_rc,
                 selector;


  dos_rc = DOSALLOCSEG(4096, (unsigned far *)&selector, 1);
  if (dos_rc == 0)  {
    FP_OFF(anbptr) = 0;
    FP_SEG(anbptr) = selector;
    return(TRUE);
  }
  else  {
    printf("OS/2 Call error RC = %04d\n",dos_rc);
    anbptr = (unsigned char far *)0;
    return(FALSE);
  }
}
/*****************************************************************************/


/*****************************************************************************/
/* free_shared_buffer                                                        */
/*                                                                           */
/* Free data buffer in the shared unnamed segment                            */
/*                                                                           */
/*****************************************************************************/

void
free_shared_buffer()
{
  unsigned short dos_rc;

  dos_rc = DOSFREESEG(FP_SEG(anbptr));
  if (dos_rc != 0)
     printf("OS/2 Call error RC = %04d\n",dos_rc);
}
/*****************************************************************************/


/*****************************************************************************/
/* TP_STARTED                                                                */
/*                                                                           */
/* Notify APPC that we are requesting resources for a transaction program    */
/* initiated as a result of a local command, rather than an incoming         */
/* allocation request.                                                       */
/*                                                                           */
/* input:                                                                    */
/*   LU_alias                                                                */
/*   tps_tp_name                                                             */
```

Figure 45 (Part 11 of 20). APPC PEM requester example program

```
/*                                                                          */
/* output:                                                                  */
/*   tp_id                                                                  */
/*   tps_tp_name will now be in EBCDIC                                      */
/*                                                                          */
/* return value                                                             */
/*   APPC return code                                                       */
/*                                                                          */
/****************************************************************************/

APPC_rc TP_STARTED(LU_alias,
                   tps_tp_name,
                   tp_id)

char LU_alias[],
     tps_tp_name[],
     tp_id[];
{
  APPC_rc rc;

  /* The TPS TP name must be in EBCDIC */
  convert_to_EBCDIC(tps_tp_name, TP_NAME_LEN);

  clear_vcb();

  vcb.tpstart.opcode = AP_TP_STARTED;

  memcpy (vcb.tpstart.lu_alias, LU_alias, LU_ALIAS_LEN);
  memcpy (vcb.tpstart.tp_name, tps_tp_name, TP_NAME_LEN);

  APPC_C ((long) vcbptr);

  rc.p = vcb.tpstart.primary_rc;
  rc.s = vcb.tpstart.secondary_rc;

  /* If call was sucessful, return the assigned TP id */
  if (rc.p == AP_OK)
    memcpy (tp_id, vcb.tpstart.tp_id, TP_ID_LEN);

  return(rc);
}
/****************************************************************************/


/****************************************************************************/
/* ALLOCATE                                                                 */
/*                                                                          */
/* Allocate a session between the local and partner LUs.  This conversation */
/* will be of type - basic, synclevel - none, security - none, and will     */
/* return control when a session is allocated.                              */
/*                                                                          */
```

Figure 45 (Part 12 of 20). APPC PEM requester example program

```
/* input:                                                                  */
/*   tp_id                                                                 */
/*   partner_LU_alias                                                      */
/*   tp_name                                                               */
/*   mode_name                                                             */
/*                                                                         */
/* output:                                                                 */
/*   conv_id                                                               */
/*   mode_name will now be in EBCDIC                                       */
/*                                                                         */
/* return value                                                            */
/*   APPC return code                                                      */
/*                                                                         */
/*************************************************************************/

APPC_rc
ALLOCATE (tp_id,
          partner_LU_alias,
          tp_name,
          mode_name,
          conv_id)

char      tp_id[],
          partner_LU_alias[],
          tp_name[],
          mode_name[];

unsigned long *conv_id;
{
  APPC_rc rc;



  /* The mode name must be in EBCDIC */
  convert_to_EBCDIC(mode_name, MODE_NAME_LEN);

  clear_vcb();

  vcb.alloc.opcode     = AP_B_ALLOCATE;
  vcb.alloc.opext      = AP_BASIC_CONVERSATION;
  vcb.alloc.sync_level = AP_NONE;
  vcb.alloc.security   = AP_NONE;
  vcb.alloc.rtn_ctl    = AP_WHEN_SESSION_ALLOCATED;

  memcpy (vcb.alloc.tp_id, tp_id, TP_ID_LEN);
  memcpy (vcb.alloc.plu_alias, partner_LU_alias, PARTNER_LU_ALIAS_LEN);
  memcpy (vcb.alloc.tp_name, tp_name, TP_NAME_LEN);
  memcpy (vcb.alloc.mode_name, mode_name, MODE_NAME_LEN);

  APPC_C ((long) vcbptr);

  rc.p = vcb.alloc.primary_rc;
  rc.s = vcb.alloc.secondary_rc;
```

Figure 45 (Part 13 of 20). APPC PEM requester example program

```
  /* if th call was successful, set the conv_id assigned */
  if (rc.p == AP_OK)
    *conv_id = vcb.alloc.conv_id;

  return(rc);
}
/*****************************************************************************/


/*****************************************************************************/
/* SEND_REQUEST                                                              */
/*                                                                           */
/* Send the PEM request to the CICS PEM Server. The send will flush the      */
/* data, and will prepare to receive.                                        */
/*                                                                           */
/* input:                                                                    */
/*   tp_id                                                                   */
/*   conv_id                                                                 */
/*   PEM_request                                                             */
/*                                                                           */
/* return value                                                              */
/*   APPC return code                                                        */
/*                                                                           */
/*****************************************************************************/

APPC_rc
SEND_REQUEST(tp_id,
             conv_id,
             PEM_request)

unsigned long conv_id;
char          tp_id[],
              PEM_request[];
{
  register unsigned msg_len;
  unsigned short send_length;
  APPC_rc rc;

  /* Set send length to low order byte of GDS variable LL */
  send_length = (int) PEM_request[1];
  msg_len = send_length;




