CICS Transaction Server for z/OS

RACF Security Guide

Version 3 Release 2

SC34-6835-03

CICS Transaction Server for z/OS

RACF Security Guide

Version 3 Release 2

SC34-6835-03

Note!

Before using this information and the product it supports, be sure to read the general information under “Notices” on page 397.

This edition applies to Version 3 Release 2 of CICS Transaction Server for z/OS, program number 5655-M15, and to all subsequent versions, releases, and modifications until otherwise indicated in new editions.

Contents

Preface

What this book is about

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

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.

Although this book shows many RACF command examples, it assumes that you have access to the *z/OS Security Server RACF Security Administrator's Guide* and that you know how to issue TSO commands (or use ISPF panels to perform equivalent functions).

Notes on terminology

When the term "CICS" without any qualification in this book, it refers to the CICS element of CICS Transaction Server for z/OS®.

Other terms used are:

RACF	refers to the Resource Access Control Facility (RACF) component of Security Server, which is an optional feature of z/OS.
MVS™	refers to the operating system, which is a base element of z/OS.

Summary of changes

This book is based on the CICS RACF Security Guide for CICS Transaction Server for z/OS, Version 2 Release 1, SC34-5720-00. Changes from that edition are marked by vertical bars in the left margin.

This softcopy version is based on the printed version. Some formatting amendments may have been made to make the information more suitable for softcopy, and it may include changes made since the most recent printed version. Any such changes (apart from very minor ones) are marked by # symbols in the left margin.

This part lists the changes that have been made for the following recent releases:

Changes for CICS Transaction Server for z/OS, Version 3 Release 2

For information about changes that have been made in CICS Transaction Server for z/OS, Version 3 Release 2, please refer to *What's New* in the information center, or the following publications:

- *CICS Transaction Server for z/OS Release Guide*
- *CICS Transaction Server for z/OS Migration from CICS TS Version 3.1*
- *CICS Transaction Server for z/OS Migration from CICS TS Version 2.3*
- *CICS Transaction Server for z/OS Migration from CICS TS Version 2.2*
- *CICS Transaction Server for z/OS Migration from CICS TS Version 1.3*

Changes for CICS Transaction Server for z/OS, Version 2 Release 3

Technical changes

- Older material has been reviewed and, where necessary, revised to take into account CICS support of HTTP and IIOP clients and other recent changes.
- New topics, “CICS users” on page 4 and “The CICS region user ID” on page 6, have been added to Chapter 1, “Security facilities in CICS,” on page 3.
- New transactions have been added to in Chapter 10, “Security for CICS-supplied transactions,” on page 139.

Structural changes

The glossary has been merged with the common CICS glossary.

Changes for CICS Transaction Server for z/OS, Version 2 Release 2

Technical changes

- The following have been added:

  Protecting Java applications in CICS by using the Java 2 security policy mechanism

  Using enterprise bean security

  Security roles

  Implementing security roles
- “Identification and authentication” on page 236 has been added.

- New transactions have been added to and in Chapter 10, “Security for CICS-supplied transactions,” on page 139.
- Migration and coexistence information has been removed where it refers to releases of CICS prior to CICS Transaction Server for OS/390®, Version 1 Release 1.
- References to RACF as a stand-alone product have been removed.

Structural changes

- Information about the sign-on table migration utility (DFHSNMIG) has been moved to Appendix C, “The sign-on table migration utility,” on page 371.
- The following have been added, containing new information, and material which was previously in *Java Applications in CICS* and the *CICS Internet Guide*:

  Chapter 18, “About security for TCP/IP clients,” on page 233

  “Identification and authentication” on page 236

  “Support for security protocols” on page 244

  Chapter 19, “Configuring CICS to use SSL,” on page 253

Changes for CICS Transaction Server for z/OS, Version 2 Release 1

“Authorizing access to named counter pools and servers” on page 50 has been added to Chapter 3, “CICS system resource security,” on page 37.

New CICS resources subject to command security checking have been added to Table 11 on page 120 in Chapter 9, “Security checking using the QUERY SECURITY command,” on page 129.

In Chapter 10, “Security for CICS-supplied transactions,” on page 139 additions have been made to the Category 1 transactions.

“Support for security protocols” on page 244 now describes the use of RACF key rings to manage certificates and keys.

New resources are described in Chapter 23, “Implementing CICSPlex SM security,” on page 317

Information about the Secure Sockets Layer (SSL), which was originally in *CICS Internet Guide*, has been added.

Changes for CICS Transaction Server for OS/390, Version 1 Release 3

Chapter 4, “Verifying CICS users,” on page 65 has additional information about automatic preset security for consoles, and describes the use of the TSO CONSOLE command.

“Changing RACF profiles that are in use—caution” on page 161 contains additional information about temporary storage, including the use of temporary storage long queue names.

Chapter 7, “Surrogate user security,” on page 111 describes CICS business transaction services (BTS) processes and activities.

In Chapter 10, “Security for CICS-supplied transactions,” on page 139, additions have been made to the Category 1 transactions.

Security for CICS Web support discusses security considerations for the HTML template manager PDS, and the alias transaction, in addition to the requirements for the CICS Web support transactions. This includes a section that describes the secure sockets layer (SSL).

Chapter 13, “APPC password expiration management,” on page 177 contains information about the PEM sample program previously featured in an appendix of this manual. The external security interface (ESI) is also described here.

Chapter 17, “Security for data tables,” on page 225 has an additional section discussing security for coupling facility data tables.

Several topics have been added to explain how to implement RACF security for CICSPlex® SM. This information was previously available in the *CICSPlex SM Setup* book at the previous release. It contains the following:

- Chapter 23, “Implementing CICSPlex SM security,” on page 317 explains how to implement RACF security for CICSPlex SM
- Chapter 24, “Invoking a user-supplied external security manager,” on page 349 provides information on using a SAF-compliant external security manager other than RACF.
- Writing an API security exit describes how to write an API security exit and describes the role of the default security routine, EYU9XESV.
- Chapter 25, “Example tasks: security,” on page 353 provides examples of typical security setup tasks that you can use as a model for your own.

Many additions have been made to the table, Appendix B, “Resource and command check cross reference,” on page 359

Part 1. Introduction to CICS security with RACF

This part is an introduction to CICS security. It explains at a high level how you can use the facilities provided by RACF to make your CICS systems, and the resources in those systems, secure against unauthorized access.

Chapter 1. Security facilities in CICS

CICS provides a number of facilities that protect your resources against unauthorized access.

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 workstations and 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 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 users. As the systems become easier to use, there is also more scope for 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 users), CICS clearly needs the protection of a security system to ensure that the resources to which it manages access are protected, and are 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 assets that CICS manages, and potential exposures. 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 users to only those functions that any particular individual user is authorized to use:

Transaction security

Ensures that users that attempt to run a transaction are entitled to do so

Resource security
Ensures that users who use CICS resources are entitled to do so

Command security
Ensures that users who use CICS system programming commands are entitled to do so

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 those 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.

CICS users

When CICS security is active, requests to attach transactions, and requests by transactions to access resources, are associated with a *user*. When a user makes a request, CICS calls the external security manager to determine if the user has the authority to make the request. If the user does not have the correct authority, CICS denies the request.

In many cases, a user is a human operator, interacting with CICS through a terminal or a workstation. However, this is not always the case: a user can also be a program executing in a client system. In general, a CICS user is an entity that is identified by a *user identifier* (or *userid*).

All CICS users must be defined to the security manager; when the security manager is RACF, information about each users is stored in a user profile.

Here are some of the ways that the user of a CICS transaction, or a CICS resource, can be identified:

- A human operator signs on (and so provides a userid) at the start of the terminal session. The userid remains associated with the terminal until the terminal operator signs off. Transactions executed from the terminal, and requests made by those transactions, are associated with that userid.

  For more information, see "Identifying CICS terminal users" on page 65
- A userid is permanently associated with a terminal. Transactions executed from the terminal, and requests made by those transactions, are associated with the preset userid.

  For more information, see "Preset terminal security" on page 72.
- A client program that is communicating with CICS using the Secure Sockets Layer (SSL) supplies a client certificate to identify itself. The security manager

maps the certificate to a userid. The transaction that services the client's request, and further requests made by that transaction, are associated with that userid.

For more information, see Chapter 18, “About security for TCP/IP clients,” on page 233.

- A CICS application program issues a START command with the USERID option. The started transaction, and requests made by that transaction, are associated with the specified userid.

  For more information, see “Security for started and XPCT-checked transactions” on page 95.
- A transaction is started when the trigger level of an intrapartition transient data queue is reached. If the USERID attribute is specified in the TDQUEUE definition for the queue, then the started transaction, and requests made by that transaction, are associated with the specified userid.

  For more information, see “Protecting non-terminal transactions” on page 85.
- A remote program executing in another system, supplies a userid when it sends an attach request to CICS. The attached transaction, and requests made by that transaction, are associated with the specified userid.

  For more information, see “User security for intercommunication” on page 155.
- A remote system connects to CICS, and link security is specified for the connection to the remote system. Transactions invoked from the remote system, and requests made by that transaction, are associated with the link userid. For more information, see “Intercommunication link security” on page 154.
- A CICS business transaction services (BTS) process is activated by a RUN command, and the DEFINE PROCESS command specified the USERID option.. The transaction under which the process runs, and requests made by that transaction, are associated with the specified userid.

  For more information, see *CICS Business Transaction Services*.
- A second phase PLT program runs during CICS initialization. Depending upon the value of the PLTPISEC system initialization parameter, requests made by the program are associated with the userid specified in the PLTPIUSR system initialization parameter.

  For more information, see “PLT programs” on page 85.

There are two user IDs that CICS uses in addition to those that identify individual end users. They are

The region user ID

is used for authorization checking when the CICS system (rather than an individual user of the system) requests access to a resource.

For more information, see “The CICS region user ID” on page 6.

The default user ID

identifies the user whose security attributes are used to protect CICS resources in the absence of other, more specific, user identification.

For more information, see “The CICS default user ID” on page 7.

By itself, a userid does not protect the system from unauthorized access: in many cases, userids are known to other people than the user they identify. To prevent impersonation, another piece of information — known only to the individual user — must be supplied in order to authenticate the user. For example:

- For a terminal user, the password which the user supplies during sign on authenticates the user.

- For a client using SSL, the client certificate authenticates the client.

The CICS region user ID

The CICS region user ID is used for authorization checking when the CICS system (rather than an individual user of the system) requests access to a resource. CICS uses the region user ID when checking authorization for these resources:

MVS system log streams
For more information, see “Authorizing access to MVS log streams” on page 44.

CICS system data sets
For more information, see “Authorizing access to CICS data sets” on page 45.

CICS user data sets
For more information, see “Authorizing access to user data sets” on page 47.

Temporary storage data sharing servers
For more information, see “Authorizing access to temporary storage servers” on page 49.

The SMSVSAM server
For more information, see “Authorizing access to SMSVSAM servers” on page 51.

Named counter servers
For more information, see “Authorizing access to named counter pools and servers” on page 50.

The VTAM® ACB
For more information, see “Controlling the opening of a CICS region's VTAM ACB” on page 53.

JES spool data sets
For more information see “JES spool protection in a CICS environment” on page 55.

The CICS interregion program
For more information, see “Logon security checking with MRO” on page 211.

Coupling facility data tables
For more information, see “Security for coupling facility data tables” on page 228.

RACF key rings
For more information, see “Building a key ring manually” on page 256.

CICS may also use the region user ID:

When submitting jobs to the JES internal reader
For more information, see “Controlling userid propagation” on page 54.

As a surrogate for other user IDs used during CICS execution
For more information, see Chapter 7, “Surrogate user security,” on page 111.

As a prefix for resource names passed to RACF
For more information, see the description of the SECPRFX system initialization parameter in “Security-related system initialization parameters” on page 56.

When executing CICS-supplied non-terminal transactions
For more information, see “Category 1 transactions” on page 140.

In CICS intersystem communication using LU6.2
For more information, see Chapter 12, “Implementing LU6.2 security,” on page 157.

In CICS intersystem communication using LU6.1
For more information, see Chapter 14, “Implementing LU6.1 security,” on page 199.

In CICS intersystem communication using MRO
For more information, see Chapter 16, “Implementing MRO security,” on page 211.

When deriving the distinguished name of an enterprise bean client
For more information, see Deriving distinguished names.

The CICS region user ID is assigned to a CICS region at initialization, and is the user ID that is associated with the job or started task. For more information, see “Specifying the CICS region userid” on page 38.

The CICS default user ID

The CICS default user ID identifies the user whose security attributes are used to protect CICS resources in the absence of other, more specific, user identification.

The default user ID is specified in the **DFLTUSER** system initialization parameter. If you do not specify the parameter, the default user ID is `CICSUSER`.

- It is assigned to a terminal or a console before a user signs on, and after the user has signed off, except when the terminal or console has preset security specified. For more information, see Chapter 4, “Verifying CICS users,” on page 65.
- It is assigned to transient data trigger-level transactions that are not associated with a terminal, and when a user ID is not specified in the definition of the transient data queue. For more information, see “Transient data trigger-level transactions” on page 113.
- It is used as the link user ID for LU6.1 and LU6.2 connections when the SECURITYNAME attribute is not specified in the CONNECTION definition. For more information, see “Link security with LU6.1” on page 199 and “Link security with LU6.2” on page 161.
- It is assigned to transactions attached by LU6.2 and MRO sessions, when the attach request does not contain security parameters, and the CONNECTION definition specifies USEDFLTUSER(YES). For more information, see “SNA profiles and attach-time security” on page 167 and “User security with MRO” on page 215.
- For transactions in an application-owning region (AOR) that are initiated using the CICS routing transaction (CRTE), the default user ID is used if the terminal user does not sign on to the AOR while using CRTE. For more information, see “Transaction routing security with LU6.2” on page 169 and “Transaction routing security with MRO” on page 218.

- It is used when an application-owning region (AOR) issues a request for a remote file that is defined as a shared data table, and the AOR is unable to sign on to the file-owning region (FOR). For more information, see "Security for CICS shared data tables" on page 225.
- In the absence of more explicit identification, it is used to identify TCP/IP clients that connect to CICS. For more information, see "Identification" on page 236.

Terminal user security

To secure resources from unauthorized access, CICS needs some means of uniquely identifying individual users of the system.

For this purpose, 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 (user ID) and the associated password, or operator identification card (OIDCARD) in the CICS-supplied sign-on transaction, CESN. Alternatively, they can use an equivalent transaction developed by your own installation by issuing the EXEC CICS SIGNON command provided for this purpose.

When users enter the CESN transaction, CICS verifies user IDs 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. If the user fails to complete signon, all subsequent transactions use the CICS default user ID.

See "Terminal profiles" on page 69 for information about the terminal security facilities provided by RACF. See Chapter 4, "Verifying CICS users," on page 65 for information about using terminal user security in CICS.

For some terminals, and for MVS consoles which are used as CICS terminals, it may be appropriate to use *preset terminal security*. Preset terminal security allows you to associate a user ID permanently 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 sign-off can be performed against it. If the user ID associated with a display terminal with preset security authorized to use any sensitive transactions, ensure that the terminal is in a secure location to which access is restricted. For example, terminals physically located within a CICS network control center might be appropriate for preset security.

For more information, see "Preset terminal security" on page 72.

Non-terminal security

You can also specify security 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 “Protecting non-terminal transactions” on page 85.

The QUERY SECURITY command

In addition to using CICS security checking for CICS-controlled resources (or as an alternative to it), you can use the 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 standard resource classes.

See “RACF general resource profiles” on page 23 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 87.

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**; by for example, intersystem communication over SNA (ISC over SNA), which uses 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 Intercommunication methods*CICS Intercommunication Guide* for more information about CICS intercommunication facilities.)

One of the ISC over SNA 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. CICS provides the following security mechanisms for the APPC environment:

- Bind-time (or session) security prevents an unauthorized remote system from connecting to CICS.
- Link security defines the complete set of CICS transactions and resources that the remote system is permitted to access across the connection.
- User security checks that a user is authorized both to attach a CICS transaction and to access all the resources and SPI commands that the transaction is programmed to use.

See Chapter 12, “Implementing LU6.2 security,” on page 157 for more information.

IPIC security

This topic introduces the security mechanisms provided for IP interconnectivity connections.

Note: An IPIC connection is also called an "IPCONN". For information about IPIC connections, see the *CICS Intercommunication Guide*.

The security mechanisms for IPCONNs are similar to those provided for APPC (LU6.2) connections (though they are implemented differently):

- Bind-time security prevents an unauthorized remote system from connecting to CICS. On IPCONNs, bind security is enforced by the exchange of Secure Sockets Layer (SSL) client certificates.
- Link security defines the complete set of CICS transactions and resources that the remote system is permitted to access across the IPCONN.
- User security checks that a user is authorized both to attach a CICS transaction and to access all the resources and SPI commands that the transaction is programmed to use. User security is a subset of link security: that is, a user cannot access a resource, even if it is included in the set defined as accessible by his user ID, if is not also included in the set of resources accessible by the link user ID.

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 16, "Implementing MRO security," on page 211 for information about MRO security.

Front End Programming Interface security

The security options provided for the Front End Programming Interface are equivalent to those provided for CICS command security (see Chapter 8, "CICS command security," on page 119). Front End Programming Interface security is not discussed in this book, but in the *CICS Front End Programming Interface User's Guide*.

CICS Business Transaction Services

CICS Business Transaction Services (BTS) also uses security options equivalent to those provided for CICS command security (see Chapter 8, "CICS command security," on page 119). Details of security for BTS are explained in *CICS Business Transaction Services*.

Generating and using RACF PassTickets

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

- A specific PassTicket may be used for authentication **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.

Front end programming interface (FEPI) security can generate a PassTicket for use on a target system. The PassTicket can be used anywhere a password can be used.

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 *z/OS Security Server RACF Security Administrator's Guide*.