  /* Copy PEM request to shared buffer */
  for(;msg_len;)  {
    msg_len--;
    anbptr[msg_len] = PEM_request[msg_len];
  }
```

Figure 45 (Part 14 of 20). APPC PEM requester example program

```
  clear_vcb();

  vcb.send.opcode  = AP_B_SEND_DATA;
  vcb.send.opext   = AP_BASIC_CONVERSATION;
  vcb.send.type    = AP_SEND_DATA_P_TO_R_FLUSH;         /* invite response */
  vcb.send.conv_id = conv_id;
  vcb.send.dlen    = send_length;
  vcb.send.dptr    = anbptr;

  memcpy (vcb.send.tp_id, tp_id, TP_ID_LEN);

  APPC_C ((long) vcbptr);

  rc.p = vcb.send.primary_rc;
  rc.s = vcb.send.secondary_rc;

  return(rc);
}
/*****************************************************************************/


/*****************************************************************************/
/* RECEIVE_RESPONSE                                                          */
/*                                                                           */
/* Receive the PEM response structure, as defined in the CICS-APPC Password  */
/* Expiration Management Guide, from the CICS PEM Server.                    */
/*                                                                           */
/* input:                                                                    */
/*   tp_id                                                                   */
/*   conv_id                                                                 */
/*                                                                           */
/* output:                                                                   */
/*   PEM_response                                                            */
/*                                                                           */
/* return value                                                              */
/*   APPC return code                                                        */
/*                                                                           */
/*****************************************************************************/

APPC_rc
RECEIVE_RESPONSE(tp_id,
                 conv_id,
                 PEM_response)

unsigned long    conv_id;
char             tp_id[],
                 PEM_response[];
{
  unsigned short received_length,
                 max_length;
```

Figure 45 (Part 15 of 20). APPC PEM requester example program

```
  APPC_rc rc;

  max_length = PEM_RESP_LEN;

  clear_vcb();

  vcb.rcv_wait.opcode     = AP_B_RECEIVE_AND_WAIT;
  vcb.rcv_wait.opext      = AP_BASIC_CONVERSATION;
  vcb.rcv_wait.rtn_status = AP_NO;
  vcb.rcv_wait.conv_id    = conv_id;
  vcb.rcv_wait.max_len    = max_length;
  vcb.rcv_wait.dptr       = anbptr;


  memcpy (vcb.rcv_wait.tp_id, tp_id, TP_ID_LEN);

  APPC_C ((long) vcbptr);

  rc.p = vcb.rcv_wait.primary_rc;
  rc.s = vcb.rcv_wait.secondary_rc;

  /* If response received ok, then copy data from shared buffer */
  if (rc.p == AP_OK) {
    received_length = vcb.rcv_wait.dlen;

    for(;received_length;) {
      received_length--;
      PEM_response[received_length] = anbptr[received_length];
    }
  }

  return(rc);
}
/*****************************************************************************/


/*****************************************************************************/
/* RECEIVE_DEALLOC                                                           */
/*                                                                           */
/* Receive deallocation notification (AP_DEALLOC_NORMAL return code) from the*/
/* CICS PEM Server.  If this is not received, then termination of the        */
/* conversation is forced by issuing a deallocate abend.                     */
/*                                                                           */
/* input:                                                                    */
/*   tp_id                                                                   */
/*   conv_id                                                                 */
/*                                                                           */
/* return value                                                              */
/*   APPC return code                                                        */
/*                                                                           */
/*****************************************************************************/
```

Figure 45 (Part 16 of 20). APPC PEM requester example program

```
APPC_rc
RECEIVE_DEALLOC(tp_id,
                conv_id)

unsigned long   conv_id;
char            tp_id[];
{
  APPC_rc rc;

  clear_vcb();

  vcb.rcv_wait.opcode     = AP_B_RECEIVE_AND_WAIT;
  vcb.rcv_wait.opext      = AP_BASIC_CONVERSATION;
  vcb.rcv_wait.rtn_status = AP_NO;
  vcb.rcv_wait.conv_id    = conv_id;
  vcb.rcv_wait.max_len    = 50;     /* Arbitrary length */
  vcb.rcv_wait.dptr       = anbptr;

  memcpy (vcb.rcv_wait.tp_id, tp_id, TP_ID_LEN);

  APPC_C ((long) vcbptr);

  rc.p = vcb.rcv_wait.primary_rc;
  rc.s = vcb.rcv_wait.secondary_rc;


  /* If conversation has not been terminated by the CICS PEM Server, then */
  /* we issue a deallocate abend to force termination.                    */
  if (!(rc.p == AP_DEALLOC_NORMAL |
        rc.p == AP_DEALLOC_ABEND_PROG |
        rc.p == AP_DEALLOC_ABEND_SVC |
        rc.p == AP_DEALLOC_ABEND_TIMER |
        rc.p == AP_CONV_FAILURE_NO_RETRY |
        rc.p == AP_CONV_FAILURE_RETRY)) {

    clear_vcb();

    vcb.dealloc.opcode       = AP_B_DEALLOCATE;
    vcb.dealloc.opext        = AP_BASIC_CONVERSATION;
    vcb.dealloc.dealloc_type = AP_ABEND;
    vcb.dealloc.conv_id      = conv_id;

    memcpy (vcb.dealloc.tp_id, tp_id, TP_ID_LEN);

    APPC_C ((long) vcbptr);
  }

  return(rc);
}
/*****************************************************************************/
```

Figure 45 (Part 17 of 20). APPC PEM requester example program

```
/*****************************************************************************/
/* DO_TP_ENDED                                                               */
/*                                                                           */
/* Notify APPC of the end of this transaction program.                       */
/*                                                                           */
/* input:                                                                    */
/*   tp_id                                                                   */
/*                                                                           */
/* return value                                                              */
/*   APPC return code                                                        */
/*                                                                           */
/*****************************************************************************/