Use the PTKTDATA resource class to define profiles that contain 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))
```

Chapter 2. RACF facilities

CICS uses a number of RACF facilities in order to protect its resources.

Overview of RACF facilities

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 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 *z/OS Security Server RACF Auditor's Guide* . 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 *z/OS Security Server 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
  - That authority which goes with being the OWNER of existing RACF profiles, 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 on the scope of a RACF group, see the *z/OS Security Server RACF Auditor's Guide*.

For information on the RACF requirements for issuing RACF commands, see the descriptions of the commands in the *z/OS Security Server RACF Command Language Reference*.

You can find out whether you have the system-SPECIAL or group-SPECIAL attribute by issuing the LISTUSER command from a TSO 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 *z/OS Security Server RACF General User's Guide*.

Delegation of RACF administrative responsibility

As CICS security administrator, you perform the following tasks (if you do not have the system-SPECIAL attribute, obtain the necessary authority):

- **Define and maintain profiles in CICS-related general resource classes**. In general, you grant authority to do this 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). For more information, see "Activating the CICS classes" on page 23.
- **Define and maintain profiles in other resource classes**. 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, you grant this authority by assigning the CLAUTH (class authority) attribute for "USER" 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, assign that user a default connect group. This 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**. You grant authority to do this 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 with prefixing active, administrators can be assigned. You do this 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 "Summary of RACF commands" on page 29.

For more information on delegating RACF administration, see the *z/OS Security Server RACF Security Administrator's Guide* .

RACF profiles

In RACF, a *profile* describes the security characteristics of a user, a group of users, or one or more computer resources:

User profiles
A *user profile* is a description of a RACF-defined user. The information in the profile includes the user ID, the user name, the user's password, the profile owner, user attributes, and other data. The user profile also contains user-related information for subsystems, including CICS.

Group profiles
A *group profile* defines a group of users. The information in the profile includes the group name, the profile owner, and the users in the group.

Data set profiles
A *data set profile* provides RACF protection for one or more data sets. The information in the profile includes the data set profile name, the profile owner, the universal access authority, the access list, and other data.

Data set profiles can be generic or discrete:

- A *generic profile* protects several resources with similar names and identical security requirements.
- A *discrete profile* protects a single resource.

General resource profiles
.

A *general resource profile* provides RACF protection for computer resources, other than data sets. The information in the profile includes the general resource profile name, the profile owner, the universal access authority, the access list, and other data. General resources with similar characteristics belong to the same *class*.

Like a generic profile, a *resource group profile* protects several resources with identical security requirements. However, the resources do not have to have similar names. Resource group profiles with similar characteristics belong to the same *resource group class*.

Resource profiles can be generic or discrete:

- A *generic profile* protects several resources with similar names and identical security requirements.
- A *discrete profile* protects a single resource.

RACF user profiles

A user profile is a description of a RACF-defined user. The information in the profile includes the user ID, the user name, the user's password, the profile owner, user attributes, and other data. The user profile also contains user-related information for subsystems, including CICS.

The user profiles consists 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. See “The RACF segment.”
- The CICS segment, which holds data for each CICS user. See “The CICS segment” on page 17.
- The LANGUAGE segment, which specifies the user's national language preference. See “The LANGUAGE segment” on page 20.

The 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 user's userid
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, 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, an administrative unit. 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 *z/OS Security Server RACF Security Administrator's Guide*.

The 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 77.

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

OPCLASS({1|*number*})
CICS uses the operator classes when routing basic mapping support (BMS) messages initiated within a CICS transaction. The operator classes are numeric values in the range 1–24.

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

For more information about message routing, see the *CICS Application Programming Guide*.

OPIDENT({blank|*name*})
The 1- to 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. It is also used when using the CEDA LOCK command, as described in the *CICS Resource Definition Guide*.

For more information about message routing, see the *CICS Application Programming Guide*.

OPPRTY({0|*number*})
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.

TIMEOUT({0000|*hhmm*})
The time that must elapse since the user last used the terminal before CICS "times-out" the terminal.

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

To specify one hour and eight minutes 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.

XRFSOFF({NOFORCE|FORCE})
The CICS persistent sessions restart and extended recovery facility (XRF) sign-off option. You specify this to indicate whether or not you want CICS to sign off the operator following a persistent sessions restart or an XRF takeover.

FORCE
Specify FORCE if you want CICS to sign off the operator automatically in the event of a persistent sessions restart or an XRF takeover.

NOFORCE
Specify NOFORCE if you want CICS to leave an operator signed on in the event of a persistent sessions restart or an XRF takeover.

You can specify the XRFSOFF function at the level of groups of similar terminals, and at the CICS system level:

- Use the RSTSIGNOFF attribute of the TYPETERM resource definition to specify the XRFSOFF function for groups of similar terminals.
- Use the **XRFSOFF**system initialization parameter to specify the function at the system level.

In both cases, the default value is NOFORCE. If you specify the FORCE option in the system initialization table or the TYPETERM, it overrides a value of NOFORCE specified in the CICS segment.

Table 1 shows how specifying FORCE or NOFORCE in the system initialization parameters, on the TYPETERM definition, and in the CICS segment together determine whether a terminal remains signed on after a persistent sessions restart or an XRF takeover.

As Table 1 shows, for a terminal to remain signed-on after a persistent sessions restart or an XRF takeover, NOFORCE must be specified in all three locations.

Table 1. Effects of FORCE and NOFORCE options

TYPETERM definition	CICS segment	System initialization parameter	Resulting terminal status
FORCE	FORCE	FORCE	Signed-off
FORCE	FORCE	NOFORCE	Signed-off
FORCE	NOFORCE	FORCE	Signed-off
FORCE	NOFORCE	NOFORCE	Signed-off
NOFORCE	FORCE	FORCE	Signed-off
NOFORCE	FORCE	NOFORCE	Signed-off
NOFORCE	NOFORCE	FORCE	Signed-off
NOFORCE	NOFORCE	NOFORCE	**Signed-on**

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:

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

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

Specifying default values in the CICS segment

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 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 77.

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
```

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, or 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(priori`ty)
             TIMEOUT(timeout_value) XRFSOFF(NOFORCE))
        LANGUAGE(PRIMARY(primary_language))
```

Before issuing these commands to define CICS operator data, 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 37 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. For more information, see "Controlling access to fields in RACF profiles" on page 33.

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))
```

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 who 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 define a CICS default user id to RACF. You then tell CICS which default user to use by specifying the **DFLTUSER** system initialization parameter. If you do not specify a default user id 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 user id has installed surrogate security to use the default user (see Chapter 7, “Surrogate user security,” on page 111).

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 who do not explicitly sign on. These security checks include **resource** and **command** security checking, in addition to **transaction-attach** security checking.

Note: If the default user's RACF profile specifies a non-zero TIMEOUT, that value does **not** apply to terminals that do not sign on.

The 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 each user only one language. 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 A, “National Language,” on page 357.

SECONDARY(secondary_language)
This parameter identifies the user's secondary language, overriding the system default. You can specify this as one of the 3-character codes listed in Appendix A, "National Language," on page 357.

For more information about national language, see "National language and non-terminal transactions" on page 80.

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 "RACF general resource profiles" on page 23.) 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. For details of how to define and use group profiles, see the *z/OS Security Server RACF Security Administrator's Guide* .

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
- Fewer 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 an excessive number of resource profiles.

RACF 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.

The authority to access a resource is kept in an access list that is part of the resource profile. The authority can be granted to a user or to a group. To add or remove a user from the access list, refresh the profile in main storage. For more information see "Refreshing resource profiles in main storage" on page 24.

If you connect and remove a user from a group that is already in the access list, that user acquires or loses the authority of the group without needing to refresh the profile. 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 *z/OS Security Server 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 **USRDELAY** system initialization parameter, which specifies how long an unused userid can remain signed on. (This can be up to 7 days.)

RACF data set profiles

Using RACF facilities, you can protect data sets on direct access storage devices (DASD) and tapes. You do this by defining profiles for the data sets you want to protect. The rules for defining data set profiles to RACF are described in the *z/OS Security Server RACF Security Administrator's Guide*, and the *z/OS Security Server RACF Command Language Reference*. For examples, see the *z/OS Security Server 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 21 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 *z/OS Security Server RACF Security Administrator's Guide* for details.)

Note: Data set profiles do not apply to CICS terminal users, but 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 required 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 *z/OS Security Server RACF Command Language Reference* for the rules governing generic profile names in the RACF DATASET class.

For example, if you have a group called CICSTS32.CICS, you can define a generic profile named 'CICSTS32.CICS.**', and any user in the access list of this profile can access, at the authorized level, data sets with the high-level qualifier CICSTS32.CICS. For example:

```
ADDSD 'CICSTS32.CICS.**' UACC(NONE) NOTIFY(admin_userid)
```

Use the SETROPTS GENERIC command before defining generic profiles, as described in “Summary of RACF commands” on page 29.

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 a single asterisk (*) in place of double asterisks. (You put enhanced generic naming into effect by issuing the RACF SETROPTS EGN command. Note that SETROPTS EGN affects only data set names. Enhanced generic naming is always in effect for general resource profiles, such as TCICSTRN.)

RACF general resource profiles

A *general resource profile* provides RACF protection for computer resources, other than data sets. The information in the profile includes the general resource profile name, the profile owner, the universal access authority, the access list, and other data. General resources with similar characteristics belong to the same *class*.

Like a generic profile, a *resource group profile* protects several resources with identical security requirements. However, the resources do not have to have similar names. Resource group profiles with similar characteristics belong to the same *resource group class*.

RACF supplies a number of resource classes that CICS uses for its resources. They are described in “RACF classes for CICS resources” on page 24. Other RACF resource classes contain profiles that are used for resources that are used by CICS and other subsystems. They are described in “RACF classes for protecting system resources” on page 25.

Protecting a resource

These are the steps that you must perform in order to protect a resource:

1. Define a profile for the resource in a suitable resource class
2. Define an *access list* which specifies:
   - the users that are permitted to access the resource
   - the level of access that each user is allowed

Activating the CICS classes

To activate the CICS resource class for use in security checking by the CICS region, use the RACF SETROPTS command. As soon as the CICS resource class is defined in the active RACF class descriptor table, administrators can define general resource profiles to the class. For more information, see the descriptions of RDEFINE and PERMIT in “RACF general resource profiles.” Note that the class must be activated before the CICS system can use the profiles that the administrators define.

The format of the SETROPTS command is SETROPTS CLASSACT(*classname*). For example:

```
SETROPTS CLASSACT(TCICSTRN)
```

All sets of RACF general resource classes that have the same POSIT number in their CDT definitions are activated and deactivated together. Therefore, you need only specify one IBM-supplied CICS class to activate all the IBM-supplied

CICS-related classes. If you define your own installation-defined classes with the same POSIT number as the IBM-supplied classes, they are activated and deactivated with the IBM-supplied classes. To provide separate controls for sets of installation-defined classes, define them with different POSIT numbers. (For more information on the POSIT number, see the *z/OS Security Server RACF Macros and Interfaces* manual.)

Refreshing resource profiles in main storage

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`.

After adding or updating a profile, either for a member class or a resource grouping class, you must issue the command:

```
SETROPTS RACLIST (TCICSTRN) REFRESH
```

Even if you have added the profile to the GCICSTRN resource grouping class, you must issue this command for the TCICSTRN member class only. When you use this command, the definitions are updated in RACF and CICS uses the changed profiles.

RACF classes for CICS resources

To protect a CICS resource, you must create a general resource profile for the resource in a suitable class or resource grouping class. RACF provides several classes for CICS resources. You can also define your own resource classes.

Note:

1. The initial character of the class names is significant when you define your own classes. For example, if you define your own classes for CICS programs, the names you choose must start with M and N for the menber class and the resource grouping class respectively.
2. There are no default resource class names for DB2ENTRY resources. You define your own resource classes for these resources. See “Resource classes for DB2ENTRYs” on page 28 for more information.

Table 2. RACF—provided resource classes for CICS resources

Member class	Resource grouping class	Description
TCICSTRN	GCICSTRN	CICS transactions, normal attach security
PCICSPSB	QCICSPSB	CICS PSBs
ACICSPCT	BCICSPCT	CICS-started transactions and the following EXEC CICS commands: COLLECT STATISTICS TRANSACTION DISCARD TRANSACTION INQUIRE TRANSACTION SET TRANSACTION
DCICSDCT	ECICSDCT	CICS transient data queues
FCICSFCT	HCICSFCT	CICS files
JCICSJCT	KCICSJCT	CICS journals

Table 2. RACF—provided resource classes for CICS resources (continued)

Member class	Resource grouping class	Description
MCICSPPT	NCICSPPT	CICS programs
SCICSTST	UCICSTST	CICS temporary storage queues
CCICSCMD	VCICSCMD	EXEC CICS SYSTEM commands and EXEC CICS FEPI system commands
RCICSRES	WCICSRES	CICS document templates

RACF classes for CICSPlex SM resources

Protection for CICSPlex SM resources is provided by the following general resource classes:

CPSMOBJ
Controls access to CICSPlex SM resources. The corresponding resource group class is GCPSMOBJ. For more information, see Chapter 23, “Implementing CICSPlex SM security,” on page 317.

CPSMXMP
Controls exemption from simulated CICS security checking in CICSPlex SM. For more information, see “Exempting users and resources from security checking” on page 341

RACF resource class names

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 as RACF loads the profiles into storage, it merges the profiles according to the ICHRLX02 selection exit. 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 *z/OS Security Server RACF Data Areas*.

RACF classes for protecting system resources

CICS uses many system resources, and these must be protected against unauthorized access. This protection is provided by profiles in the following general resource classes:

APPCLU
Verifies the identity of APPC partner logical units (LU type 6.2) during VTAM session establishment. For more information, see “Defining profiles in the APPCLU general resource class” on page 158.

APPL Controls terminal users' access to VTAM applications, including CICS. For more information, see "Authorizing access to the CICS region" on page 52.

CONSOLE
Controls user access to consoles. For more information, see "Console profiles" on page 71.

DIGTCERT
Contains digital certificates, and related information. For more information, see "Creating new RACF certificates" on page 258.

EJBROLE
Contains security roles used for enterprise bean security roles. The corresponding resource group class is GEJBROLE. For more information, see *Java Applications in CICS*.

FACILITY
The FACILTY general resource class is used to protect several different system resources. These are described in "Resources protected by the FACILITY general resource class" on page 27.

FIELD Controls access to fields in RACF profiles. For more information, see "Controlling access to fields in RACF profiles" on page 33.

JESSPOOL
Protects JES spool data sets. For more information, see "JES spool protection in a CICS environment" on page 55.

LOGSTRM
Controls access to the MVS logstreams that CICS uses for its system logs and general logs. For more information, see "Authorizing access to MVS log streams" on page 44.

OPERCMDS
- Controls which console users are allowed to issue MODIFY commands directed to particular CICS regions. For more information, see "Using an MVS system console as a CICS terminal" on page 75.
- Controls which operator commands CICS can issue; for example, commands in the command list table (CLT), and MODIFY network commands.

PROGRAM
Controls which users can start CICS. For more information, see "Protecting CICS load libraries" on page 37.

PROPCNTL
Prevents the CICS region userid being propagated to jobs that are submitted from CICS to the JES internal reader, and that do not specify the USER operand. For more information, see "Controlling userid propagation" on page 54.

PTKTDATA
Contains the encryption keys used for generating and validating PassTickets. For more information, see "Generating and using RACF PassTickets" on page 10.

SERVAUTH
Define profiles in the SERVAUTH general resource class to establish a trust relationship between servers when using asserted identity authentication for IIOP clients. For more information, see "Authentication" on page 240.

STARTED

Contains profiles that provide the userids for MVS started jobs. For more information, see "Using STARTED profiles for started jobs" on page 40.

SUBSYSNM

Authorizes subsystems (such as instances of CICS) to open a VSAM ACB and use VSAM Record Level Sharing (RLS) functions. For more information, see "Authorizing access to SMSVSAM servers" on page 51.

SURROGAT

Specifies which userids can act as surrogates for other userids. For more information, see Chapter 7, "Surrogate user security," on page 111.

TERMINAL

Controls the ability of users to sign on at individual terminals. The corresponding resource group class is GTERMINL. For more information, see "Preset terminal security" on page 72.

VTAMAPPL

Controls the ability of users to open a VTAM ACB. For more information, see "Controlling the opening of a CICS region's VTAM ACB" on page 53.

Resources protected by the FACILITY general resource class

The FACILTY general resource class is used to protect several different system resources. They are:

Library lookaside (LLA) libraries

The FACILITY class controls a program's ability to use the LLACOPY macro. For more information, see "Authorizing access with the MVS library lookaside (LLA) facility" on page 47.

The CICS interregion communication program, DFHIRP

The FACILITY class controls a region's ability to log on to the CICS interregion communication program, DFHIRP. For more information, see Chapter 16, "Implementing MRO security," on page 211.

Shared data tables

The FACILITY class controls a file-owning region's ability to act as a shared data table server. For more information, see "Security for CICS shared data tables" on page 225.

Coupling facility data tables

The FACILITY class controls the following:

- The ability of a CICS region to connect to a coupling facility data table (CFDT)
- The ability of the CFDT server to act as a server for a CFDT pool
- The ability of the CFDT server region to connect to the coupling facility list structure for its CFDT pool

ability of a coupling facility data table (CFDT) server to act as a server for a CFDT pool. It also controls the server's access to the coupling facility list structure for the pool. For more information, see "Security for coupling facility data tables" on page 228.

Named counter servers

The FACILITY class controls the following:

- The ability of a CICS region to connect to a named counter server
- The ability of the named counter server to act as server for a named counter pool

- The ability of the named counter server region to connect to the coupling facility list structure for its named counter pool

For more information, see "Authorizing access to named counter pools and servers" on page 50.

Shared temporary storage
The FACILITY class controls the following:
- The ability of a CICS region to connect to a shared temporary storage (TS) server
- The ability of the shared TS server to act as server for a shared TS pool
- The ability of the shared TS server region to connect to the coupling facility list structure for its shared TS pool

For more information, see "Authorizing access to the temporary storage pools" on page 48.

Log streams
The FACILITY class controls the ability of a CICS region to connect to the coupling facility list structures used for MVS log streams. For more information, see "Authorizing access to MVS log streams" on page 44.

AUTHTYPE and COMAUTHTYPE userids in DB2® definitions
The FACILITY class controls the following:
- The ability of users to install DB2CONN and DB2ENTRY definitions that specify the AUTHID or COMMAUTHID attributes
- The ability of users to use a CREATE DB2CONN or CREATE DB2ENTRY command that specifies the AUTHID or COMMAUTHID attributes
- The ability of users to change the AUTHID or COMMAUTHID attributes of an installed DB2CONN or DB2ENTRY

For more information, see the *CICS DB2 Guide*.

CICSPlex SM resources
The FACILITY class controls access to a many CICSPlex SM resources. For more information, see Chapter 23, "Implementing CICSPlex SM security," on page 317.