APPC_rc
TP_ENDED(tp_id)

char     tp_id[];
{
  APPC_rc rc;

  clear_vcb();

  vcb.tpend.opcode = AP_TP_ENDED;

  memcpy (vcb.tpend.tp_id, tp_id, TP_ID_LEN);

  APPC_C ( (long) vcbptr );

  rc.p = vcb.tpend.primary_rc;
  rc.s = vcb.tpend.secondary_rc;

  return(rc);
}
/*****************************************************************************/

/*****************************************************************************/
/* display_PEM_response                                                      */
/*                                                                           */
/* Displays the date returned in the PEM response structure.                 */
/*                                                                           */
/* input:                                                                    */
/*   PEM_response                                                            */
/*                                                                           */
/*****************************************************************************/



void display_PEM_response(PEM_response)
```

Figure 45 (Part 18 of 20). APPC PEM requester example program

```
char PEM_response[];
{
  unsigned int index,
               length,
               sub_field_len,
               i;

  length = (unsigned int) PEM_response[1];

  /* Display GDS LLID */
  printf("GDS LLID\n");

  for(index = 0; index <= 3; index++)
    printf("%02hx ", (unsigned char) PEM_response[index]);
    printf("\n\n");

  /* Display Sign-On Reply LLID */
  printf("Sign-On Reply LLID\n");

  for(; index <= 7; index++)
    printf("%02hx ", (unsigned char) PEM_response[index]);
  printf("\n\n");

  /* Display Sign_on Completion Status Subfield */
  printf("Sign-On Completion Status Subfield\n");

  for(; index <= 10; index++)
    printf("%02hx ", (unsigned char) PEM_response[index]);
  printf("\n\n");

  while (index < length) {
    sub_field_len = (unsigned int) PEM_response[index];

    /* Switch on Sub Field ID */
    switch((unsigned int) PEM_response[index + 1]) {
    case 0x01 : printf("Sign-On Request Formating Error Subfield\n");

                for(i = 0; i < sub_field_len; i++)
                  printf("%02hx ", (unsigned char) PEM_response[index+i]);
                printf("\n\n");
                index = index + i;
                break;
    case 0x02 : printf("Date & Time of Current Successful Sign-On Subfield\n");

                for(i = 0; i < sub_field_len; i++)
                  printf("%02hx ", (unsigned char) PEM_response[index+i]);
                printf("\n\n");
                index = index + i;
                break;
    case 0x03 : printf("Date & Time of Last Successful Sign-On Subfield\n");
```

Figure 45 (Part 19 of 20). APPC PEM requester example program

```
                for(i = 0; i < sub_field_len; i++)
                  printf("%02hx ", (unsigned char) PEM_response[index+i]);
                printf("\n\n");
                index = index + i;
                break;
    case 0x04 : printf("Date & Time Password will Expire Subfield\n");

                for(i = 0; i < sub_field_len; i++)
                  printf("%02hx ", (unsigned char) PEM_response[index+i]);
                printf("\n\n");
                index = index + i;
                break;
    case 0x05 : printf("Revoke Count Subfield\n");



                for(i = 0; i < sub_field_len; i++)
                  printf("%02hx ", (unsigned char) PEM_response[index+i]);
                printf("\n\n");
                index = index + i;
                break;
    default :   printf("Invalid Subfield ID\n");
                index = PEM_RESP_LEN;
    }
    }
    }
```

Figure 45 (Part 20 of 20). APPC PEM requester example program

Appendix B. National Language

This appendix contains the language codes with which the user can specify a preferred language if that language is defined in their CICS system.

A language request is coded as follows:

```
ALTUSER userid LANGUAGE(PRIMARY(language-code) SECONDARY(language-code))
```

For PRIMARY or SECONDARY, you can specify one of the language codes under "IBM code" in Table 41 on page 288.

CICS attempts to use the PRIMARY language for a user if it corresponds to a language suffix in the NATLANG system initialization parameter. Otherwise it attempts to use the SECONDARY language. If neither the PRIMARY nor the SECONDARY language corresponds to a NATLANG value, the language must be provided from elsewhere. See "Obtaining CICS-related data at signon" on page 72.

Note: CICS ignores the RACF default national language defined by the command:

```
SETROPTS LANGUAGE(PRIMARY(...) SECONDARY(...))
```

Only the languages listed in Table 41 on page 288 are available for use in CICS. Languages other than ENU and JPN are available only if you provide translated message tables for them, using the message editing utility program, DFHMEU, and then specify the CICS language suffix in the NATLANG system initialization parameter. See the *CICS/ESA Operations and Utilities Guide* for information on creating translated message tables.

Table 41. CICS language suffixes

Suffix	IBM Code	Language name
A	ENG	United Kingdom English
B	PTB	Brazilian Portuguese
C	CHS	Simplified Chinese
D	DAN	Danish
E	ENU	US English
F	FRA	French
G	DEU	German
H	KOR	Korean
I	ITA	Italian
J	ISL	Icelandic
K	JPN	Japanese
L	BGR	Bulgarian
M	MKD	Macedonian
N	NOR	Norwegian
O	ELL	Greek
P	PTG	Portuguese
Q	ARA	Arabic
R	RUS	Russian
S	ESP	Spanish
T	CHT	Traditional Chinese
U	UKR	Ukrainian
V	SVE	Swedish
W	FIN	Finnish
X	HEB	Hebrew
Y	SHC	Serbo-Croatian (Cyrillic)
Z	THA	Thai
1	BEL	Byelorussian
2	CSY	Czech
3	HRV	Croatian
4	HUN	Hungarian
5	PLK	Polish
6	ROM	Romanian
7	SHL	Serbo-Croatian (Latin)
8	TRK	Turkish
9	NLD	Dutch

Appendix C. API command and resource check cross reference

This appendix provides a complete API command and resource check cross reference.

Table 42 (Page 1 of 5). API command resource check cross reference

EXEC CICS COMMAND	Resource Check			Check class=XCMD	
	Class	**Access**	**Resource**	**Access**	**Resource**
ABEND					
ACQUIRE				UPDATE	TERMINAL
ADDRESS					
ALLOCATE					
ASKTIME					
ASSIGN					
BIF DEEDIT					
BUILD ATTACH					
CANCEL (see note 7 on page 294)	XPCT	READ	transid		
CHANGE PASSWORD					
CHANGE TASK					
COLLECT STATISTICS				READ	STATISTICS
COLLECT FILE	XFCT	READ	file	READ	STATISTICS
COLLECT JOURNALNUM	XJCT	READ	journal	READ	STATISTICS
COLLECT PROGRAM	XPPT	READ	program	READ	STATISTICS
COLLECT TDQUEUE	XPCT	READ	tdqueue	READ	STATISTICS
COLLECT TRANSACTION	XDCT	READ	transid	READ	STATISTICS
CONNECT PROCESS					
CONVERSE					
DELAY					
DELETE	XFCT	UPDATE	file		
DELETEQ TD	XDCT	UPDATE	tdqueue		
DELETEQ TS (see note 1 on page 293)	XTST	UPDATE	tsqueue		
DEQ					
DISABLE PROGRAM	XPPT	UPDATE	program	UPDATE	EXITPROGRAM
DISCARD AUTINSTMODEL				UPDATE	AUTINSTMODEL
DISCARD FILE	XFCT	UPDATE	file	UPDATE	FILE

Table 42 (Page 2 of 5). API command resource check cross reference					
DISCARD PARTNER				UPDATE	PARTNER
DISCARD PROFILE				UPDATE	PROFILE
DISCARD PROGRAM	XPPT	UPDATE	program	UPDATE	PROGRAM
DISCARD TRANCLASS				UPDATE	TCLASS
DISCARD TRANSACTION	XPCT	UPDATE	transid	UPDATE	TRANSACTION
DUMP TRANSACTION					
ENABLE PROGRAM	XPPT	UPDATE	program	UPDATE	EXITPROGRAM
ENDBR (see note 2 on page 293)					
ENQ					
ENTER TRACENUM					
EXTRACT					
EXTRACT EXIT	XPPT	READ	program	UPDATE	EXITPROGRAM
FORMATTIME					
FREE					
FREEMAIN					
GDS					
GETMAIN					
HANDLE ABEND PROGRAM	XPPT	READ	program		
HANDLE AID					
HANDLE CONDITION					
IGNORE CONDITION					
INQUIRE AUTINSTMODEL				READ	AUTINSTMODEL
INQUIRE AUTOINSTALL				READ	AUTOINSTALL
INQUIRE CONNECTION				READ	CONNECTION
INQUIRE DSNAME				READ	DSNAME
INQUIRE DUMPDS				READ	DUMPDS
INQUIRE EXITPROGRAM	XPPT	READ	program	READ	EXITPROGRAM
INQUIRE FILE	XFCT	READ	file	READ	FILE
INQUIRE IRC				READ	IRC
INQUIRE JOURNALNUM	XJCT	READ	program	READ	EXITPROGRAM
INQUIRE MODENAME				READ	MODENAME
INQUIRE MONITOR				READ	MONITOR
INQUIRE NETNAME				READ	TERMINAL
INQUIRE PARTNER				READ	PARTNER
INQUIRE PROFILE				READ	PROFILE

Table 42 (Page 3 of 5). API command resource check cross reference

INQUIRE PROGRAM	XPPT	READ	program	READ	PROGRAM
INQUIRE REQID (see note 6 on page 293)	XPCT	READ	transid	READ	REQID
INQUIRE STATISTICS				READ	STATISTICS
INQUIRE STORAGE				READ	STORAGE
INQUIRE SYSDUMPCODE				READ	SYSDUMPCODE
INQUIRE SYSTEM				READ	SYSTEM
INQUIRE TASK				READ	TASK
INQUIRE TCLASS				READ	TCLASS
INQUIRE TDQUEUE	XDCT	READ	tdqueue	READ	TDQUEUE
INQUIRE TERMINAL				READ.	TERMINAL
TRACEDEST				READ	TRACEDEST
INQUIRE TRACEFLAG				READ	TRACEFLAG
INQUIRE TRACETYPE				READ	TRACETYPE
INQUIRE TRANCLASS				READ	TRANCLASS
INQUIRE TRANDUMPCODE				READ	DUMPCODE
INQUIRE TRANSACTION	XPCT	READ	program	READ	TRANSACTION
INQUIRE TSQUEUE (see note 1 on page 293)	XTST	READ	tsqueue	READ	TSQUEUE
INQUIRE VOLUME				READ	VOLUME
INQUIRE VTAM				READ	VTAM
ISSUE					
LINK	XPPT	READ	program		
LOAD	XPPT	READ	program		
MONITOR					
PERFORM DUMP				UPDATE	DUMP
PERFORM RESETTIME				UPDATE	RESETTIME
PERFORM SECURITY				UPDATE	SECURITY
PERFORM SHUTDOWN				UPDATE	SHUTDOWN
PERFORM STATISTICS				UPDATE	STATISTICS
POINT					
POP HANDLE					
POST					
PURGE MESSAGE					
PUSH HANDLE					

Table 42 (Page 4 of 5). API command resource check cross reference					
QUERY SECURITY (see note 3 on page 293)					
READ	XFCT	READ	file		
READ NEXT (see note 2 on page 293)					
READ PREV (see note 2 on page 293)					
READQ TD	XDCT	UPDATE	tdqueue		
READQ TS (see note 1 on page 293)	XTST	READ	tsqueue		
RECEIVE					
RELEASE	XPPT	READ	program		
RESETBR (see note 2 on page 293)					
RESYNC ENTRYNAME				UPDATE	EXITPROGRAM
RETRIEVE					
RETURN					
REWRITE	XFCT	UPDATE	file		
ROUTE					
SEND					
SET AUTOINSTALL				UPDATE	AUTOINSTALL
SET CONNECTION				UPDATE	CONNECTION
SET DSNAME				UPDATE	DSNAME
SET DUMPDS				UPDATE	DUMPDS
SET FILE	XFCT	UPDATE	file	UPDATE	FILE
SET IRC				UPDATE	IRC
SET JOURNALNUM	XJCT	UPDATE	journal	UPDATE	JOURNALNUM
SET MODENAME				UPDATE	MODENAME
SET MONITOR				UPDATE	MONITOR
SET NETNAME				UPDATE	TERMINAL
SET PROGRAM	XPPT	UPDATE	program	UPDATE	PROGRAM
SET STATISTICS				UPDATE	STATISTICS
SET SYSDUMPCODE				UPDATE	SYSDUMPCODE
SET SYSTEM				UPDATE	SYSTEM
SET TASK				UPDATE	TASK
SET TCLASS				UPDATE	TCLASS
SET TDQUEUE (see note 5 on page 293)	XDCT	UPDATE	tdqueue	UPDATE	TDQUEUE
SET TERMINAL				UPDATE	TERMINAL
SET TRACEDEST				UPDATE	TRACEDEST
SET TRACEFLAG				UPDATE	TRACEFLAG

Table 42 (Page 5 of 5). API command resource check cross reference					
SET TRACETYPE				UPDATE	TRACETYPE
SET TRANCLASS				UPDATE	TCLASS
SET TRANDUMPCODE				UPDATE	TRANDUMPCODE
SET TRANSACTION	XPCT	UPDATE	transid	UPDATE	TRANSACTION
SET VOLUME				UPDATE	VOLUME
SET VTAM				UPDATE	VTAM
SIGNOFF					
SIGNON					
SPOOLCLOSE					
SPOOLOPEN					
SPOOLREAD					
SPOOLWRITE					
START (see note 4 on page 293)	XPCT	READ	transid		
STARTBR	XFCT	READ	file		
SUSPEND					
SYNCPOINT					
UNLOCK					
VERIFY PASSWORD					
WAIT					
WAIT JOURNALNUM	XJCT	READ	journal		
WAITCICS					
WRITE	XFCT	UPDATE	file		
WRITE JOURNALNUM	XJCT	UPDATE	journal		
WRITE OPERATOR					
WRITEQ TD	XDCT	UPDATE	tdqueue		
WRITEQ TS (see note 1 on page 293)	XTST	UPDATE	tsqueue		
XCTL	XPPT	READ	program		

Notes:

1. A Security check is performed only if a DFHTST TYPE=SECURITY macro has been coded in the TST with a name that matches the tsname.
2. No security check is performed, because the STARTBR command must be issued before this command and security check is issued on STARTBR command.
3. Using query security does not issue resource or command checks, but it can cause them to be issued.
4. A start surrogate check can also occur.
5. An install surrogate check can also occur.
6. The resource check for the transid is only done if the reqid is associated with a transaction.

7. CANCEL does two checks. One is done against the transaction specified on the CANCEL command, and the other is done against the transaction associated with the reqid you are canceling (where applicable).

Glossary

For definitions of terms not in this glossary, see the *IBM Dictionary of Computing*, New York: McGraw-Hill, 1994.

access. The ability to obtain the use of a protected resource.

access authority. An authority that relates to a request for a type of access to protected resources. In RACF, the access authorities are: NONE, EXECUTE, READ, UPDATE, CONTROL, and ALTER.

access list. Synonym for standard access list. See also **conditional access list**.

ACEE (accessor environment element). A description of the current user including userid, current connect group, user attributes, and group authorities. An ACEE is constructed during user identification and verification.

AOR (application-owning region). A CICS address space whose primary purpose is to manage application programs. It receives transaction routed requests from a terminal-owning region (TOR). It may also contain file-related resources in a system that does not have a data-owning region (DOR). See also **DOR (data-owning region)** and **TOR (terminal-owning region)**.

APPC (advanced program-to-program communication). The implementation of the LU6.2 architecture. It is one of the ISC protocols that CICS uses.

attribute. See **user attribute**.

authority. The right to access objects, resources, or functions. See **access authority**, **group authority**, and **class authority**.

authorization checking. The action of determining whether a user is permitted access to a protected resource. RACF performs authorization checking as a result of a RACHECK or FRACHECK request.

base segment. Synonym for RACF segment.

bind. Refers to the SNA BIND command used to establish SNA sessions between systems, and to the CICS connection request used to establish multiregion operation (MRO) sessions for interregion communication. See also **bind-time security**.

bind-time security. In LU6.2 and MRO, the level of security applied when a request to establish a session is received from, or sent to, a remote system. Used to verify that the remote system is really the system it claims to be. Also known, in SNA terms, as **session security**. See also **bind**, **link security** and **user security**.

BWO (backup while open). A means of taking backups of VSAM files that CICS is concurrently updating.

category. See **security category**.

CDT (class descriptor table). A RACF table consisting of an entry for each class except the USER, GROUP, and DATASET classes. The table is generated by invoking the ICHERCDE macro once for each class.

CEDF. A CICS-supplied transaction for initiating the execution diagnostic facility program (DFHEDFP). It allows CICS commands issued by an application program to be traced online.

CICS default userid. The userid assigned to a terminal before the user signs on to CICS, and after the user signs off.

CICS region userid. The userid assigned to a CICS region at CICS initialization. It is specified **either** in the RACF started procedures table when CICS is started as a started task, **or** on the USER parameter of the JOB statement when CICS is started as a job.

CICS segment. The portion of a RACF user profile containing data for CICS.

class. A collection of RACF-defined entities that is, users, groups, or resources (including general resources) that have similar characteristics. The class names are USER, GROUP, DATASET, and the classes that are defined in the class descriptor table. See also **general resource** and **CDT (class descriptor table)**.

class descriptor. RACF-supplied control block for all the classes in the class descriptor table (which are all the classes except the USER, GROUP, and DATASET classes). See **CDT (class descriptor table)**.

CLAUTH (class authority). An authority that allows a user to define RACF profiles in a class defined in the class descriptor table. A user can have class authority to one or more classes.

conditional access list. An access list within a resource profile that associates a condition with a userid or group id and the corresponding access authority. If a user does not otherwise have the requested access, a conditional access list entry can allow access if the specified condition is true. For example, for program

access to data sets, the condition is that the user must be executing the program specified in the access list. See also **access list**. and **standard access list**.

cross-memory services. services that apply to more than one private address space. Cross-memory services use the MVS common system area (CSA) storage for control blocks, not for data transfer. MVS requires that an address space using cross-memory services is non-swappable.

current connect group. The group with which a user is associated, for access checking purposes, during a terminal session or batch job. If a user does not specify a group on CICS signon, the user's default group is used.

A user may specify a group name when signing on. In this case, the group name specified becomes the current group

data security. The protection of data from unauthorized disclosure, modification, or destruction, whether accidental or intentional.

data set profile. A profile that provides RACF protection for one or more data sets. The information in the profile can include the data set profile name, profile owner, universal access authority, access list, and other data. See **discrete profile** and **generic profile**.

default group. In RACF, the group specified in a user profile that is the default current connect group.

delegation. The act of giving other users or groups authorities to perform RACF operations.

discrete profile. A resource profile that can provide RACF protection for only a single resource. For example, a discrete profile can protect only a single data set or minidisk.

DOR (data-owning region). A CICS address space whose primary purpose is to manage files and databases. Also known as a file-owning region (FOR). See also **AOR (application-owning region)** and **TOR (terminal-owning region)**.