Resource classes for DB2ENTRYs

CICS supports resource security checking for CICS-defined DB2ENTRY resources, for which there are no IBM-supplied RACF resource classes. For DB2ENTRYs, you define security profiles in user-defined class names, and use the XDB2 system initialization parameter to specify the class name to CICS. The syntax for the XDB2 system initialization parameter is XDB2=NO|*name*, which does not support a default class name like the other security system initialization parameters. Use the DFH$RACF sample job as an example of how to define DB2 resource class names for CICS use.

Do not use one of the CICS default resource classes in which to define DB2ENTRY profiles. CICS uses RACLIST to activate the profiles in the default resource classes according to the X*name* system initialization security parameters you specify, and XDB2 should specify a user-defined class name defined specifically for DB2ENTRY resources.

Summary of RACF commands

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

Note:

1. The commands described here, and the operands used in the examples, are not exhaustive.
2. The sequences of commands shown here demonstrate one way to accomplish a given task. There may be other sequences of commands that you can use.

For full details of RACF commands, refer to *z/OS Security Server RACF Command Language Reference*

Creating general resource profiles

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.

In this example, the three RDEFINE commands define three profiles named CEMT, CEDA, and CEDB in the TCICSTRN resource class. The three PERMIT commands allow two groups of users to access each 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)
```

Creating a resource group profile

To define a profile in a resource grouping class, use the RDEFINE command with the ADDMEM operand to add resources as members of the group. Generally, once you have created a profile, you then create an access list for the profile using the PERMIT command.

In this example, the RDEFINE command defines a resource group profile named CICSTRANS in the GCICSTRN resource grouping class. The PERMIT command allows two groups of users to access all transactions in the profile.

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

Creating a general resource profile

Use the RDEFINE command to create a profile in a general resource class:

```
RDEFINE class profile UACC(NONE)
```

where:

class is the name of the general resource class
profile is the name of the new profile

Specify `UACC(NONE)` to ensure that there is no default access to the profile.

Permitting access to a general resource

To permit access to a general resource, use the PERMIT command to create an access list for the general resource profile:

```
PERMIT profile CLASS(class)
       ID(user) ACCESS(authority)
```

where:

profile is the name of the new profile
class is the name of the general resource class
user is the user (or group of users) that is being given access authority to the resource
authority is the level of authority that is being granted to the user

Removing an entry from an access list

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|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 profile, use the RDELETE command. For example:

```
RDELETE class_name profile_name
```

Copying from a profile

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 *z/OS Security Server RACF Command Language Reference*.

Listing profiles in a class

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 "RACF general resource profiles" on page 23.

Note: 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 *z/OS Security Server RACF Command Language Reference*, and "Which profile is used to protect the resource?" on page 303.

Activating protection for a class
: To begin protecting all the resources protected by profiles in a RACF class, activate that class by issuing the SETROPTS command with CLASSACT specified:

```
SETROPTS CLASSACT(class_name)
```

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

```
SETROPTS GENERIC(class_name)
```

Deactivating protection for 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. CICS treats this response as "access denied". To deactivate a RACF class, issue the SETROPTS command with NOCLASSACT specified:

```
SETROPTS NOCLASSACT(class_name)
```

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

```
SETROPTS LIST
```

| **Activating support for mixed case passwords**
| To turn support for mixed case passwords on, issue the SETROPTS command with PASSWORD specified:

```
SETROPTS PASSWORD(MIXEDCASE)
```

| To turn support for mixed case passwords off, issue the SETROPTS command:

```
SETROPTS PASSWORD(NOMIXEDCASE)
```

| Mixed case passwords are supported in z/OS Security Server (RACF) 1.7 and above.

Security classification of data and users

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

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 *z/OS Security Server 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 *z/OS Security Server 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 belong to one or more groups, one of which is their default. The sign-on process allows the user to override the default RACF user group name. If "list of groups checking" is inactive, signing on with different group names might give a user different authorities.

Defining your own resource classes

You can define your own resource classes so that you have a unique resource class name for each CICS region. 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 "Security-related system initialization parameters" on page 56.)

Group administrator for each region

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

To get this benefit, 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.

Use the SETROPTS GENERIC command before defining generic profiles, as described in "Summary of RACF commands" on page 29.

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, think of 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 21, “Customizing security processing,” on page 289.

All installation-defined classes 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 *z/OS Security Server RACROUTE Macro Reference*. Also see “Specifying user-defined resources to RACF” on page 294.

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 *z/OS Security Server 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 *z/OS Security Server RACF Macros and Interfaces* manual for details of valid values for these operands. See the same manual for information about creating installation-defined resource classes. For an example of how to add resource classes, see the IBM-supplied sample, DFH$RACF, which is in CICSTS32.CICS.SDFHSAMP.

For CICS resources, the first character of the resource class name is predefined by CICS, consistent with the default resource class name. You can define the second through eighth characters of the resource class name, but for ease of administration it is recommended that you specify the same characters for both the member and group class. The seven 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 following:

- XDB2, which has no CICS-defined prefix letter, so any defined class name of 1- to 8-characters can be specified. It is recommended that you use a specific class or classes dedicated to these resources.
- XAPPC and XUSER, which have no “name” option, and are either YES or NO to say whether security is activie or not.

You should avoid using the letters “CICS” in the second through fifth characters in any class name you define. RACF requires that at least one of the characters in the classname should be a national or numeric character.

Controlling access to fields in RACF profiles

Use the FIELD resource class to define profiles that control access to fields in the RACF database. By creating profiles in the RACF FIELD class, in 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 use the SETROPTS GENERIC command, as described in “Summary of RACF commands” on page 29.

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 *z/OS Security Server RACF Security Administrator's Guide*.

Part 2. Implementing RACF protection in a single CICS region

This part explains how to protect CICS resources in a single CICS region.

Chapter 3. CICS system resource security

This topic describes how to protect the system resources that CICS requires.

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, use the CICS-supplied RACF dynamic parse validation routines. Install these routines in SYS1.CICSTS32.CICS.SDFHLINK, which should be made an APF-authorized library in your MVS linklist. (For more information, see the *CICS Transaction Server for z/OS Installation Guide*.)

The routines are as follows:

DFHSNNFY
CICS segment update notification

DFHSNPTO
CICS segment TIMEOUT print formatting

DFHSNVCL
CICS segment OPCLASS keyword validation

DFHSNVID
CICS segment OPIDENT keyword validation

DFHSNVPR
CICS segment OPPRTY keyword validation

DFHSNVTO
CICS segment TIMEOUT keyword validation

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.

Setting options on the MVS program properties table

If you have an entry for the DFHSIP program in your MVS program properties table (PPT), ensure that the NOPASS option is **not** set for DFHSIP in the PPT statement of the SCHEDxx member of the SYS1.PARMLIB library. Setting the NOPASS option would bypass 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 Transaction Server for z/OS 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 CICSTS32.CICS.SDFHAUTH. This library must be defined to the operating system as APF-authorized.

To prevent unauthorized or accidental modification of CICSTS32.CICS.SDFHAUTH, make this library 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, 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. Instead, 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.

Also give RACF protection to SYS1.CICSTS32.CICS.SDFHLINK and to SYS1.CICSTS32.CICS.SDFHLPA; and the other libraries (including CICSTS32.CICS.SDFHLOAD) that make up the STEPLIB and DFHRPL library concatenations.

See “Authorizing access to CICS data sets” on page 45 for more information about protecting CICS data sets and creating suitable data set security profiles.

Note: The source statements of your application programs are sensitive; consider having RACF protect the data sets containing them.

Specifying the CICS region userid

When you start a CICS region (either as a job or as 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 which 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” on page 39.) 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 *z/OS Security Server RACF System Programmer's Guide*.

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 the authorizations for different CICS regions are properly differentiated, 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. For more information, see "Link security with LU6.2" on page 161, "Link security with LU6.1" on page 199, or "Link security with MRO" on page 214, depending on the environment you are using.

Using protected user IDs

If you run CICS as a started task, consider defining the CICS region user ID and the CICS default userid as protected user IDs.

Protected user IDs cannot be used to enter the system by any means that requires a password, such as logging on to TSO, signing on to CICS, or running a batch job that specifies a password on the JOB card. Users cannot, by inadvertently or deliberately entering an incorrect password, cause a protected user ID to be revoked.

To create a protected user ID, specify the NOPASSWORD and NOIDCARD options attributes on the user profile. For more information about protected user IDs, see *z/OS Security Server RACF Security Administrator's Guide*.

Authorizing CICS procedures to run under RACF

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

Using the ICHRIN03 table for started tasks

If you run CICS as a started task, 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 *z/OS Security Server RACF System Programmer's Guide* for more information about this table, and how you can add the default entry for the cataloged procedure name for starting CICS.

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

If your ICHRIN03 table does not contain the default entry (or you choose not to set the default entry), 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, ensure that the RACF user identified through ICHRIN03 has 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

If you start your CICS regions as started jobs, you can use separate userids for each started region, 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.

Ensure that the userids specified in STDATA segments are defined to RACF. 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 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 (AORs):

```
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 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 where 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 (CICSDAA2) 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 bringing up a CICS region, ensure that the required userids are defined - 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, 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 you do not specify DFLTGRP, 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” on page 41 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
```

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, 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 54 for more information. The region userid must also have surrogate authority to use the default user; see Chapter 7, “Surrogate user security,” on page 111.

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 needs to use. There are two types of these resources:

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

Authorizing external resources

Like a 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.

Give access to these resources:

- The MVS system logger

  CICS needs authority to use log streams defined in the MVS logger. See “Authorizing access to MVS log streams” on page 44.
- External data sets used by CICS

  CICS needs authority to open all the data sets that it uses. See “Authorizing access to CICS data sets” on page 45.
- External data sets used by application programs

  CICS needs authority to open all the data sets that your own application programs need. See “Authorizing access to user data sets” on page 47.
- Temporary storage servers

CICS needs authority to access temporary storage servers if any TS queues are defined as shared. See “Authorizing access to temporary storage servers” on page 49.

- SMSVSAM servers

  CICS needs authority to access the SMSVSAM server if you are using VSAM record-level sharing (RLS). See “Authorizing access to SMSVSAM servers” on page 51.
- VTAM applications

  Consider carefully for each program whether you will allow it 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 53.
- 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 54.

  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 54.
- 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 55.

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 they 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 140.
- Auxiliary user ids

  If CICS surrogate user checking is specified with the **XUSER** system initialization parameter (the default), CICS needs authority to use certain additional user ids. These are:
  - The default userid

    See “CICS default user” on page 111.
  - The userid used for post-initialization processing (PLTPIUSR)

    See “Post-initialization processing” on page 111.
  - The userid used for transient data trigger transactions

    See “Transient data trigger-level transactions” on page 113.
- 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, give the CICS region userid access to all the CICS resources that these programs use. See “PLT programs” on page 85.

Defining the default CICS userid to RACF

For each CICS region where you specify SEC=YES, define a RACF user profile whose userid matches the value of the **DFLTUSER** system initialization parameter.

For example, if you specify DFLTUSER=NOTSIGND, 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. Define a RACF user profile named CICSUSER.

Define a different default CICS userid for each CICS region if any of the following considerations applies:

- 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))
```

The security administrator should always define the password for default userids and started tasks, instead of allowing it to default.

Each CICS region should use its own default user, as an aid to 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), 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 sign-on 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 77 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 terminal user begins to sign on. 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.

Note: If the default user's RACF profile specifies a non-zero TIMEOUT, that value does **not** apply to terminals that do not sign on.

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.

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

- Give the default user access to the region's APPLID. See “Authorizing access to the CICS region” on page 52.
- Give the default user access to the CICS-supplied transactions that are intended to be used by everybody. See the definitions in “Identifying CICS terminal users” on page 65, especially those transactions that are recommended for inclusion in the ALLUSER example group of transactions.

Authorizing access to MVS log streams

Ensure that you authorize the CICS region userid to write to (and create if necessary) the log streams that are used for its system log and general logs. You do this by granting the appropriate access authorization to log stream profiles in the LOGSTRM general resource class.

The level of authorization required depends on whether log streams are always explicitly defined to the MVS system logger:

- If CICS is expected to create log streams dynamically, give CICS **ALTER** authority to the relevant log stream profiles, and **UPDATE** authority to the relevant coupling facility structures.
- If all the log streams to which CICS writes are already defined to MVS, give CICS only **UPDATE** authority to the log stream profiles.
- Permit **READ** access to those users who need to read the CICS log streams.

For example, the generic profile in the following example could be defined to cover all the log streams referenced by the CICS region and identified by its region userid and applid:

```
RDEFINE LOGSTRM region_userid.** UACC(NONE)
```

If, however, you have multiple CICS systems sharing the same region userid, but with differing security requirements, include the applid in the generic profile, as follows:

```
RDEFINE LOGSTRM region_userid.applid.* UACC(NONE)
```

The following example allows the CICS region userid under which CICS is running to write journal and log records to log streams in the named coupling facility structure:

```
PERMIT IXLSTR.structurename CLASS(FACILITY) ACCESS(UPDATE)
       ID(region_userid)
```

The following examples give access to three categories of user:

```
PERMIT region_userid.applid.* CLASS(LOGSTRM) ACCESS(ALTER)
       ID(region_userid)
PERMIT region_userid.applid.* CLASS(LOGSTRM) ACCESS(READ)
       ID(authorized_browsers)
PERMIT region_userid.applid.* CLASS(LOGSTRM) ACCESS(UPDATE)
       ID(archive_userid)
```

In these examples, `region_userid` is the CICS region userid under which CICS is running, either as a started task or batch job. The identifier `archive_userid` is the userid under which an application program runs to purge old data from CICS logs when the data is no longer needed. The identifier `authorized_browsers` refers to the userids of users allowed to read log streams, but not purge data.

If several CICS regions share the same CICS region userid, you can make profiles more generic by specifying * for the *applid* qualifier.

The number of profiles you define depends on the naming conventions of the logs, and to what extent you can use generic profiling.

Authorizing access to CICS data sets

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

When authorizing access to CICS system data sets, choose appropriately from the following levels of access: READ, UPDATE, and CONTROL. Also define data set profiles with UACC(NONE) to ensure that only CICS region user IDs can access those data sets. For information about the CICS region userid, see “Specifying the CICS region userid” on page 38.

For CICS load libraries, 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)

Permit UPDATE access 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. See Table 3.

Table 3. Summary of generic data set profiles

Required access level	**Type of CICS data sets protected**
READ	Load libraries
UPDATE	Auxiliary trace; transaction dump; 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 46 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. Any of these methods enables 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  'CICSTS32.CICS.SDFHLOAD'  NOTIFY(cics_sys_admin_id) UACC(NONE)
PERMIT 'CICSTS32.CICS.SDFHLOAD' ID(cics_id1,...,cics_group1,..,cics_groupn)
                           ACCESS(READ)
ADDSD  'CICSTS32.CICS.SDFHAUTH'  NOTIFY(cics_sys_admin_id) UACC(NONE)
PERMIT 'CICSTS32.CICS.SDFHAUTH' ID(cics_id1,...,cics_group1,..,cics_groupn)
                           ACCESS(READ)
ADDSD  'CICSTS32.CICS.applid.**'  NOTIFY(cics_sys_admin_id) UACC(NONE)
PERMIT 'CICSTS32.CICS.applid.**'  ID(applid_userid) ACCESS(UPDATE)
ADDSD  'CICSTS32.CICS.applid.DFHXR*'  NOTIFY(cics_sys_admin_id) UACC(NONE)
PERMIT 'CICSTS32.CICS.applid.DFHXR*'  ID(applid_userid) ACCESS(CONTROL)
ADDSD  'CICSTS32.CICS.applid.DFHINTRA' NOTIFY(cics_sys_admin_id) UACC(NONE)
PERMIT 'CICSTS32.CICS.applid.DFHINTRA'  ID(applid_userid) ACCESS(CONTROL)
ADDSD  'CICSTS32.CICS.applid.DFHTEMP' NOTIFY(cics_sys_admin_id) UACC(NONE)
PERMIT 'CICSTS32.CICS.applid.DFHTEMP'  ID(applid_userid) ACCESS(CONTROL)
ADDSD  'CICSTS32.CICS.DFHCSD'          NOTIFY(cics_sys_admin_id) UACC(NONE)
PERMIT 'CICSTS32.CICS.DFHCSD' ID(cics_group1,..,cics_groupn) ACCESS(UPDATE)
/*
//
```

Figure 1. Example of a job 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 with profiles that give all CICS region groups (or userids) READ access.

You could also consider protecting all these data sets with one generic profile called 'CICSTS32.CICS.**'. However, you must strictly control who has read access to CICSTS32.CICS.SDFHAUTH, because it contains APF-authorized programs, and the profile protecting this data set **must** be defined with UACC(NONE). In Figure 1 all of the partitioned data sets are defined with UACC(NONE) and have an explicit access list.

Although CICS modules exist in libraries SYS1.CICSTS32.CICS.SDFHLPA and SYS1.CICSTS32.CICS.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 CICSTS32.CICS, 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 DDNAMEs 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

Note: The auxiliary trace data set, the transaction dump data set, and the MVS dump data set may contain sensitive information. Protect them from unauthorized access.

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

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

DFHINTRA
Transient data intrapartition data set

DFHTEMP
Temporary storage data set

DFHXRMSG
XRF message data set

The CICS system definition data set (CSD) is protected by a discrete profile to which all CICS groups have access. 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, define discrete or generic data set profiles as appropriate.

You must grant the CICS region user ID read access to the VSAM catalog for the DFHCSD file for the CICS system definition data set (CSD).

Authorizing access with the MVS library lookaside (LLA) facility

If you are using the library lookaside (LLA) facility of MVS, you can control a program's ability to use the LLACOPY macro. Authorize the CICS region's userid 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. For example:

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

Authorizing access to user data sets

When you have defined the RACF user ids for your CICS regions and given them access to the CICS system data sets, permit the user IDs to access the CICS **application** data sets with the necessary authority.

You must grant the CICS region user ID read access to each VSAM catalog for files for which CICS has file definitions installed and are to be either opened during CICS startup or at any time after.

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 'CICSTS32.CICS.appl1.dataset1'  ID(user or group)  ACCESS(READ)
PERMIT 'CICSTS32.CICS.appl1.dataset2'  ID(user or group)  ACCESS(READ)
PERMIT 'CICSTS32.CICS.appl2.dataset3'  ID(user or group)  ACCESS(UPDATE)
PERMIT 'CICSTS32.CICS.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 (non-RLS mode only). 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), or BWO(TYPECICS) in the integrated catalog facility (ICF) catalog, give the CICS region userid RACF ALTER authority to the data set or to the ICF catalog in which that data set is defined. If you do not, the OPEN command fails with message DFHFC5803. See the *CICS Recovery and Restart Guide* for guidance on using BWO.

Authorizing access to the temporary storage pools

You can control access by temporary storage (TS) servers to the TS pools in the coupling facility. Each TS server can be started as a job or started task. The name of the TS queue pool for a TS server is specified at server startup. For each TS pool there can be only one TS server running on each MVS image in the sysplex.

Two security checks are made against the TS server's userid—that is, the userid under which the job or started task is running. To ensure the server passes these checks, do the following:

- Authorize the TS server region to connect to the coupling facility list structure used for its own TS pool. This requires that the TS server userid has ALTER authority to a coupling facility resource management (CFRM) RACF profile called IXLSTR.*structure_name* in the FACILITY general resource class.

  For example, if the userid of the server is DFHXQTS1, and the list structure is called DFHXQLS_TSPRODQS, the following RACF commands define the profile and provide the required access:

  ```
  RDEFINE FACILITY IXLSTR.DFHXQLS_TSPRODQS UACC(NONE)
  PERMIT IXLSTR.DFHXQLS_TSPRODQS CLASS(FACILITY) ID(DFHXQTS1) ACCESS(ALTER)
  ```

To reduce security administration, use the same TS server userid to start each TS server that supports the same TS pool.

- Give the TS server's userid CONTROL access to the CICS RACF profile called DFHXQ.*poolname* in the FACILITY general resource class. This authorizes the TS server to act as a server for the named TS pool.

  For example, if the userid of the server is DFHXQTS1, and the pool name is TSPRODQS, the following RACF commands define the profile and provide the required access:

```
RDEFINE FACILITY DFHXQ.TSPRODQS UACC(NONE)
PERMIT DFHXQ.TSPRODQS CLASS(FACILITY) ID(DFHXQTS1) ACCESS(CONTROL)
```

See “System authorization facility (SAF) responses to the TS server” for information about the responses to the TS server.

Authorizing access to temporary storage servers

You can control access by CICS regions to the TS servers. A security check is made against the CICS region userid to verify that the region is authorized to use the services of a TS server. This check is made each time that a CICS region connects to a TS server.

Give each CICS region that connects to a TS server userid UPDATE access to the CICS RACF profile called DFHXQ.*poolname* in the FACILITY general resource class. This authorizes the CICS region to use the services of the TS server for the named TS pool.

For example, if the userid of a CICS region is CICSDAA1, and the pool name is TSPRODQS, the following RACF commands define the profile and provide the required access:

```
RDEFINE FACILITY DFHXQ.TSPRODQS UACC(NONE)
PERMIT DFHXQ.TSPRODQS CLASS(FACILITY) ID(CICSDAA1) ACCESS(UPDATE)
```

When a CICS region has connected to a TS pool, it can write, read, and delete TS queues without any further security checks being performed by the server. However, the CICS application-owning regions issuing TS API requests can use the existing mechanisms for TS resource security checking

System authorization facility (SAF) responses to the TS server

If the security profile for a TS pool cannot be retrieved, SAF neither grants nor refuses the access request. In this situation:

Access to the TS pool, either by a CICS region or by the TS server itself, is rejected 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 is active, it might retrieve a profile that does not permit access to the TS pool.

Access to the TS pool, either by a CICS region or by the TS server itself, is accepted 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 access request is allowed in this case because there is no evidence that you want to control access to the TS server.

Access is permitted to any TS server without a specific DFHXQ.*poolname* profile, or an applicable generic profile. No messages are issued to indicate this. To avoid any potential security exposures, you can use generic profiles to protect all, or specific groups of, TS servers. For example, specifying:

```
RDEFINE FACILITY (DFHXQ.*) UACC(NONE)
```

ensures that access is allowed only to TS servers with a more specific profile to which a TS server or CICS region is authorized.

Authorizing access to named counter pools and servers

You can control access by:

- Coupling facility data table (named counter) servers to named counter pools (see "Access to named counter pools")
- CICS regions to the named counter servers (see "Access to named counter servers").

Access to named counter pools

You can control access by named counter servers to the named counter pools in the coupling facility. Each named counter server can be started as a job or started task. The name of the named counter pool for a named counter server is specified at server startup. For each named counter pool there can be only one server running on each MVS image in the sysplex.

Two security checks are made against the named counter server's userid—that is, the userid under which the job or started task is running. To ensure the server passes these checks, do the following:

- Authorize the named counter server region to connect to the coupling facility list structure used for its own named counter pool. This requires that the named counter server userid has ALTER authority to a coupling facility resource management (CFRM) RACF profile called IXLSTR.*structure_name* in the FACILITY general resource class.

  For example, if the userid of the server is DFHNCSV1, and the list structure is called DFHNCLS_DFHNC001, the following RACF commands define the profile and provide the required access:

  ```
  RDEFINE FACILITY IXLSTR.DFHNCLS_DFHNC001 UACC(NONE)
  PERMIT IXLSTR.DFHNCLS_DFHNC001 CLASS(FACILITY) ID(DFHNCSV1) ACCESS(ALTER)
  ```

To reduce security administration, use the same named counter server userid to start each named counter server that supports the same named counter pool.

- Give the named counter server's userid CONTROL access to the CICS RACF profile called DFHNC.*poolname* in the FACILITY general resource class. This authorizes the named counter server to act as a server for the named counter pool.

  For example, if the userid of the server is DFHNCSV1, and the pool name is DFHNC001, the following RACF commands define the profile and provide the required access:

  ```
  RDEFINE FACILITY DFHNC.DFHNC001 UACC(NONE)
  PERMIT DFHNC.DFHNC001 CLASS(FACILITY) ID(DFHNCSV1) ACCESS(CONTROL)
  ```

  See "System authorization facility (SAF) responses to the named counter server" on page 51 for information about the responses to the CFDT server.

Access to named counter servers

You can control access by CICS regions to the named counter servers. A security check is made against the CICS region userid to verify that the region is authorized to use the services of a named counter server. This check is made each time that a CICS region connects to a named counter server.

Give each CICS region userid that connects to a named counter server UPDATE access to the CICS RACF profile called DFHCF.*poolname* in the FACILITY general resource class. This authorizes the CICS region to use the services of the named counter server for the named named counter pool.

For example, if the userid of a CICS region is CICSDAA1, and the pool name is DFHNC001, the following RACF commands define the profile and provide the required access:

```
RDEFINE FACILITY DFHNC.DFHNC001 UACC(NONE)
PERMIT DFHNC.DFHNC001 CLASS(FACILITY) ID(CICSDAA1) ACCESS(UPDATE)
```

When a CICS region has connected to a named counter pool, it can define, update, delete, get, rewind, and query named counters without any further security checks being performed by the server.

Note: Unlike shared temporary storage pools and coupling facility data table pools, named counters can also be accessed by batch application regions. Batch jobs are subject to the same security mechanisms as a CICS region.

System authorization facility (SAF) responses to the named counter server

If the security profile for a named counter pool cannot be retrieved, SAF neither grants nor refuses the access request. In this situation:

Access to the named counter pool, either by a CICS region or by the named counter server itself, is rejected 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 is active, it might retrieve a profile that does not permit access to the named counter pool.

Access to the named counter pool, either by a CICS region or by the named counter server itself, is accepted 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 access request is allowed in this case because there is no evidence that you want to control access to the named counter server.

Access is permitted to any named counter server without a specific DFHCF.*poolname* profile, or an applicable generic profile. No messages are issued to indicate this. To avoid any potential security exposures, you can use generic profiles to protect all, or specific groups of, named counter servers. For example, specifying:

```
RDEFINE FACILITY (DFHNC.*) UACC(NONE)
```

ensures that access is allowed only to named counter servers with a more specific profile to which a named counter server or CICS region is authorized.

Authorizing access to SMSVSAM servers

SMSVSAM is a data-sharing subsystem running on its own address space to provide the RLS support required by CICS.

| In a test environment you might wish to use the default action and allow any CICS
| region using VSAM RLS to connect to an SMSVSAM server. If you wish to protect
| this access, the RACF SUBSYSNM general resource class must be active and you
| must authorize each CICS region that connects to an SMSVSAM server to have

access to that server. This means granting access to the appropriate profile in the RACF SUBSYSNM general resource class.

For CICS regions using VSAM record-level sharing (RLS), access to SMSVSAM servers is controlled by RACF security checks. The security check is made against the CICS region userid to verify that the region is authorized to register with an SMSVSAM server.

The general resource class, SUBSYSNM, supports authorizations for subsystems that want to connect to SMSVSAM. The SUBSYSNM profile name is the name by which a given subsystem is known to VSAM. CICS uses its applid as its subsystem name; define a profile for the CICS applid in the SUBSYSNM resource to enable CICS to register the control ACB.

When CICS attempts to register the control ACB during initialization, SMSVSAM calls RACF to check that the CICS region userid is authorized to the CICS profile in the SUBSYSNM class. If the CICS region userid does not have READ authority, the open request fails.

For example, if the applid of a CICS AOR is CICSDAA1, and the CICS region userid (shared by a number of AORs) is CICSDA##, define and authorize the profile as follows:

```
RDEFINE SUBSYSNM  CICSDAA1  UACC(NONE) NOTIFY(userid)
PERMIT  CICSDAA1  CLASS(SUBSYSNM)  ID(CICSDA##)  ACCESS(READ)
```

You can use wildcard characters on the applid to specify more than one CICS region, for example:

```
PERMIT  CICSD%%%  CLASS(SUBSYSNM)  ID(CICSDGRP) ACCESS(READ)
```

Authorizing 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 this purpose, the APPLID of a CICS region is:

- The VTAM generic resources name if GRNAME is specified as a system initialization parameter
- The XRF generic APPLID if XRF=YES is specified as a system initialization parameter
- The generic APPLID if one is specified on the APPLID system initialization parameter
- The specific APPLID if only one is specified on the 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)
```

You need to define only one APPL profile name in the RACF database for all the CICS regions that are members of the same VTAM generic resources name. All sign-on verifications in a CICSplex, where all the terminal-owning regions have the same VTAM generic resources name, are made against the same APPL profile.

For MRO only, the APPLID is propagated from the terminal-owning region (TOR) to the other regions that the user accesses — for example, from the TOR to the application-owning region (AOR), and from the AOR to the file-owning region (FOR). As a consequence:

- You do not need to include users of the AOR and FOR in the APPL profiles for those regions.
- You can force users to sign on through a TOR, by denying access to other APPLIDs

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, consider activating profiles in the APPL class using RACLIST.

```
SETROPTS RACLIST(APPL)
```

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

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

Note:

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, ensure that authorized remote CICS regions can sign on to a CICS region protected in this way.

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

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), 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, ensure that the RACF database is shared, or define the VTAMAPPL profiles in the other system's RACF database.

The VTAMAPPL class must be activated using RACLIST for this protection to be in effect:

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

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

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

Controlling userid propagation

Jobs submitted from CICS to the JES internal reader without the USER operand being specified on the JOB statement run under the CICS region user ID. 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 region user ID from being propagated to these batch jobs by defining a profile in the PROPCNTL class where the profile name is the CICS regions user ID. For example, if the CICS region userID is CICS1, define a PROPCNTL profile named CICS1:

```
RDEFINE  PROPCNTL  CICS1
```

The PROPCNTL class must be activated using RACLIST for this protection to be in effect:

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

If the PROPCNTL class is already active, 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.

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 activated using RACLIST for this protection to be in effect:

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

Attention:

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.

You can use an EXEC CICS ASSIGN USERID command 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 *z/OS Security Server 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. Grant authorization 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)
- All userids used for transient data trigger level transactions
- All userids specified on the AUTHID or COMAUTHID parameters of a DB2 resource definition.

For more information about surrogate user checking, see Chapter 7, “Surrogate user security,” on page 111.

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.

Security-related system initialization parameters

There are several system initialization parameters that CICS provides for specifying your security requirements at the system level.

For full details of system initialization parameters, see the *CICS 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, and control of CICS security is determined by the other security-related system initialization parameters:

CMDSEC	**XAPPC**	**XPPT**
DFLTUSER	**XCMD**	**XPSB**
EJBROLEPRFX	**XDB2**	**XRES**
ESMEXITS	**XDCT**	**XRFSOFF**
PSBCHK	**XEJB**	**XRFSTME**
PLTPISEC	**XFCT**	**XTRAN**
RESSEC	**XHFS**	**XTST**
SECPRFX	**XJCT**	**XUSER**
SNSCOPE	**XPCT**	

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 sign-on cannot take place.

Note: Even if you have specified SEC=NO, with MRO bind-time security, the CICS region userid is sent to the secondary system, and bind-time checking is carried out in the secondary system. See "Bind-time security with MRO" on page 211 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*.

The options on the SECPRFX parameter are:

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 the CICS region userid when passing authorization requests to RACF.

To change these values employ an ICHRTX00 SAF preprocessing exit. For more information, see "Determining the userid of the CICS region" on page 293.

prefix
CICS prefixes the resource names with the specified prefix when passing authorization requests to RACF.

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) 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, 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 *z/OS Security Server RACF Security Administrator's Guide*.)

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 the *CICS System Definition 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 43.

EJBROLEPRFX
Specifies a prefix that is used to qualify the security role defined in an enterprise bean's deployment descriptor. The prefix is applied to the security role:

- when a role is defined to an external security manager
- when CICS maps a security role to RACF user ID

For more information about how the **EJBROLEPRFX** parameter is used to qualify security roles for enterprise beans, see *Java Applications in CICS*.

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 21, "Customizing security processing," on page 289.

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.

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

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 CICS is to check the remote link but not the remote user. The remote user is checked by specifying PSBCHK=YES.

RESSEC
Code RESSEC 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. RESSEC 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 the *CICS System Definition Guide*.

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

The sign-on 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 the CRTE transaction to connect to another system. For more information about using SNSCOPE, and the restrictions involved, see the*CICS 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 X*name* system initialization parameters.

The full list of the CICS resource classes is shown in Table 4, each with corresponding X*name* system initialization parameter.

Table 4. System initialization parameters for the CICS resource classes

System initialization parameter	Resource
XAPPC={**NO**|YES}	APPC partner-LU verification

Table 4. System initialization parameters for the CICS resource classes (continued)

System initialization parameter	Resource
XCMD={**YES**|name|NO}	EXEC CICS system commands EXEC CICS FEPI system commands
XDB2={**NO**|name}	CICS DB2 resources
XDCT={**YES**|name|NO}	Transient data destinations
XEJB={**YES**|NO}	Security roles is enabled
XFCT={**YES**|name|NO}	Files
XHFS={**YES**|NO}	z/OS UNIX files managed by z/OS UNIX System Services
XJCT={**YES**|name|NO}	Journals and logs
XPCT={**YES**|name|NO}	Started transactions and EXEC CICS commands: COLLECT STATISTICS TRANSACTION DISCARD TRANSACTION INQUIRE TRANSACTION SET TRANSACTION
XPPT={**YES**|name|NO}	Programs
XPSB={**YES**|name|NO}	DL/I program specification blocks (PSBs)
XRES={**YES**|name|NO}	CICS document templates
XTRAN={**YES**|name|NO}	Attached transactions
XTST={**YES**|name|NO}	Temporary storage entries
XUSER={**YES**|NO}	Surrogate user checking DB2 AUTHTYPE checking

Note:

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

If you specify YES for any of the X*name* system initialization parameters where RACF is used to manage resource security, CICS uses the default class name for that parameter. See “RACF classes for CICS resources” on page 24 for a list of these.

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 5. Specifying external security with default resource classes

System initialization parameter	Effect
SEC=YES	CICS initializes 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 a second example, the effect of specifying SEC=YES with the same three associated resource class parameters specified as X*name*=*username* is shown in Table 6.

Table 6. 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, XDB2, and XEJB) X*name*=YES is the default in the system initialization parameters, and CICS will use the default classnames, for example, GCICSTRN. 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.

The system initialization parameter XHFS is an exception to this process. Access controls for z/OS UNIX files are not managed directly by RACF, so they do not require individual RACF profiles, even if XHFS=YES is specified. Access controls for z/OS UNIX files are specified in z/OS UNIX System Services, which makes use of RACF to manage user IDs and groups, but keeps control of the permissions set for the files. If you are using access control lists (ACLs) for z/OS UNIX files, the RACF class FSSEC must be active.

For guidance on the syntax of external security system initialization parameters, see the the *CICS 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 65 and Chapter 5, "Transaction security," on page 81 for information about specifying resource and command security for transactions.

XAPPC, XEJB, XHFS, and XUSER

The syntax of the XAPPC, XEJB, XHFS, 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. If XAPPC=YES is specified and the APPCLU class is not activated in RACF, CICS fails to initialize. For more information on what happens in these circumstances, see "CICS initialization failures related to security" on page 308.

XAPPC enables RACF LU6.2 bind-time (also known as APPC) security. For more information, see "Bind-time security with LU6.2" on page 157.

For more information on the APPCLU class, see “Defining profiles in the APPCLU general resource class” on page 158.

XEJB specifies whether support of security roles is enabled. For more information, see *Java Applications in CICS*.

XHFS activates access control for Web client access to z/OS UNIX files in the CICS region. For more information, see “Security for z/OS UNIX files” on page 104.

XUSER activates surrogate user security, and AUTHTYPE checking for DB2. For more information, see Chapter 7, “Surrogate user security,” on page 111. If XUSER=YES is specified and the SURROGAT class is not activated in RACF, CICS fails to initialize.

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 topic.

Before you define a profile, activate the relevant classes, using the SETROPTS CLASSACT and SETROPTS GENERIC commands, as described in “Summary of RACF commands” on page 29.

To ensure the least interruption to actual business processes, work in a test region first.

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        XDB2=NO
                 XPSB=YES        XDCT=NO
                                 XHFS=NO
                                 XJCT=NO
                                 XPCT=NO
                                 XPPT=NO
                                 XRES=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 24.)

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 "Summary of RACF commands" on page 29.

To ensure the least interruption to actual business processes, work in a test region first.

Note: The following examples assume that the CICS region userid is CICS1, and that SECPRFX=YES.

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         XDB2=NO
                       XPSB=YES         XDCT=NO
                                       XHFS=NO
                                        XJCT=NO
                                        XPCT=NO
                                        XPPT=NO
                                       XRES=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 24.)