dynamic parse. A method of parsing TSO commands according to syntax given in an external file.

EDF (execution diagnostic facility). A mechanism for debugging CICS transactions by displaying the results of CICS commands.

entity. A user, group, or resource (for example, a CICS resource) that is defined to RACF.

entity class. A resource class that contains individual resources rather than groups of resources.

equivalent systems. CICS regions having identical region userids. In regions connected by MRO, the link security userid can be the same as the userid of the region being connected to. In LU6.1 and LU6.2, the link security userid has to be the same as the userid belonging to the CICS region.

EXCI (External CICS interface). An application programming interface (API) that enables an MVS client program to call to call a program running in a CICS/ESA 4.1 system, and to pass and receive data using a communications area. The CICS program is invoked as if linked-to by another CICS program via a distributed program link (DPL) request.

explicit signon. Signon initiated through EXEC CICS SIGNON.

explicit signoff. Sign-off initiated through EXEC CICS SIGNOFF.

field-level access checking. The RACF facility by which a security administrator can control access to fields or segments in a RACF profile.

FOR (file-owning region). A CICS address-space whose primary purpose is to manage CICS files and data tables, especially shared data tables.

FRACHECK request. The issuing of the FRACHECK macro or the RACROUTE macro with REQUEST=FASTAUTH specified. The primary function of a FRACHECK request is to check a user's authorization to a RACF-protected resource or function. A FRACHECK request uses only in-storage profiles for faster performance. See also **authorization checking**.

general resource. Any system resource, other than an MVS data set, that is defined in the RACF class descriptor table (CDT). On MVS, general resources include DASD volumes, tape volumes, load modules, terminals, IMS and CICS transactions and other CICS resources, and installation-defined resource classes. See also **class**.

general resource profile. A profile that provides RACF protection for one or more general resources. The information in the profile can include the general resource profile name, profile owner, universal access authority, access list, and other data.

generic profile. A resource profile that can provide RACF protection for one or more resources. The resources protected by a generic profile have similar names and identical security requirements. For example, a generic data set profile can protect one or more data sets.

global access checking. The ability to allow an installation to establish an in-storage table of default values for authorization levels for selected resources.

RACF refers to this table before performing normal RACHECK processing, and grants the request without performing a RACHECK if the requested access authority does not exceed the global value. Global access checking can grant the user access to the resource, but it cannot deny access.

group. A collection of RACF-defined users who can share access authorities for protected resources.

group authority. An authority that describes which functions a user can perform in a group. The group authorities are USE, CREATE, CONNECT, and JOIN.

group data set. On MVS, a RACF-protected data set in which either the high-level qualifier of the data set name or the qualifier supplied by an installation exit routine is a RACF group name.

groupid (group identifier). A string of one to eight characters that identifies a group to RACF. The first character must be A through Z, #, $, or @. The rest can be A through Z, #, $, @, or 0 through 9.

group profile. A profile that defines a group. The information in the profile includes the group name, profile owner, and users in the group.

group terminal option. A RACF function that allows users within a group to log on only from those terminals to which they have been specifically authorized.

group-related user attribute. A user attribute assigned at the group level that allows the user to control the resource, group, and user profiles associated with the group and its subgroups. Some of the group-related user attributes are group-SPECIAL, group-AUDITOR, and group-OPERATIONS.

implicit signon. Signon other than by means of CESN, CESF or EXEC CICS SIGNON

implicit signoff. Signoff other than by means of CESN, CESF or EXEC CICS SIGNOFF

intersystem communication (ISC). A protocol for communication between CICS regions using telecommunication.

LANGUAGE segment. The portion of a RACF profile containing information about the national language in which the user receives messages.

link security. Link security limits one system's authorization to attach transactions and access resources in another. It works by signing on each end of a session to RACF when the session is bound. Each half-session then has the access requirements of a user, whose user profile is applied when a transaction is attached and whenever that transaction accesses a protected resource. See also **bind-time security**.

list-of-groups checking. A RACF option that allows a user to access all resources available to all groups of which the user is a member, regardless of the user's current connect group. For any particular resource, RACF allows access based on the highest access among the groups of which the user is a member.

logging. The recording of data about specific events.

logon. In CICS, the act of establishing a session with VTAM. Contrast with **sign-on**.

multiregion operation (MRO). Communication between CICS systems in the same processor without the use of SNA networking facilities.

MVS. Multiple virtual storage. Implies MVS/370, MVS/XA, or MVS/ESA.

OIDCARD (operator identification card). A small card with a magnetic stripe encoded with unique characters and used to verify the identity of a terminal operator to RACF.

owner. The user or group who creates a profile, or is named the owner of a profile. The owner can modify, list, or delete the profile.

PassTicket. A password substitute that can be used only once and is only valid for a 10 minute interval between creation and use.

password. In computer security, a string of characters known to the computer system and a user, who must specify it to gain full or limited access to a system and to the data stored within it. In RACF, the password is used to verify the identity of the user.

Port of Entry (POE). The name and type of device from which a user signs on. CICS recognizes only TERMINALS and CONSOLES.