Using installation-defined classes without prefixing

To set up external security for transactions, files, and PSBs in installation-defined classes, without prefixing, take the steps described in this topic.

For an example of how to define installation-defined classes (T$USRTRN and G$USRTRN) for the XTRAN parameter, see the IBM-supplied sample, DFH$RACF, in CICSTS32.CICS.SDFHSAMP. See also "Specifying user-defined resources to RACF" on page 294.

Before you define a profile, activate the relevant classes, using the SETROPTS CLASSACT and SETROPTS GENERIC commands, as described in "Summary of RACF commands" on page 29.

To ensure the least interruption to actual business processes, work in a test region first.

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 *z/OS Security Server RACF System Programmer's Guide*.

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          XDB2=NO
                     XPSB=$USRPSB          XDCT=NO
                                           XHFS=NO
                                           XJCT=NO
                                           XPCT=NO
                                           XPPT=NO
                                           XRES=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 24.)

Chapter 4. Verifying CICS users

To protect resources from unauthorized access, CICS must be able to identify users of the system when they invoke transactions.

Identifying CICS terminal users

If you are running CICS with RACF security checking, you control users' access to CICS resources through levels of authorization you define in 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 is defined in the CICS segment of the RACF user profile. See “Obtaining CICS-related data for a user” on page 77 for more information about CICS terminal-user data, and how CICS obtains it.

The sign-on process

When users log-on to CICS through VTAM, 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 sign-on

Users can explicitly sign on either by using the CICS-supplied transaction CESN or by using an installation-provided sign on transaction which uses the **`SIGNON`** command.

OIDCARD users can use CESN to sign on if the card reader supports the DFHOPID identifier (AID). If it does not, use your own installation-provided sign-on transaction.

- For programming information about the SIGNON command, see the *CICS Application Programming Reference*.
- For information about CESN, see *CICS Supplied Transactions*.

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

`Scoping`
After the sign-on 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.

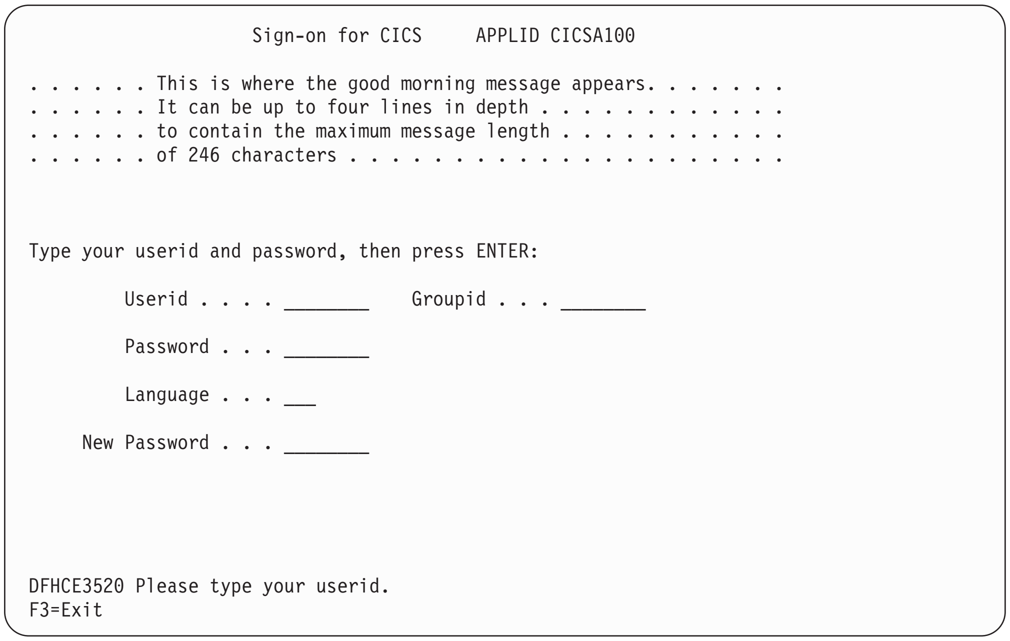
`Identification`
CICS calls RACF with the supplied userid to confirm that a profile has been defined for the user.

`Verification`
CICS passes information to RACF to verify that the user is genuine. For RACF this is 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.

The CESN transaction does not warn users when a password is about to expire. To display expiry warnings to users, you must replace CESN with your own transaction that calls a custom sign-on program. This program can use the **VERIFY PASSWORD** API command to return expiry information from RACF.

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 of the CICS address space that may be taken.

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



```
                    Sign-on for CICS     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 application named in the APPLID or GRNAME system initialization parameter. RACF does this by checking the access list of the CICS application profile defined in the RACF APPL resource class. (See “Authorizing access to the CICS region” on page 52 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 68.

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

Explicit sign-on, with the CESN transaction, or the SIGNON command, is performed by the user at the port of entry.

Table 7. Explicit and implicit signons

Phase	Explicit	Implicit
Scoping	Yes	No
Identification	Yes	Yes
Verification	Yes	No except with ATTACHSEC(IDENTIFY)
Authorization	Yes	Yes

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

The sign-off process

The sign-off process dissociates a user from a terminal where the user had been previously signed on. The user can explicitly sign off using the CESF transaction or an installation-provided transaction that uses the SIGNOFF command. If the attributes of the signed-on user include a non-zero value for TIMEOUT, an implicit sign-off occurs if this interval expires after a transaction terminates at this terminal.

When the time-out period expires, if the default GNTRAN=NO is specified, CICS performs an immediate signoff. If GNTRAN specifies a transaction-id to be scheduled and that transaction performs a signoff, the action CICS takes depends on the SIGNOFF option specified in the terminal's TYPETERM resource definition.

An exceptional case is that the goodnight transaction is not used for the user of a CRTE session. A surrogate user whose time expires is signed off, losing the security capabilities the terminal previously had. Message DFHSN1200 is sent to the CSCS log, and indicates what has happened.

For more information about the use of system initialization parameter GNTRAN, see “Goodnight transaction” on page 68. 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 log on again, CICS deletes the terminal entry (TCTTE) from the TCT. For information about CICS autoinstall, see the *CICS Resource Definition Guide*.

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

Explicit sign-off

Explicit sign-off removes the user's scoping. The user must be explicitly signed on before signing off with the CESF transaction or the SIGNOFF command. 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 sign-on attempt subsequently fails. It is not recommended that CESN be used for the Goodnight transaction.

Implicit sign-on and implicit sign-off

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

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 USERID option on the ASSIGN command obtains the userid, and the VERIFY PASSWORD command 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 sign-off a user terminal. The goodnight transaction is not available for a surrogate terminal that is timed out during a CRTE session. Sign-off occurs with a loss of the security capabilities the terminal previously had, leaving a DFHSN1200 message in the log.

Controlling access to CICS from specific ports of entry

During sign-on processing, CICS issues 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. Autoinstalled consoles that are using automatic sign-on are treated as though they have a preset security definition (see “Preset terminal security” on page 72). 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 access, or at 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.

Note: The TERMINAL class does not control access from MVS consoles. These are controlled by the CONSOLE resource class. See “Console profiles” on page 71.

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 70 for more information about the SETROPTS command for terminals, and about the TERMUACC|NOTERMUACC option on groups.

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. You can use associated port of entry profiles to control whether a user can sign on at a particular device.

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 *z/OS Security Server RACF Security Administrator's Guide*.

- Creating a profile in the TERMINAL or GTERMINL class
- Preventing the use of an undefined terminal
- Restricting specific groups of users to specific terminals
- Restricting the days or times when a terminal can be used
- Using a security label to control a terminal.

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

TERMINAL
For defining a profile of an **individual** terminal.

GTERMINL
For defining a profile of a **group** of terminals.

Note: For a GTERMINL profile, RACF always uses an in-storage profile, 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 a profile of an individual terminal

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 “Summary of RACF commands” on page 29. 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 a profile of a group 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 or GTERMINAL class

For CICS, the terminal profiles to define to RACF in the TERMINAL or GTERMINL class are used only for VTAM terminals. The name of the profile is the value of the NETNAME that is specified in the RDO terminal definition or autoinstall. It is not possible to use TERMINAL profiles with non-VTAM terminals.

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 terminal.

Note: Before issuing the SETROPTS TERMINAL(NONE) command, 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 option 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 any 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 a user in the group needs explicit authorization to use a terminal. To enable a group with the NOTERMUACC option to access terminals, you must add group userid to the access list of the appropriate TERMINAL or GTERMINL profile.

Conditional access processing

RACF can give you a greater authority to access 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, issue the following command:

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

Note:

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 *z/OS Security Server RACF Security Administrator's Guide*.

Console profiles

Use the CONSOLE resource class to define profiles that control user access to a console.

If the CONSOLE class has been activated, you can control whether a user is allowed to sign on to a console. Console protection is implemented in a method similar to that for protecting terminals, with the exception of the following, which were discussed in “Overriding the SETROPTS TERMINAL command” on page 70:

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 *z/OS MVS Planning: Operations* manual for the effects of console protection on MVS consoles.

The profile used in the console class is the console name or number. For example:

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

The user must have READ access to the console name to sign-on at a console. The following example shows how user CICSOPR would be permitted to sign on to 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 sign-on 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.

Auditing sign-on and sign-off activity

RACF can log all sign-on and sign-off activity to SMF, including any invalid or unsuccessful sign-on attempts. You can only properly interpret the logging of unsuccessful sign-on attempts by also recording successful sign-ons. For example, if a user makes one or two unsuccessful attempts followed immediately by a successful sign-on, the unsuccessful sign-ons 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 sign-on activity being recorded, may well be cause for a security concern that warrants investigation.

Recording the successful sign-on and sign-off activities 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 DFHSN1118 and DFHSN1119 are sent to CSCS. The DFHSNxxxx messages include reason codes that indicate the precise nature of the invalid sign-on attempt.

Preset terminal security

For some terminals, and MVS consoles when used as CICS terminals, it is appropriate to use preset terminal security as an alternative to terminal user security.

A terminal becomes a preset security terminal when you specify the USERID attribute on the TERMINAL resource definition.

There are two types of preset security for consoles:

1. Normal preset security (the same as preset security for other terminals)
2. Automatic preset security

Normal preset security

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

You can also use the normal preset 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. The terminal remains signed on as long as it is installed, and no explicit sign-off can be performed against it. If the userid associated with a display terminal with preset-security has been authorized to use any sensitive transactions, 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 170 (LU6.2 security) and "Preset-security terminals and transaction routing" on page 219 (MRO security).

Automatic preset security for consoles

Automatic preset security applies only to console definitions. CICS automatic preset security allows you to associate the userid, which MVS has already verified through RACF, with the CICS definition for the console.

Instead of specifying an actual user id on the TERMINAL definition, you specify a special value (*FIRST or *EVERY), to indicate that CICS is to use the user id passed by MVS on the MODIFY command. This means that CICS implicitly signs on the console when it is being installed, and optionally on each input message, instead of a subsequent sign-on of that console by a user. Particularly in the context of autoinstalled consoles, this allows you to gain the advantage of preset security without having to define the user id/console relationship in the CICS terminal definition. Thus, console users do not have to sign-on with passwords in the clear to each CICS region.

You can use this automatic form of preset security on predefined consoles, autoinstalled consoles, and consoles installed with the **CREATE TERMINAL** command.

For example, to define a console with automatic preset-security, which is checked, and altered (if necessary) on every MODIFY, use CICS (CEDA) commands as follows:

```
CEDA DEFINE TERMINAL(cics_termid)
            CONSNAME(consolename) USERID(*EVERY)
            TYPETERM(cics_typeterm)
```

To define a console with automatic preset-security that is defined on the first valid MODIFY command only, use CICS (CEDA) commands as follows:

```
CEDA DEFINE TERMINAL(cics_termid)
            CONSNAME(consolename) USERID(*FIRST)
            TYPETERM(cics_typeterm)
```

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. Automatic preset security for consoles does not carry the same risks because the console user is associated with their true identity (verified by RACF). For this reason, no checking is carried out when a console device is defined to CICS with either USERID(*EVERY) or USERID(*FIRST).

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, and is specified by the RACF SURROGAT resource class. The CICS *userid.*DFHINSTL resource can be defined in the SURROGAT resource class for authorization 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 performing the installation. See Chapter 7, “Surrogate user security,” on page 111 for more information.

The CEDA command checks the authority of the user to install preset terminals. Consider, therefore, whether to restrict the following functions with a view to controlling who can define and install terminals with preset security:
- The CEDA transaction
- The SURROGAT resource class
- 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. However, 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, restrict its use to authorized users. This gives you control over who can define resources, such as terminals, to CICS. See Chapter 10, “Security for CICS-supplied transactions,” on page 139 for information about protecting CICS-supplied transactions.

Using the SURROGAT resource class

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.

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 58 for information about defining this 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 an initial or 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. 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 CEDA LOCK and CEDA UNLOCK commands. For more information about the LOCK and UNLOCK commands, see the *CICS Resource Definition Guide*.

Other preset security considerations

If you intend to use preset security, consider these additional topics:
- Autoinstall models
- Sessions with preset security
- Terminals defined in the TCT

Autoinstall models

If you are using autoinstall models with preset security, CICS makes the same surrogate authorization check as for ordinary terminals when the model is installed. It does not check surrogate authorization when the autoinstall model is used to perform autoinstall for a device. Also, CICS does not make a surrogate authorization check when installing models defined with automatic preset security for consoles.

If an autoinstall model with a preset userid becomes invalid (for example, if the userid is revoked), any attempt to install a terminal with the model fails.

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, BSAM terminals), the userid is also defined in the TCT, and, when CICS initializes, it signs on these terminals. If the sign-on fails (for example, if the userid is revoked), the terminal is put out of service. If the userid later becomes valid (for example, if it is resumed), setting the terminal in service results in a successful sign-on. CICS does not perform a surrogate user check for these terminals.

Using an MVS system console as a CICS terminal

If you intend to use an MVS system console as a CICS terminal, you may need authorization to use the MVS MODIFY command. This is done using the OPERCMDS resource class.

We recommend that you specify automatic preset security on the console's CICS terminal definition, so that the console user obtains the correct level of authority without explicitly performing a CICS signon (which exposes the password).

If preset security is not defined, console users must sign on to get authority different from the default user. In this case, the password can generally be seen on the console and 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 Transaction Server for z/OS 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). Provide a response using 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 sign-on process, a REPLY command with a null operand is required. That is, enter:

```
  REPLY nn,
```

with nothing after the comma, where `nn` is the number of the message corresponding to the reply.

You can authorize TSO users to use the TSO CONSOLE command. (For information on this command, see *z/OS TSO/E System Programming Command Reference*.) These users must be defined to CICS as consoles, using the CONSNAME option of the DEFINE TERMINAL command, or be supported by autoinstall for consoles, as described in the *CICS Resource Definition Guide*.

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 (the normal preset security for CICS terminals, or automatic preset security for consoles). When the TSO user uses the CONSOLE command, that user's userid, by default, becomes a console name. (But it can be changed to be any name using the CONSNAME(*name*) option on the TSO CONSOLE command). This console name can then be used as a CICS terminal if there is a corresponding TERMINAL definition (or one can be autoinstalled) with the CONSNAME option in CICS. If another name has been specified, that name is the one CICS uses to communicate with the console. For example, it is possible for one TSO user to use a name that is the same as another TSO user's ID.

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. We recommend that you use automated preset security for the following reasons:

- It means that TSO users do not have to sign on, which may expose their ID and password on the log.

- It means that you do not have to define a relationship, in a CICS definition, between a console name and a user, which may change frequently or become invalid.
- It allows you to define one autoinstall model which covers the majority of your console definitions and gives each user the correct level of preset security.

To define automatic preset security, specify USERID(*EVERY) to ensure that the correct user ID is signed on for every command, or USERID(*FIRST) to sign on the console using the userid that first issues a MVS MODIFY command to CICS, and retain this for subsequent commands.

- Choose USERID(*FIRST) if use of a console is restricted to one or more users who have similar security characteristics to CICS using RACF, and you don't use the user ID as an identifier in applications.
- Use USERID(*EVERY) if you need to ensure that each input request is tested to be sure that the console user has the correct security level. You should be aware that checking the user ID imposes an overhead on MODIFY, and changing the preset userid imposes another overhead which is equivalent to the console user signing on using CESN.

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.

This section explains how the data is obtained, for the default user and terminal users signing on.

Obtaining CICS-related data for the default user

When implicitly signing on the CICS default user during initialization, CICS obtains attributes 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 is present for the default userid, RACF returns this data to CICS. See “The CICS segment” on page 17 for details of the information that you can define in the CICS segment. See “The LANGUAGE segment” on page 20 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 sign-on for a CICS terminal user, CICS obtains the 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 is present for the terminal user, RACF returns this data to CICS. See "The CICS segment" on page 17 for details of the information that you can define in the CICS segment. See "The LANGUAGE segment" on page 20 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" on page 77.)

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 `SIGNON` command, if supported by CICS. A **supported** national language is a **valid** national language that has been specified in the NATLANG system initialization parameter and has the corresponding message definitions. See the *CICS 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 in "Obtaining CICS-related data for the default user" on page 77.

See Appendix A, "National Language," on page 357 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 an administrative unit. You can define such users to RACF as **groups** of individual users who have similar access requirements to CICS system resources. See the *z/OS Security Server 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 group, as follows:

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

Use the ALTUSER command to add CICS data for an existing userid. See “The CICS segment” on page 17 for details of the CICS optional data.

See the *z/OS Security Server RACF Command Language Reference*for the full syntax of these commands.

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))
```

Note:

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.

Support for mixed case passwords

When the security manager used with CICS supports the use of mixed case passwords, such as the z/OS Security Server (RACF) for z/OS 1.7, CICS Transaction Server for z/OS, Version 3 Release 1 does not convert passwords to uppercase before passing them to the security manager.

When a password is entered using one of the following mechanisms:

API commands
The following commands have a PASSWORD option:
CHANGE PASSWORD
VERIFY PASSWORD
SIGNON

The signon transaction (CESN)
The transaction offers two fields where passwords can be entered:
Password
New password

CICS can handle the password in one of two ways, depending upon whether the external security manager used with CICS supports mixed case passwords.

- If the security manager supports mixed case passwords, then CICS passes the password to the security manager unchanged.
- If not, then CICS converts the password to uppercase before passing it to the security manager.

To turn support for mixed case passwords on, see “Summary of RACF commands” on page 29.

National language and non-terminal transactions

When a user specifies a national language during sign-on, the sign-on option overrides the language specified in the user's RACF CICS segment. The language thus specified is set for the time 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 sign-on.

However, if a transaction uses the START command to start another transaction, the national language attribute for the started transaction 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 sign-on).

If the national language of the original terminal is required, the terminal's national language can be inquired about before the START command is issued. The information can then be passed as data in the START command for the use of the transaction that has been started.

Chapter 5. Transaction security

Transacation security (also known as *attach-time security*, and *transaction-attach security*) ensures that users that attempt to run a transaction are entitled to do so.

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 "Security-related system initialization parameters" on page 56.)

SECPRFX
Specify SECPRFX=YES if your transaction profiles are defined to RACF with a prefix that corresponds to the userid of the CICS region.

Specify SECPRFX=*prefix* if your transaction profiles are defined to RACF with any other prefix. (For more information, see "Security-related system initialization parameters" on page 56.)

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 "RACF classes for CICS resources" on page 24 for details of these resource classes.)

If you specify a resource class name, CICS uses the name you specified, prefixed with T for the resource class, and G for the grouping class.

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 on page 82 is an example which shows the main elements of CICS transaction security.

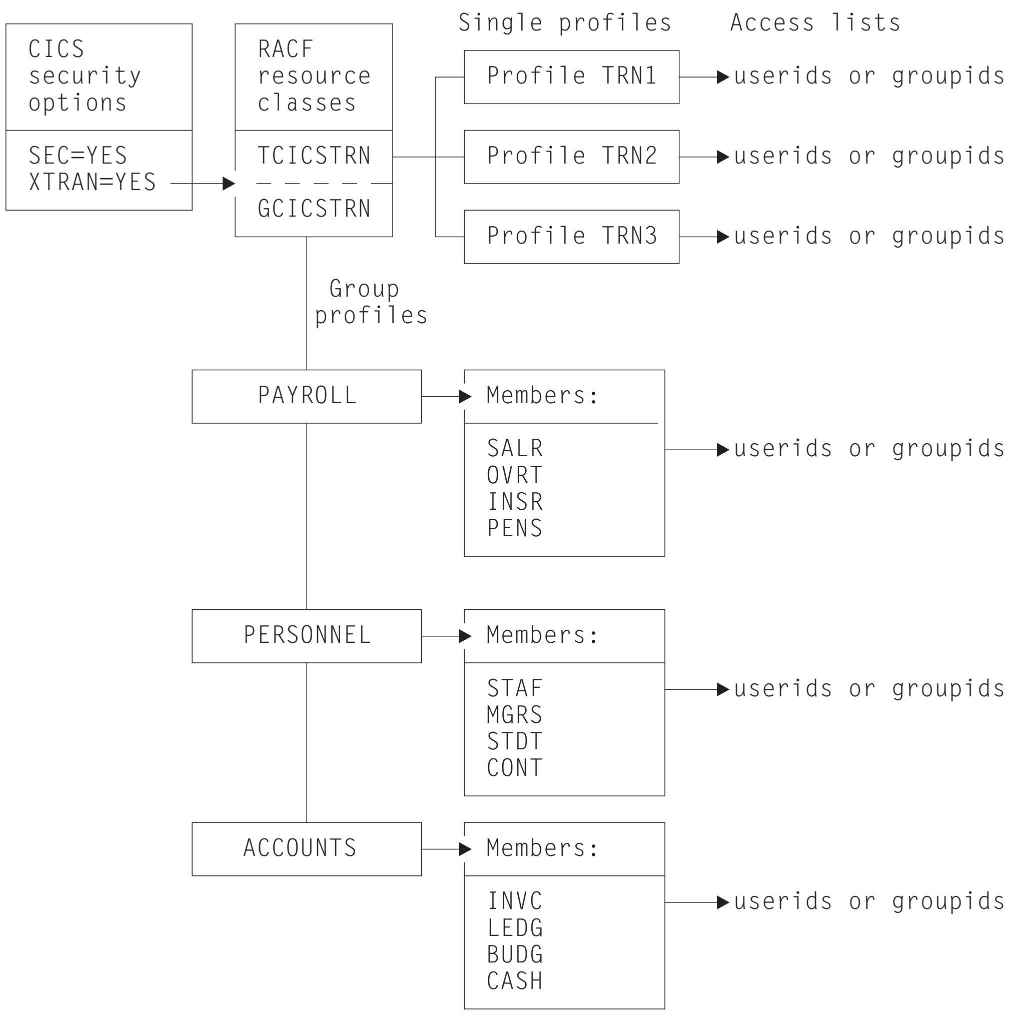


Figure 3. An example of the main elements of CICS transaction security.

In this example:

- *The following system initialization parameters are specified:*

  ```
  SEC=YES
  XTRAN=YES
  ```

  Because `XTRAN=YES` is specified, the resource class name is TCICSTRN and the grouping class name is GCICSTRN.
- *The resource class TCICSTRN contains profiles TRN1, TRN2, and TRN3.*
- *The grouping class GCICSTRN contains the following group profiles:*
  - *PAYROLL (members SALR, OVRT, INSR, and PENS)*
  - *PERSONNEL (members STAF, MGRS, STDT, and CONT)*
  - *ACCOUNTS (members INVC, LEDG, BUDG, and CASH)*

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 24.)

Defining transaction profiles to RACF

For those CICS regions running with transaction security checking, define transaction profiles for all transactions that need to be protected from unauthorized access. You can define these profiles either in the default transaction resource classes, or in installation-defined classes that you have added to the RACF class descriptor table. (See “RACF classes for CICS resources” on page 24 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 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 instead 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 you cannot group classes 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 any of your transactions to RACF, you can specify universal access as follows:

```
RDEFINE TCICSTRN ** UACC(READ)
```

You then need to define to RACF only those transactions that require more restrictive security.

Note: If you use a profile like that described above, define new profiles to RACF before installing 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 earlier 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(*))
```

Note:

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 uses only the console and terminal ports of entry.

Protecting the 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 “Using an MVS system console as a CICS terminal” on page 75.

Authorization failures and error messages

If a terminal user tries to initiate an unauthorized transaction, 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 normally issues an ICH408I message to the CICS region's job log and to the security console (the console defined for routing code 9 messages). However, if the transaction has been defined to RACF with LOG(NONE), no ICH408I message is issued. For a description of the ICH408I message, see the *z/OS Security Server RACF Messages and Codes* manual.

For more information on resolving authorization problems, see Chapter 22, “Problem determination in a CICS-RACF security environment,” on page 301.

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 *z/OS Security Server RACF Auditor's Guide*.

Protecting non-terminal transactions

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 CEDA transaction, the CEDA DEFINE TDQUEUE, the EXEC CICS CREATE, and the ATIUSERID option of the EXEC CICS SET command establish security for non-terminal transactions started by a transient data trigger level. The user issuing the SET, INSTALL, or CREATE 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 attribute of the transient data queue definition.

PLT programs

If PLT programs are to be executed during CICS startup, CICS performs a surrogate user security check for the region userid. See “Defining user profiles for CICS region userids” on page 40. This check determines whether the CICS job is authorized to be the surrogate of the userid specified on 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).

If the PLTPIUSR parameter is not specified, the PLT programs are run under the CICS region userid when the PLTPISEC=NONE option is defined. No surrogate check is required for this. If your PLT programs issue START commands, the jobstep userid has surrogate authority to start them when no userid is coded. Note that the starter always has surrogate authority to itself. When the started transaction starts up, another check is made to see if the userid has authority to attach the transaction and access this transaction in the TCICSTRN class. Rather than giving the jobstep access to additional resources, you can use the PLTPIUSR and PLTPISEC parameters.

During shutdown, CICS runs PLT programs under the authority of the userid for the transaction that requested 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.

Chapter 6. Resource security

This topic describes resource security, which ensures that users that attempt to use CICS resources are entitled to do so.

Introduction to resource security

Transaction security controls access to CICS transactions. Resource security provides a further level of security, by controlling access to the resources used by the CICS transactions. The implication of this is that a user who is authorized to invoke a particular CICS transaction may not be authorized to access files, PSBs, or other general resources used within the transaction. Unlike transaction security, which cannot be turned 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. Each resource is described briefly in Table 8, with the associated CICS system initialization parameter that you use to specify the RACF class name. For comprehensive information about application programming commands and system programming commands associated with each system initialization parameter, see Appendix B, “Resource and command check cross reference,” on page 359.

Note that no authorization processing is done for BMS commands.

Table 8. General resource checking by CICS

CICS parameter	General resource protected	Further information
XAPPC	Partner logical units (LU6.2).	Chapter 12, “Implementing LU6.2 security,” on page 157.
XCMD	The subset of CICS application programming commands that are subject to command security checking. EXEC CICS FEPI system commands are also controlled by this parameter.	Chapter 8, “CICS command security,” on page 119

Table 8. General resource checking by CICS (continued)

CICS parameter	**General resource protected**	**Further information**
XDB2	DB2 resource classes for DB2ENTRY are specified to CICS on the XDB2 system initialization parameter.	"Resource classes for DB2ENTRYs" on page 28
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.	"Security for transient data" on page 91.
XEJB	Enterprise bean methods	Security roles
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.	"Security for files" on page 93.
XHFS	z/OS UNIX files managed by z/OS UNIX System Services. This is a special case, because access controls for z/OS UNIX files are specified in z/OS UNIX System Services, so z/OS UNIX files do not require individual RACF profiles. No application programming commands or system programming commands are associated with this resource.	"Implementing security for z/OS UNIX files" on page 105.
XJCT	CICS system log and general logs. Define profiles in the journal class to control who is allowed to access CICS journals on CICS log streams.	"Security for journals and log streams" on page 94.
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.	"Security for started and XPCT-checked transactions" on page 95.
XPPT	CICS application programs. Define profiles in the program class to control who is allowed to access CICS application programs.	"Security for application programs" on page 98.
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.	"Security for program specification blocks" on page 107.
XRES	CICS document templates (DOCTEMPLATE resource definitions). Define profiles in this resource class to control who is allowed to access document templates.	"Security for CICS document templates" on page 101.
XTRAN	CICS transactions.	Chapter 5, "Transaction security," on page 81.
XTST	CICS temporary storage destinations. Define profiles in the temporary storage class to control who is allowed to access CICS temporary storage queues.	"Security for temporary storage" on page 99.
XUSER	Surrogate user security.	Chapter 7, "Surrogate user security," on page 111.

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 82.

For most simple (or single-function) transactions, this extra layer of security is not necessary. For example, if the transaction is designed to enable the terminal user to update a 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 a complex transaction that offers users a choice of functions, or you are unsure about all the options available within a transaction, you may want to add the extra layer of security to restrict access to the data as well as to the transaction. Before implementing resource security checking, 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 each CICS command (or in the case of z/OS UNIX files, each operation) that applies to one of the following resources, for which you have requested security, using an X*name* resource class parameter:

- extrapartition and intrapartition transient data destinations (XDCT parameter)
- file-control-managed VSAM and BDAM files (XFCT parameter)
- system logs and general log (XJCT parameter)
- started transactions and EXEC CICS commands: COLLECT STATISTICS TRANSACTION, DISCARD TRANSACTION, INQUIRE TRANSACTION, INQUIRE REQID, SET TRANSACTION, and CANCEL (XPCT parameter)
- application programs (XPPT parameter)
- CICS document templates (XRES parameter)
- temporary storage destinations (XTST parameter)

Consider this example:

The system initialization parameters include:

```
SEC=YES
XDCT=NO
XFCT=YES
XTRAN=YES
XTST=YES
```

Transaction TRN1 contains the following EXEC CICS commands:
- 4 file control READ commands
- 1 file control WRITE command
- 2 transient data WRITEQ commands
- 1 temporary storage WRITEQ command

When TRN1 executes, the seven calls are made to RACF:
- 1 at transaction attach, because XTRAN=YES is specified
- 5 for file control access, because XFCT=YES is specified
- 1 for temporary storage access, because XTST=YES is specified

RACF is not called for transient data access, because XDCT=NO is specified.

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). Design your CICS applications 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

Except when processing certain security commands (see Chapter 9, “Security checking using the QUERY SECURITY command,” on page 129), 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 *z/OS Security Server 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 those accesses 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: Specify the NOTIFY option, if you want to be notified at once when access is denied to a user.

Security for 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 from 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 if you define profiles with 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, define profiles only for:

- intrapartition transient data queues
- extrapartition transient data queues.

Do not define profiles for indirect transient data queues; CICS directs all requests for an indirect queues to another queue, which can be extrapartition, intrapartition, or remote. The redirection can also be to another indirect queue.

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 queue, 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:

```
TDQ definition: DEFINE TDQUEUE(QID1)
                       TYPE(EXTRA)
                       TYPEFILE(OUTPUT)
                       RECORDSIZE(132)
                       BLOCKSIZE(136)
                       RECORDFORMAT(VARIABLE)
                       BLOCKFORMAT(UNBLOCKED)
                       DDNAME(CICSMSGS)
                       GROUP(DFHDCTG)

                DEFINE TDQUEUE(QID2)
                       TYPE(INDIRECT)
                       INDIRECTNAME(QID1)
                       GROUP(DFHDCTG)

 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).

For information about the access authorization levels for system programming commands which apply to transient data queues, see Appendix B, “Resource and command check cross reference,” on page 359.

CICS-required destination control table entries

CICS itself uses a number of queues. These queues are defined in the sample group, DFHDCTG. If you want to protect access to these definitions 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 queues: CPLI, CSSL, or CCSO. Therefore, if you use the definitions as supplied, you need define to RACF only the queue names CPLI, CSSL, and CCSO, 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, CICS derives the userid associated with the triggered transaction from the following sources:

- The USERID parameter specified on the intrapartition transient data resource definition (DESTFAC=FILE).

- The userid associated with the terminal (for queues that have been defined with a destination facility of terminal) (DESTFAC=TERMINAL). This can be the CICS default userid if no user is signed on at the terminal.
- The link userid on the connection definition (for queues that have been defined with a destination facility of system) (DESTFAC=SYSTEM).

Security for 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:

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 from 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 if you define profiles with 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.

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.

For information about the access authorization levels for system programming commands which apply to files, see Appendix B, "Resource and command check cross reference," on page 359.

Security for journals and log streams

The CICS log manager provides facilities to write to and read from:

- The CICS system log
- The CICS general logs, which comprise user journals, forward recovery logs, and autojournals

The system log is used only for recovery purposes—for example, during dynamic transaction backout, or during emergency restart. Do not use it for any other purpose. Do not, therefore, write to it from a user application using the **WRITE JOURNALNAME** command.

CICS uses journal identifier **DFHLOG** for its primary system log. Do not permit user transactions to write to this. You can prevent them doing so by using the following command to define the system log in the JCICSJCT class, without any access list:

```
RDEFINE  JCICSJCT  DFHLOG  UACC(NONE) NOTIFY(sys_admin_userid)
```

In addition to the automatic journaling and forward recovery logging 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 WRITE JOURNALNAME command.

Users needing to write journal records must have authority to write to the JOURNALNAME (as defined in JCICSJCT). CICS calls RACF to perform a security check only for attempts to access a user journal by a CICS API command, and not for the journaling it performs in response to journaling options in the file resource definition. 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:

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 to identify the profiles.

   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(JRNL001, JRNL002, ....)
                      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 if you define profiles with 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.

For information about the access authorization levels for system programming commands which apply to journals, see Appendix B, “Resource and command check cross reference,” on page 359.

Transactions that use WRITE JOURNALNUM command

The WRITE JOURNALNUM command is supported in CICS Transaction Server for z/OS, Version 3 Release 2 for compatability with earlier releases: the WRITE JOURNALNAME command is preferred for new applications. If resource security applies to a transaction executing WRITE JOURNALNUM, the journal number is prefixed with 'DFHJ' before the security check is applied. Thus, writing to journal number 2 requires UPDATE access to the resource DFHJ02.

Security for 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:
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 commands to define a transaction in the ACICSPCT class, and to 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 if you define profiles with 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 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), 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 72 for more information.

Transactions started without terminals

The 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 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 START command. If the USERID option is specified on the START command, the specified userid is used instead.

When an 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 "Intercommunication link security" on page 154. For information about EDF authorization of surrogate users, see "Conditional access processing" on page 70.

Access to resources by a non-terminal started transaction

If the USERID option is not specified on the **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, or resource access determined by a userid for EDF when used in dual-screen mode. This means:

- If a transaction-routed transaction executes a START command, or if a 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 a 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, omit the USERID option. Without the USERID option, resource access by the non-terminal started transaction is determined by the sign-on 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 sign-on parameters, userid, and groupid. It can also include a port of entry. The userid, groupid, and port of entry can be those inherited from the terminal transaction that started the non-terminal transaction.

If the USERID option is specified on a 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 70.

If a START TRANSID USERID command is executed by a terminal transaction specifying 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 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 a START TRANSID USERID command is executed by a terminal transaction specifying 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 effective for the terminal transaction that executed the START USERID command.

Access authorization levels

CICS requires a minimum authorization of READ for started transactions.

For information about the access authorization levels for system programming commands which apply to transactions, see Appendix B, “Resource and command check cross reference,” on page 359.

Security for application programs

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 LINK, LOAD, and XCTL commands. Also, the load status of programs can be altered by the CICS RELEASE, ENABLE, 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:

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 to 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 if you define profiles with 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 a **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.

For information about the access authorization levels for system programming commands which apply to programs, see Appendix B, "Resource and command check cross reference," on page 359.

Security for temporary storage

Unlike the other resources for which you specify RESSEC(YES), temporary storage queues, for which you require RACF protection, also require the security attribute in a suitable TSMODEL resource definition. You specify TSMODEL definitions in the CSD. See the *CICS Resource Definition Guide* for information about TSMODEL resource definitions.

Implementing security for temporary storage queues

To implement security for temporary storage queues:

1. Specify RESSEC(YES) in the CSD resource definition of the appropriate transactions.

2. Specify the security attribute on suitable TSMODEL resource definitions in the CSD. CICS does not perform any security checks on temporary storage queues that specify SECURITY=NO on the matching TSMODEL definition.
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. For example, use the following commands to define queues in the SCICSTST class, and to authorize users to both read from 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 if you define profiles with 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).

Note: CICS continues to support the DFHTST TYPE=SECURITY macro for defining temporary storage security. However, you are recommended to migrate your temporary storage tables (TSTs) to the CSD as TSMODEL definitions.