POSIT. A keyword in the ICHERCDE macro that determines the position of a resource class in the RACF class descriptor table (CDT). All classes with the same POSIT value are controlled together by the SETROPTS command.

preset terminal security. When a CICS region is started, the signing on of selected terminals as "users" whose userids are permanently associated with the terminal. Persons using these terminals have the authorizations given to the terminals.

profile. Data that describes the significant characteristics of a user, a group of users, or one or more computer resources. See also **connect profile**, **data set profile**, **discrete profile**, **directory profile**, **file profile**, **general resource profile**, **generic profile**, **group profile**, and **user profile**.

profile list. A list of profiles indexed by class (for general resources) or by the high-level qualifier (for DATASET profiles) and built in storage by the RACF routines.

protected resource. A resource that is defined to RACF for the purpose of controlling access to the resource. This book is primarily concerned with CICS resources. Some other resources that can be protected by RACF include DASD and tape data sets, DASD volumes, tape volumes, terminals, IMS transactions, IMS transaction groups, and any other resources defined in the class descriptor table.

RACF (Resource Access Control Facility). An IBM licensed product that provides for access control by identifying and verifying users to the system, authorizing access to protected resources, logging detected authorized and unauthorized attempts to enter the system, and logging detected accesses to protected resources.

RACF database. A collection of interrelated or independent data items stored together without unnecessary redundancy, to serve the Resource Access Control Facility (RACF).

RACF-protected. Pertaining to a resource that has either a discrete profile or an applicable generic profile. A data set that is RACF-protected by a discrete profile must also be RACF-indicated.

RACF report writer. A RACF function that produces reports on system use and resource use from information found in the RACF SMF records.

RACF segment. The portion of a RACF profile that contains basic information needed to define a user, group, or resource to RACF. Also called base segment.

RACHECK request. The issuing of the RACHECK macro or the RACROUTE macro with REQUEST=AUTH specified. The primary function of a RACHECK request is to check a user's authorization to a RACF-protected resource or function. See also **authorization checking**.

RACINIT request. The issuing of the RACINIT macro or the RACROUTE macro with REQUEST=VERIFY or REQUEST=VERIFYX specified. A RACINIT request is used to verify the authority of a user to enter work into the system.

RACROUTE macro. An assembler macro that provides an means of calling RACF to provide security functions. See also **FRACHECK request**, **RACHECK** request, and **RACINIT** request.

remote user. A user from another region.

resource class. See **resource group class**.

resource group class. A RACF class in which resource group profiles can be defined. A resource group class is related to another class, sometimes called a "member class". For example, resource group class GTERMINL is related to resource member class TERMINAL. See also **resource group profile**.

resource member class. See **resource group class**.

resource group profile. A general resource profile in a resource group class. A resource group profile can provide RACF protection for one or more resources with **unlike** names. See also **resource group class**.

resource profile. A profile that provides RACF protection for one or more resources. User, group, and connect profiles are not resource profiles. The information in a resource profile can include the data set profile name, profile owner, universal access authority, access list, and other data. Resource profiles can be discrete profiles or generic profiles. See **discrete profile** and **generic profile**.

SAF (MVS System Authorization Facility). An MVS interface invoked by CICS to communicate with an external security manager, such as RACF.

scoping. A mechanism for controlling multiple signon of the same userid to one or more CICS regions.

segment. A portion of RACF profile containing logically related fields. See CICS segment, LANGUAGE segment, SESSION segment, and RACF segment.

session security, SNA. See **bind-time security**.

SESSION segment. The portion of a RACF profile in the APPCLU class containing data used to control the establishment of sessions between logical units under LU 6.2.

sign-on. In CICS, to perform user identification and verification. Contrast with **logon**.

SIT (system initialization table). A table containing user-specified data that controls a system initialization process.

SMF. System Management Facility, a component of MVS for recording management data.

SNSCOPE. See scoping.

SP commands. The subset of CICS API commands (COLLECT, DISCARD, INQUIRE, PERFORM, and SET) that require the special CICS translator option, SP, and for which command security checking can be done.

standard access list. A list within a profile of all authorized users and their access authorities. Synonymous with access list. See also **conditional access list**.

started transactions. A CICS transaction initiated by a terminal user can start other transactions by means of a CICS START command. Transactions started in this way are known as **started transactions**.

surrogate terminal. A logical representation of a terminal that is physically connected to another CICS region.

surrogate user. A user who is authorized to start work on behalf of another user. A surrogate user has authority to submit jobs for, or start CICS transactions for, or associate CICS resources with, the other user without needing to supply that user's password.

sysplex. A systems complex, consisting of multiple MVS images coupled together by hardware elements and software services. When multiple MVS images are coupled using XCF, which provides the services to form a sysplex, they can be viewed as a single entity.

Systems Network Architecture (SNA). The description of the logical structure, formats, protocols, and operational sequences for transmitting information units through, and controlling the configuration and operation of, networks. The structure of SNA allows the end users to be independent of, and unaffected by, the specific facilities used for information exchange.

TDQ. System messages that CICS produces are commonly sent to Transient Data Queues, either intrapartition or extrapartition. For more information about TDQ, see the *CICS/ESA Resource Definition Guide*.