Other temporary storage security considerations

You can define the queue names on the PREFIX attribute of the TSMODEL resource definition 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 16 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 15 characters, because if you specify the full 16 characters 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 temporary storage queue 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, for example, a minimum of READ for READQ TS, and a minimum of UPDATE for DELETEQ TS and WRITEQ TS.

For information about the access authorization levels for system programming commands which apply to temporary storage queues, see Appendix B, “Resource and command check cross reference,” on page 359.

Security for CICS document templates

A document template is a complete document or a portion of a document which is created offline, or by a CICS program. Document templates are often used to supply Web pages through CICS Web support, either as static pages provided by a URIMAP definition, or as part of a dynamic Web page created by an application program. Document templates are defined by DOCTEMPLATE resource definitions.

Access to CICS document templates is controlled in the following cases:

- Document templates delivered as a static response to a Web client's request (specified on the TEMPLATENAME attribute of the URIMAP definition for the request).
- Document templates delivered as part of an application-generated response to a Web client's request (used by an application program that handles the request).
- All EXEC CICS CREATE, INQUIRE, DISCARD and SET DOCTEMPLATE commands.
- All EXEC CICS DOCUMENT INSERT and CREATE commands with the TEMPLATE option.

When calling the external security manager, CICS uses the name of the DOCTEMPLATE resource definition for the CICS document template, prefixed by its resource type, DOCTEMPLATE. For example, for a document template whose resource definition is named “WELCOME", the profile name passed to the security manager is DOCTEMPLATE.WELCOME. If you have used the system initialization parameter SECPRFX to add an additional prefix specific to the CICS region, this prefix is also used. For example, if the document template is in a development region where SECPRFX is set to TEST, then the profile name TEST.DOCTEMPLATE.WELCOME is passed to the security manager. You need to set up the profile names using this format. Note that the security checking process is case sensitive; for example, if the security manager is passed the profile name TEST.DOCTEMPLATE.WELCOME, it would be checked against the profile TEST.DOCTEMPLATE.WELCOME, but not against the profile TEST.DOCTEMPLATE.Welcome.

The EXEC CICS DOCUMENT commands reference document templates using the 48-character name of the template (as specified in the TEMPLATENAME attribute of the DOCTEMPLATE resource definition). However, security checking for these commands uses the name of the DOCTEMPLATE resource definition that corresponds to the TEMPLATENAME attribute. This means that you only need to set up one profile name for each document template, using the name of the DOCTEMPLATE resource definition, and not the TEMPLATENAME attribute.

Note: Document templates can be retrieved from a variety of sources, including partitioned data sets, CICS programs, CICS files, z/OS UNIX System

Services files, temporary storage queues, transient data queues, and exit programs. When resource security checking is carried out for a document template, CICS does **not** perform any additional security checking on the resource that supplies the document template, even if resource security is specified for that type of resource in the CICS region.

To implement resource security for CICS document templates, follow these steps:

1. Define profiles to RACF in the default RCICSRES resource class or WCICSRES grouping class, or their equivalents if you have user-defined resource class names. For the profile names, use the name of the DOCTEMPLATE resource definition, prefixed by the resource type DOCTEMPLATE, and any additional prefix specified by the SECPRFX system initialization parameter for the CICS region. For example, use the following commands to define document templates in the RCICSRES class, and authorize users to use them to assemble documents:
   ```
   RDEFINE  RCICSRES  (DOCTEMPLATE.doc1, DOCTEMPLATE.doc2, .., DOCTEMPLATE.docn) UACC(NONE)
                     NOTIFY(sys_admin_userid)
   PERMIT DOCTEMPLATE.doc1 CLASS(RCICSRES) ID(group1, group2) ACCESS(READ)
   ```
   To define document templates as members of a profile in the WCICSRES resource grouping class, with an appropriate access list, use the following commands:
   ```
   RDEFINE  WCICSRES  (doc_groupname) UACC(NONE)
                      ADDMEM(DOCTEMPLATE.doca, DOCTEMPLATE.docb) NOTIFY(sys_admin_userid)
   PERMIT doc_groupname CLASS(WCICSRES) ID(group_userid) ACCESS(READ)
   ```
   After you have issued the RDEFINE command for the RCICSRES or WCICSRES class, if the class is not yet active, you need to activate it by issuing a SETROPTS command. For example:
   ```
   SETROPTS CLASSACT(RCICSRES) RACLIST(RCICSRES)
   ```
   If the class is active, you need to refresh it using the SETROPTS command:
   ```
   SETROPTS RACLIST(RCICSRES) REFRESH
   ```
2. Specify SEC=YES as a CICS system initialization parameter (and SECPRFX if you define profiles with a prefix).
3. Specify XRES=YES for the default resource class names of RCICSRES and WCICSRES, or XRES=*name* for user-defined resource class names. XRES=YES is the default.
4. Specify RESSEC(YES) in the TRANSACTION resource definition of the transactions that access the CICS document templates. For CICS Web support, the transaction for all static responses is CWXN, or an alternate transaction that you have specified in place of CWXN using the TRANSACTION attribute on your TCPIPSERVICE definitions. The transaction for application-generated responses is an alias transaction, which can be specified in the URIMAP definition for the request or set by an analyzer program, and defaults to CWBA. As supplied by CICS, the definition for CWXN specifies RESSEC(YES), but the definition for CWBA specifies RESSEC(NO), and for TRANSACTION resource definitions in general the default is RESSEC(NO).

Access authorization levels

A minimum of READ access is required for CICS document templates used as static Web pages, document templates used by application programs as part of application-generated responses, and document templates used with all EXEC CICS DOCUMENT INSERT and CREATE commands with the TEMPLATE option.

For information about the access authorization levels for system programming commands which apply to CICS document templates, see Appendix B, “Resource and command check cross reference,” on page 359.

User IDs for access to document templates and z/OS UNIX files used by CICS Web support

When resource security is active for a transaction, the external security manager checks whether the user ID associated with the transaction is authorized to access the required resources. For CICS Web support, the user ID associated with the transaction for a particular Web request can be obtained from different sources. Depending on the level of security that you require, you can arrange your CICS Web support architecture to determine the user IDs that are used for resource security checking against the secured document templates or z/OS UNIX files.

Application-generated responses

For CICS Web support, the transaction for application-generated responses is an alias transaction, which can be specified in the URIMAP definition for the request or set by an analyzer program, and defaults to CWBA. CWBA is defined as RESSEC(NO), so if you require resource security for the alias transaction, you must either copy the CWBA definition to your own group and change its RESSEC attribute, or use a different alias transaction.

When a Web client makes a request to CICS Web support, and the response is provided by an application, CICS selects a userid for the alias transaction in the following order of priority:

1. A user ID that you set using an analyzer program. This user ID can override a user ID obtained from the Web client or supplied by a URIMAP definition.
2. A user ID that you obtained from the Web client using basic authentication, or a user ID associated with a client certificate sent by the Web client.
3. A user ID that you specified in the URIMAP definition for the request.
4. The CICS default user ID, if no other can be determined.

For application-generated responses, the user ID selected for the Web client applies to the whole alias transaction, and must be authorized to attach the transaction and use the Web application program, as well as to use secured document templates.

Web clients' user IDs do not need specific permissions on z/OS UNIX files for application-generated responses, because applications can only manipulate z/OS UNIX files using EXEC CICS commands when the files are defined as CICS document templates. Security checking is only carried out for the CICS document template, and not again for the z/OS UNIX file.

Static responses

The transaction for all static responses specified in URIMAP definitions is the default Web attach task CWXN, or any alias transaction that you have specified in place of CWXN using the TRANSACTION attribute on your TCPIPSERVICE definitions.

When a Web client makes a request to CICS Web support, and the response is a static response specified in a URIMAP definition, the user ID used for the Web client is a user ID that you obtained from the Web client using basic authentication, or a user ID associated with a client certificate sent by the Web client.

Resource security checking for document templates is controlled by the XRES system initialization parameter and the RESSEC attribute for the transaction (CWXN or its alias). Access control for z/OS UNIX files is controlled by the XHFS system initialization parameter only.

The user ID for the Web client is used only during the process of resource security checking for the document template or z/OS UNIX file that is to be delivered as the static response. The user ID must be authorized to access the document template or z/OS UNIX file.

Security for z/OS UNIX files

Files stored in the z/OS UNIX System Services file system can be used to supply Web pages through CICS Web support, as static responses provided by URIMAP definitions. When access control for these files is specified, you can control access to them on the basis of the user IDs for individual Web clients. Access control for z/OS UNIX files is enabled by default.

Access control for z/OS UNIX files is activated by the XHFS system initialization parameter. The default for this parameter is YES, meaning that resource security for z/OS UNIX files is active. If you do **not** want resource security for these files, set this system initialization parameter to NO.

Access control for z/OS UNIX files is based on a user ID that is obtained from the Web client using basic authentication, or a user ID associated with a client certificate sent by the Web client. The user ID is used only during the process of security checking.

Access control for z/OS UNIX files differs from standard resource security for the other resource types controlled by X*name* system initialization parameters, in some important ways:

- Access controls for z/OS UNIX files are not managed directly by RACF. They are specified in z/OS UNIX System Services, which makes use of RACF to manage user IDs and groups of user IDs, but keeps control of the permissions set for the files and directories. Because of this, you do not need to define RACF profiles for individual files, and you cannot use the QUERY SECURITY command to check access to them. You check and specify permissions for z/OS UNIX files and directories in the z/OS UNIX System Services shell environment, using z/OS UNIX commands. RACF is used to manage user profiles, groups and access control lists (ACLs). If you are using ACLs, you need to activate the FSSEC class for these to be checked.
- Security checking for z/OS UNIX files is not affected by the RESSEC attribute in the TRANSACTION resource definition of the transactions that access the files. If XHFS=YES is specified as a system initialization parameter for the CICS region, all z/OS UNIX files used by CICS Web support as static responses (and their directories) are subject to security checking, regardless of the RESSEC attribute for the transaction that is accessing them. (However, the SEC system initialization parameter does affect whether or not security checking is carried out, as for all resources.)
- z/OS UNIX files are not referenced directly by any CICS application programming commands or system programming commands. They can only be referenced by EXEC CICS commands when they are defined as CICS document templates. In

this situation, resource security for CICS document templates (specified by the XRES system initialization parameter) controls access to them for users. CICS does **not** perform any additional permissions check on the z/OS UNIX files using the Web client's user ID. This is the case even if access control is specified for z/OS UNIX files in the CICS region, or if resource security is not active for document templates. Where z/OS UNIX files are defined as CICS document templates, you therefore need to set up Web clients' user ID access controls in RACF for the CICS document templates, rather than in z/OS UNIX System Services for the z/OS UNIX files. (However, the CICS region user ID always needs to have **read** permissions on z/OS UNIX files, even if they are defined as document templates.) Note in particular that this situation applies to all application-generated responses from CICS Web support, and to any URIMAP definitions for static responses where the TEMPLATENAME attribute is used, rather than the HFSFILE attribute.

Implementing security for z/OS UNIX files

To implement access control for z/OS UNIX files used by CICS Web support, when they are specified as static responses in URIMAP definitions using the HFSFILE attribute, follow the steps listed in this topic.

Note that the CICS region user ID must always have a minimum of **read** access to all z/OS UNIX files that it uses for CICS Web support, and to the directories containing them. The user ID of the Web client is only used when accessing z/OS UNIX files as a static response, but the CICS region user ID applies to all other attempts to access the file. If the CICS region user ID does not have permission to access the file, even an authorized Web client will be unable to view it. This is the case even when the file is defined as a CICS document template.

1. Select an appropriate method to give permissions to Web clients to access the z/OS UNIX files and directories. You might choose to use the group permissions for the files and directories, or access control lists (ACLs). Even if it is possible for you to use group permissions, the use of ACLs is the recommended solution for giving permissions to Web clients' user IDs. With ACLs, you can allow access to multiple user groups, and the access can be set for single files or once for all the files in a directory. Directory permissions can be arranged in the same way. You can also use ACL commands to modify permissions for the files and directories. Although you work with ACLs in the z/OS UNIX System Services shell environment, they are actually created and checked by RACF, so if you are using a different security product, check its documentation to see if ACLs are supported. When you have chosen your preferred method, follow the relevant steps in the remainder of this procedure.
2. Identify the authenticated user IDs used by Web clients. These must be the basis of your access control. (You cannot supply an override using an analyzer program, as you can with application-generated responses.) Authenticated user IDs will already have a user profile defined in your security manager.
3. For each Web client user ID, choose and assign a suitable z/OS UNIX user identifier (UID). The UIDs are numbers that can be in the range 0 to 16 777 216. (0 is a superuser ID with system privileges, which should not be assigned to CICS users.) To assign UIDs, specify the UID value in the OMVS segment of the user profile for each user ID. “RACF user profiles” on page 16 tells you how to update a RACF user profile using the ALTUSER command. For example, if the Web client's user ID is WEBUSR1, and the UID you want to assign is 2006, use the command:

   ```
   ALTUSER WEBUSR1 OMVS(UID(2006))
   ```

All users must have a z/OS UNIX user identifier (UID) in their user profile in order to use z/OS UNIX function, even if you are not assigning permissions on the basis of the UID. *z/OS UNIX System Services Planning*, GA22-7800, explains how to manage the UIDs and GIDs for your z/OS UNIX system.

4. Choose, or create, RACF groups that can be used by groups of Web clients with the same permissions. For best performance, even if you are using ACLs, you should assign permissions to groups rather than individual users.
5. For each RACF group, choose a suitable z/OS UNIX group identifier (GID), and assign the GID to the RACF group. To assign a GID, specify the GID value in the OMVS segment of the RACF group profile. For example, if the RACF group is CICSWEB1, and the GID you want to assign is 9, use the command:
   ```
   ALTGROUP CICSWEB1 OMVS(GID(9))
   ```
6. Make sure that each of your Web client user IDs connects to a RACF group to which you assigned a z/OS UNIX group identifier (GID). If your Web clients need to connect to more than one RACF group, RACF list of groups must be active in your system.
7. Before you modify the permissions for the z/OS UNIX files and directories, ensure that your user ID is either a superuser on z/OS UNIX, or the owner of each z/OS UNIX file and directory you want to work with. Also, if you are working with groups, the owner of the files and directories must be connected to the RACF groups that you are using.
8. Optional: If you have chosen to use ACLs, set up ACLs that apply to all of the z/OS UNIX files and directories used by CICS Web support for static responses, using the `setfacl` command in the z/OS UNIX System Services shell environment. *z/OS UNIX System Services Planning*, GA22–7800, has information about using ACLs, and examples of how to use the `setfacl` command.
   a. For files, you can set up access ACLs, which apply to an individual file, or file default ACLs, which apply to all the files within a directory and within its subdirectories.
   b. For directories, you can set up access ACLs, which apply to an individual directory, or directory default ACLs, which apply to the subdirectories within a directory.
   c. To minimize the impact to performance, assign group permissions for the files to the RACF groups to which your Web clients' user IDs connect, rather than using individual user IDs. (There is also a limit on the number of items that can be specified in an ACL.)
   d. If you need to change the permissions granted to the groups (the base permission bits which specify **read**, **write** and **execute** access), you can specify this using the `setfacl` command as well. Web clients need to have **read** access to the z/OS UNIX files and directories.
   e. If you are using ACLs, ensure that the FSSEC class is activated. Use the RACF command `SETROPTS CLASSACT(FSSEC)` to do this. You can define ACLs prior to activating the FSSEC class, but you must activate the FSSEC class before ACLs can be used in access decisions.
9. Optional: If you have chosen to use group permissions without using ACLs, assign the group permissions for each z/OS UNIX file and directory to a group to which your Web clients connect, and give the group **read** permissions. Use the UNIX command `chmod` to do this. *z/OS UNIX System Services Command Reference*, SA22-7802, and *z/OS UNIX System Services User's Guide*, SA22-7801, have information about using this command. (Note that as group permissions can only be assigned to one group if you are using this method,

some of your Web clients' user IDs might need to connect to more than one group to acquire all the correct permissions.)

10. Specify SEC=YES as a CICS system initialization parameter. (SECPRFX is not relevant for z/OS UNIX files, as they do not have RACF profiles).
11. Specify XHFS=YES as a CICS system initialization parameter. This step activates access control for all z/OS UNIX files in the CICS region.

When you have completed the setup procedure, from this point onwards:

- All Web clients who use a connection with basic authentication or client certificate authentication and attempt to access any HFS files, must have a user profile in the security manager which contains a valid z/OS UNIX UID, and connects to a RACF group with a valid z/OS UNIX GID.
- To be able to view a Web page derived from a z/OS UNIX file, Web clients who use a connection with basic authentication or client certificate authentication must have **read** permissions to the file and to the directories containing it, either individually or through the RACF groups to which they are connected.

If these conditions are not in place, Web clients will receive a 403 (Forbidden) status code, and CICS issues message DFHXS1116.

Access authorization levels

The user IDs of Web clients, and also the CICS region user ID, must have a minimum of **read** access for z/OS UNIX files, and the directory containing them, which are used with CICS Web support as static responses provided by URIMAP definitions (specified by the HFSFILE attribute).

If z/OS UNIX files are defined as CICS document templates, and the document templates are used either in URIMAP definitions as static responses (specified by the TEMPLATENAME attribute), or by applications, CICS does **not** perform any additional permissions check on the files using the Web client's user ID. However, the CICS region user ID still needs to have read permissions on the files, even when they are defined as document templates.

z/OS UNIX files are not manipulated directly by any CICS application programming commands or system programming commands. They can only be manipulated by EXEC CICS commands when they are defined as CICS document templates. In this situation, resource security for CICS document templates controls access to them for users.