TOR (terminal-owning region). A CICS address space whose primary purpose is to manage terminals. See also **AOR (application-owning region)** and **DOR (data-owning region)**.

UACC (universal access authority). The default access authority that applies to a resource if the user or group is not specifically permitted access to the resource. The universal access authority can be any of the access authorities.

user. A person who requires the services of a computing system.

user attribute. In RACF, the extraordinary privileges, restrictions, and processing environments assigned to a user. The user attributes are SPECIAL, AUDITOR, CLAUTH, OPERATIONS, GRPACC, ADSP, and REVOKE. In CICS, the attributes of a user obtained from the CICS segment of the user profile, namely OPCLASS, OPIDENT, OPPRTY, TIMEOUT, and XRFSOFF.

user data set. On MVS, a data set defined to RACF in which either the high-level qualifier of the data set name or the qualifier supplied by an installation exit routine is a RACF userid.

user identification and verification. The acts of identifying and verifying a RACF-defined user to the system during logon or batch job processing.

user name. One to twenty alphanumeric characters that represent a RACF-defined user.

user profile. A description of a RACF-defined user that includes the userid, user name, default group name, password, profile owner, user attributes, and other information. A user profile can include information for subsystems such as CICS, DFP, and TSO. See also **CICS segment**.

user security. The facilities for, or action of, verifying that a user is authorized (1) to run a transaction and (2) to access the resources and use the commands that a transaction invokes.

userid (user identifier). A string of characters that uniquely identifies a user to a system. On CICS, a userid is one to eight alphanumeric characters. On TSO, userids cannot exceed seven characters and must begin with an alphabetic, #, $, or @ character.

verification. The act of confirming that a user is eligible to use a RACF-defined userid. RACF identifies the user by the userid and verifies the user by the password (or PassTicket) or operator identification card (OIDCARD) supplied during signon processing, or the password supplied on a batch JOB statement.

VLF. Virtual Lookaside Facility, a service offered by MVS that makes it possible to create and retrieve named data objects, such as members of a partitioned data set, in virtual storage. VLF uses data spaces to keep large amounts of data in virtual storage.

Xname resource classes. The general resource classes that CICS uses based on X**name** system initialization parameters. For example, if XTRAN=YES is specified, TCICSTRN and GCICSTRN are used.

Xname system initialization parameters. In this book, we refer to the CICS system initialization parameters: XAPPC, XCMD, XDCT, XFCT, XJCT, XPCT, XPPT, XPSB, XTRAN, and XTST, which are related to resource security checking, as the X**name** parameters.

XRF (extended recovery facility). A software function that minimizes the impact of various system failures on

users by transferring activity to an alternate system in the same MVS image or a different one.

Index

Special Characters

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Sending your comments to IBM

CICS/ESA

CICS-RACF Security Guide

SC33-1185-02

If you especially like or dislike anything about this book, please use one of the methods listed below to send your comments to IBM.

Feel free to comment on what you regard as specific errors or omissions, and on the accuracy, organization, subject matter, or completeness of this book. Please limit your comments to the information in this book and the way in which the information is presented.

To request additional publications, or to ask questions or make comments about the functions of IBM products or systems, you should talk to your IBM representative or to your IBM authorized remarketer.

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

You can send your comments to IBM in any of the following ways:

- By mail, use the Readers' Comment Form
- By fax:
  - From outside the U.K., after your international access code use 44 962 870229 (after 16 April 1995, use 44 1962 870229)
  - From within the U.K., use 0962 870229 (after 16 April 1995, use 01962 870229)
- Electronically, use the appropriate network ID:
  - IBM Mail Exchange: GBIBM2Q9 at IBMMAIL
  - IBMLink: WINVMJ(IDRCF)
  - Internet: idrcf@winvmj.vnet.ibm.com

Whichever you use, ensure that you include:

- The publication number and title
- The page number or topic to which your comment applies
- Your name and address/telephone number/fax number/network ID.

Readers' Comments

CICS/ESA

CICS-RACF Security Guide

SC33-1185-02

Use this form to tell us what you think about this manual. If you have found errors in it, or if you want to express your opinion about it (such as organization, subject matter, appearance) or make suggestions for improvement, this is the form to use.

To request additional publications, or to ask questions or make comments about the functions of IBM products or systems, you should talk to your IBM representative or to your IBM authorized remarketer. This form is provided for comments about the information in this manual and the way it is presented.

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

Be sure to print your name and address below if you would like a reply.

Name	Address
Company or Organization	
Telephone	Email

CICS/ESA

CICS/ESA CICS-RACF Security Guide
SC33-1185-02

You can send your comments POST FREE on this form from any one of these countries:

Australia	Finland	Iceland	Netherlands	Singapore	United States
Belgium	France	Israel	New Zealand	Spain	of America
Bermuda	Germany	Italy	Norway	Sweden	
Cyprus	Greece	Luxembourg	Portugal	Switzerland	
Denmark	Hong Kong	Monaco	Republic of Ireland	United Arab Emirates	

1 Cut along this line

If your country is not listed here, your local IBM representative will be pleased to forward your comments to us. Or you can pay the postage and send the form direct to IBM (this includes mailing in the U.K.).

2 Fold along this line

By air mail
Par avion

IBRS/CCRI NUMBER: PHQ - D/1348/SO

NE PAS AFFRANCHIR

NO STAMP REQUIRED

REPONSE PAYEE
GRANDE-BRETAGNE

IBM United Kingdom Laboratories Limited
Information Development Department (MP 095)
Hursley Park
WINCHESTER, Hants
SO21 2ZZ United Kingdom

3 Fold along this line

From: Name _____
Company or Organization _____
Address _____
_____
EMAIL _____
Telephone _____

1 Cut along this line

4 Fasten here with adhesive tape