Security for 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:

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 to 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)
```

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 if you define profiles with 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 both types of DL/I interface (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, see *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, CICS determines the security processing for the target transaction from the logical OR of RESSEC in the resource definitions for the target transaction and the CEDF transaction.

Table 9 shows the security checking performed for the transaction XSUB for different settings of RESSEC.

Table 9. Security checking of transactions running under CEDF

CEDF	XSUB	Security checking
RESSEC(YES)	RESSEC(YES)	Any access to CICS resources causes a security check.
RESSEC(YES)	RESSEC(NO)	Any access to CICS resources causes a security check. (Logical OR results in RESSEC on.)
RESSEC(NO)	RESSEC(YES)	Any access to CICS resources causes a security check. (Logical OR results in RESSEC on.)
RESSEC(NO)	RESSEC(NO)	Access to CICS resources does 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. The

IBM-supplied definition of CEDF in the DFHEDF group specifies RESSEC(YES). Definitions in the IBM-supplied groups cannot be modified, so to change the definition, copy it to another group.

When the 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 to test a transaction, 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 those programs 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 only 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). This could have occurred if the C* profile was created before generic profiles had been enabled with a SETROPTS command. 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 can find neither a specific nor generic profile, it returns a “no profile found” condition. 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. If RACF cannot find the APPL class, it returns a “READ access intent” condition.

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 *z/OS Security Server RACF System Programmer's Guide*.

Chapter 7. Surrogate user security

A surrogate user is a RACF-defined user who is authorized to act on behalf of another user (the original user). CICS uses surrogate user security in a number of different situations.

Where surrogate user checking applies

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.

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. If surrogate user checking is in force, it applies to:

- The CICS default user
- PLT post-initialization processing
- Preset terminal security
- Started transactions
- The userid associated with a CICS business transaction services (BTS) process or activity that is started by a RUN command
- The userid associated with a transient data destination
- The userid supplied as a parameter on an EXCI call
- The userid supplied on the AUTHID and COMAUTHID attributes of the DB2CONN and DB2ENTRY resource definitions
- The userid supplied on the USERID attribute of URIMAP resource definitions
- A CPSM MAS agent started with the COLM transaction
- A CPSM local MAS agent started with the CORM transaction

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 the 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 “Preset terminal security” on page 72.

Started transactions

CICS performs surrogate user checks when you use a 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 to be 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 a 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 “Intercommunication link security” on page 154.

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 because of link security or because of EDF user security.

BTS processes and activities

When a CICS business transaction services (BTS) process or activity is activated by a **RUN** command, it may run under a different userid from that of the transaction that issues the RUN. For more information about BTS, see *CICS Business Transaction Services*.

The application programmer can specify under whose authority a process or activity is to run, when it is activated by a RUN command, by coding the USERID option of the **DEFINE PROCESS** or **DEFINE ACTIVITY** command. If the USERID option is omitted, the value defaults to the userid of the transaction that issues the DEFINE command.

If the USERID option is specified, CICS performs (at define time) a surrogate security check to verify that the userid of the transaction that issued the DEFINE command is authorized to use the userid specified by USERID.

Transient data trigger-level transactions

When a transient data queue is defined by RDO with a non-terminal trigger-level transaction and a USERID parameter, the user installing the definition is checked. Likewise, when such a transient data queue is created with the CREATE TDQUEUE command, the user executing the command is checked.

The userid for a transient data trigger-level transaction that is not associated with a terminal can be specified on the TDQUEUE resource definition or on the SET TDQUEUE command.

Intrapartition transient data resources

CICS uses the userid specified on transient data queue definition for security checking in any trigger-level transactions that are not associated with a terminal. Code the USERID operand 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 facility is a 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 default userid specified on the DFLTUSER system initialization parameter. Ensure that the userid of any CICS region in which the transient data queue definition is installed is defined as a surrogate of all the userids specified in the transient data destination definition. This is because, during a cold start, CICS performs a surrogate user security check for the CICS region userid against all the userids specified in transient data queue definitions that are being installed. If the surrogate security check fails, the transient data queue definition is not installed.

EXEC CICS SET TDQUEUE ATIUSERID

The system programming command, SET TDQUEUE with the ATIUSERID option, specifies the userid for a transient data trigger-level transaction that is not associated with a terminal. The destination facility must be a file.

CICS performs a surrogate user security check against the userid of the transaction that issues the SET TDQUEUE command, to verify that the transaction userid is authorized as a surrogate user of the userid specified on the ATIUSERID parameter.

Surrogate user checking for EXCI calls

A surrogate user check is performed to verify that the batch region's userid is authorized to issue DPL calls for another user (that is, is authorized as a surrogate of the userid specified on the DPL_request call).

If you want your external CICS interface (EXCI) client jobs to be subject to surrogate user checking, specify SURROGCHK=YES in the EXCI options table, DFHXCOPT. If you specify SURROGCHK=YES, authorize the batch region's userid as a surrogate of the userid specified on all DPL_request calls. This means the batch region's userid must have READ access to a profile named *userid*.DFHEXCI in the SURROGAT general resource class (where *userid* is the userid specified on the DPL call). For example, the following commands define a surrogate profile for a DPL userid, and grant READ access to the EXCI batch region:

```
RDEFINE  SURROGAT dpl_userid.DFHEXCI UACC(NONE) OWNER(DPL_userid)
PERMIT   userid.DFHEXCI CLASS(SURROGAT) ID(batch_region_userid)
                ACCESS(READ)
```

If surrogate user checking is enabled (SURROGCHK=YES), but no userid is specified on the DPL call, no surrogate user check is performed, because the userid on the DPL call defaults to the batch region's userid.

If you do not want surrogate user security checking, specify SURROGCHK=NO in the DFHXCOPT options table.

Surrogate user checking is useful when the batch region's userid is the same as the CICS server region userid, in which case the link security check is bypassed. In this case, a surrogate user check is recommended, because the USERID specified on the DPL call is not an authenticated userid (no password is passed).

If the batch region's userid and the CICS region userid are different, link security checking is enforced. With link security, an unauthenticated userid passed on a DPL call cannot acquire more authority than that allowed by the link security check. It can acquire only the same, or less, authority than allowed by the link security check.

The userid on DB2 AUTHID and COMAUTHID parameters

When you install a DB2CONN resource definition that specifies the AUTHID, SIGNID, or COMAUTHID attribute, or when you install a DB2ENTRY definition that specifies AUTHID, or when you modify one of these attributes, CICS checks that the userid performing the operation is authorized as a surrogate user of AUTHID, COMAUTHID, or SIGNID. This also applies to the CICS region userid during group list install on a CICS cold or initial start.

For more information about these attributes, see the *CICS Resource Definition Guide*.

Note: The XUSER system initialization parameter is also used to control access to the AUTHTYPE and COMAUTHTYPE attributes, but the security control for these parameters is managed through the facility general resource class. See the *CICS DB2 Guide* for more information.

The userid on URIMAP resource definitions

When you install a URIMAP resource definition that specifies the USERID attribute, or when you modify this attribute, CICS checks that the userid performing the operation is authorized as a surrogate user of the user ID specified for the USERID attribute. This also applies to the CICS region userid during group list install on a CICS cold or initial start.

URIMAP resource definitions are used for CICS Web support. For more information about these resource definitions, see the *Resource Definition Guide*.

RACF definitions for surrogate user checking

To enable CICS surrogate user checking:

- 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**

userid represents one of the following:
- The userid under which a started transaction is to run
- The userid associated with a CICS business transaction services (BTS) process or activity that is started by a RUN command

userid.**DFHINSTL**

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
- The userid specified on the AUTHID or COMAUTHID parameter of a DB2 resource definition.

There is also a form of surrogate class profile that you can define for external CICS interface (EXCI) security checking:

userid.**DFHEXCI**

userid represents the user specified on the DPL call in the client batch region.

To authorize a surrogate to this EXCI profile, grant the EXCI batch region's userid READ access.

Note that surrogate security checks in an EXCI batch region are independent of security definitions in the target CICS region. If SURROGCHK is specified in the EXCI options table (DFHXCOPT), surrogate security checks are performed in the EXCI client program's address space regardless of the CICS security settings.

To authorize a surrogate user to one of these profiles, you must grant READ access.

You do not need to define a user as that user's own surrogate. CICS bypasses the surrogate check in this case.

The *z/OS Security Server RACF Security Administrator's Guide* gives more information about defining surrogate resource classes. 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 requiring 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, 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, 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 112 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 135.

Chapter 8. CICS command security

CICS command security controls the use of system programming commands; that is, commands that require the special CICS translator option, SP.

Introduction to command security

CICS command security applies to system programming 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. Table 10 shows the commands that are subject to command security checking:

Table 10. Access required for system programming commands

Command name	Access required
COLLECT EXTRACT STATISTICS INQUIRE	READ
DISABLE ENABLE EXTRACT but not EXTRACT STATISTICS PERFORM RESYNC SET	UPDATE
CREATE DISCARD	ALTER
Note: Because the PERFORM CORBASERVER SCAN may result in the dynamic creation and installation of DJAR resources, the PERFORM CORBASERVER SCAN command requires ALTER access to the DJAR command security resource as well as UPDATE authority to the CORBASERVER resource.	

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.

Front End Programming Interface security uses the same mechanism for authorization as the system programming commands, using the FEPIRESOURCE resource name. Front End Programming Interface security is not discussed in this book. See the *CICS Front End Programming Interface User's Guide* for details.

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 *z/OS Security Server RACF Security Administrator's Guide*. DFHEITBS is the language definition table that defines the system programming 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 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 11. 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 11. CICS resources subject to command security checking

Resource name (see note 1)	Related CICS command(s)
ASSOCIATION	INQUIRE ASSOCIATION
AUTINSTMODEL	INQUIRE AUTINSTMODEL DISCARD AUTINSTMODEL
AUTOINSTALL	INQUIRE AUTOINSTALL SET AUTOINSTALL
BEAN	INQUIRE BEAN
BRFACILITY	INQUIRE BRFACILITY SET BRFACILITY
CFDTPOOL	INQUIRE CFDTPOOL
CLASSCACHE	INQUIRE CLASSCACHE PERFORM CLASSCACHE SET CLASSCACHE

CONNECTION	INQUIRE CONNECTION
	SET CONNECTION
	CREATE CONNECTION
	DISCARD CONNECTION

CORBASERVER	INQUIRE CORBASERVER SET CORBASERVER CREATE CORBASERVER DISCARD CORBASERVER PERFORM CORBASERVER

DB2CONN	INQUIRE DB2CONN
	SET DB2CONN
	CREATE DB2CONN
	DISCARD DB2CONN

DB2ENTRY	INQUIRE DB2ENTRY
	SET DB2ENTRY
	CREATE DB2ENTRY
	DISCARD DB2ENTRY

DB2TRAN	INQUIRE DB2TRAN
	SET DB2TRAN
	CREATE DB2TRAN
	DISCARD DB2TRAN

DELETSHIPPED	INQUIRE DELETSHIPPED SET DELETSHIPPED PERFORM DELETSHIPPED

Table 11. CICS resources subject to command security checking (continued)

Resource name (see note 1)	Related CICS command(s)
DISPATCHER	INQUIRE DISPATCHER SET DISPATCHER
DJAR	INQUIRE DJAR CREATE DJAR DISCARD DJAR PERFORM DJAR **Note:** ALTER access to the associated DJAR resource is required for the PERFORM CORBASERVER SCAN command.

DOCTEMPLATE	INQUIRE DOCTEMPLATE
	CREATE DOCTEMPLATE
	DISCARD DOCTEMPLATE
	SET DOCTEMPLATE

DSNAME	INQUIRE DSNAME SET DSNAME
DUMP	PERFORM DUMP **CEMT** PERFORM SNAP
DUMPDS	INQUIRE DUMPDS SET DUMPDS

ENQMODEL	INQUIRE ENQMODEL
	SET ENQMODEL
	CREATE ENQMODEL

EXCI	INQUIRE EXCI
EXITPROGRAM	**EXEC CICS** ENABLE PROGRAM **EXEC CICS** DISABLE PROGRAM **EXEC CICS** EXTRACT EXIT **EXEC CICS** RESYNC ENTRYNAME INQUIRE EXITPROGRAM
FEPIRESOURCE	Certain EXEC CICS FEPI commands (see note 3)

FILE	INQUIRE FILE
	SET FILE
	CREATE FILE
	DISCARD FILE

HOST	INQUIRE HOST SET HOST

IPCONN	INQUIRE IPCONN
	SET IPCONN
	CREATE IPCONN
	DISCARD IPCONN

IRC	INQUIRE IRC SET IRC
JOURNALMODEL	**EXEC CICS** INQUIRE JOURNALMODEL **EXEC CICS** CREATE JOURNALMODEL **EXEC CICS** DISCARD JOURNALMODEL **CEMT** INQUIRE JMODEL
JOURNALNAME	INQUIRE JOURNALNAME SET JOURNALNAME
JVM	INQUIRE JVM
JVMPOOL	INQUIRE JVMPOOL SET JVMPOOL

Table 11. CICS resources subject to command security checking (continued)

Resource name (see note 1)	Related CICS command(s)
JVMPROFILE	INQUIRE JVMPROFILE

LIBRARY	DISCARD LIBRARY
	INQUIRE LIBRARY
	SET LIBRARY
	CREATE LIBRARY

LINE	**CEMT** INQUIRE LINE **CEMT** SET LINE
LSRPOOL	CREATE LSRPOOL
MAPSET	CREATE MAPSET
MODENAME	INQUIRE MODENAME SET MODENAME
MONITOR	INQUIRE MONITOR SET MONITOR
MVSTCB	COLLECT STATISTICS INQUIRE MVSTCB
PARTITIONSET	CREATE PARTITIONSET
PARTNER	INQUIRE PARTNER CREATE PARTNER DISCARD PARTNER
PIPELINE	CREATE PIPELINE DISCARD PIPELINE INQUIRE PIPELINE PERFORM PIPELINE SET PIPELINE
PROCESSTYPE	**EXEC CICS** CREATE PROCESSTYPE **EXEC CICS** DISCARD PROCESSTYPE **CEMT** INQUIRE PROCESSTYPE **CEMT** SET PROCESSTYPE
PROFILE	INQUIRE PROFILE CREATE PROFILE DISCARD PROFILE

PROGRAM	INQUIRE PROGRAM
	SET PROGRAM
	CREATE PROGRAM
	DISCARD PROGRAM

REQID	**EXEC CICS** INQUIRE REQID
RESETTIME	PERFORM RESETTIME (see note 4)
REQUESTMODEL	INQUIRE REQUESTMODEL
RRMS	INQUIRE RRMS
SECURITY	PERFORM SECURITY REBUILD
SESSIONS	CREATE SESSIONS
SHUTDOWN	PERFORM SHUTDOWN (see note 2)
STATISTICS	INQUIRE STATISTICS SET STATISTICS **EXEC CICS** COLLECT STATISTICS **EXEC CICS** EXTRACT STATISTICS **EXEC CICS** PERFORM STATISTICS RECORD

Table 11. CICS resources subject to command security checking (continued)

Resource name (see note 1)	**Related CICS command(s)**
STORAGE	INQUIRE STORAGE
STREAMNAME	INQUIRE STREAMNAME
SUBPOOL	INQUIRE SUBPOOL
SYSDUMPCODE	INQUIRE SYSDUMPCODE (see note 4) SET SYSDUMPCODE (see note 4)
SYSTEM	INQUIRE SYSTEM SET SYSTEM
TASK	INQUIRE TASK INQUIRE TASK LIST SET TASK LIST

TCLASS	INQUIRE TCLASS
	SET TCLASS
	INQUIRE TRANCLASS
	SET TRANCLASS
	CREATE TRANCLASS
	DISCARD TRANCLASS

TCPIP	INQUIRE TCPIP SET TCPIP

TCPIPSERVICE	INQUIRE TCPIPSERVICE
	SET TCPIPSERVICE
	CREATE TCPIPSERVICE
	DISCARD TCPIPSERVICE

TDQUEUE	INQUIRE TDQUEUE
	SET TDQUEUE
	CREATE TDQUEUE
	DISCARD TDQUEUE

TERMINAL	INQUIRE TERMINAL
	SET TERMINAL
	CREATE TERMINAL
	DISCARD TERMINAL
	INQUIRE NETNAME
	SET NETNAME

TRACEDEST	**EXEC CICS** INQUIRE TRACEDEST **EXEC CICS** SET TRACEDEST
TRACEFLAG	**EXEC CICS** INQUIRE TRACEFLAG **EXEC CICS** SET TRACEFLAG
TRACETYPE	**EXEC CICS** INQUIRE TRACETYPE **EXEC CICS** SET TRACETYPE
TRANDUMPCODE	INQUIRE TRANDUMPCODE (see note 4) SET TRANDUMPCODE (see note 4)

TRANSACTION	INQUIRE TRANSACTION
	SET TRANSACTION
	CREATE TRANSACTION
	DISCARD TRANSACTION

TSMODEL	INQUIRE TSMODEL CREATE TSMODEL DISCARD TSMODEL
TSPOOL	INQUIRE TSPOOL

Table 11. CICS resources subject to command security checking (continued)

Resource name (see note 1)	**Related CICS command(s)**
TSQUEUE	**EXEC CICS** INQUIRE TSQUEUE
TSQNAME	INQUIRE TSQNAME SET TSQNAME
TYPETERM	CREATE TYPETERM
UOW	INQUIRE UOW SET UOW
UOWDSNFAIL	INQUIRE UOWDSNFAIL
UOWENQ	INQUIRE UOWENQ
UOWLINK	INQUIRE UOWLINK **EXEC CICS** SET UOWLINK

URIMAP	INQUIRE URIMAP
	SET URIMAP
	CREATE URIMAP
	DISCARD URIMAP

VTAM	INQUIRE VTAM SET VTAM
WEB	INQUIRE WEB SET WEB

WEBSERVICE	CREATE WEBSERVICE
	DISCARD WEBSERVICE
	INQUIRE WEBSERVICE
	SET WEBSERVICE

WORKREQUEST	INQUIRE WORKREQUEST SET WORKREQUEST

Note:

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 Front End Programming Interface User's Guide*.
4. See “CEMT considerations” on page 127.

If you are running CICS with command security, define resource profiles to RACF, with access lists as appropriate, using the resource names in Table 11 on page 120 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, VTAM)
PERMIT CMDSAMP CLASS(VCICSCMD) ID(operator_group) ACCESS(READ)
```

The second 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, 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 11 on page 120.

Parameters for specifying command security

In addition to the SEC and SECPRFX system initialization parameters, which are described in “Security-related system initialization parameters” on page 56, CICS provides the XCMD system initialization parameter and the CMDSEC attribute on the TRANSACTION resource definition option to enable you to specify that you want command security:

XCMD system initialization parameter

Use the XCMD system initialization parameter 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. CICS then 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. The CMDSEC option is recommended for installations that need total control of the system programming commands.

The CMDSEC transaction definition attribute

You specify which transactions you want command security to apply to by using the CMDSEC attribute on the TRANSACTION resource definition, as follows:

CMDSEC(NO)
You do not want command security checking the transaction.

CMDSEC(YES)
You want command security checking on the system programming commands in Table 10 on page 119.

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 runs under the CEDF transaction, CICS uses the CMDSEC attribute in the definition of the target transaction *and* the CEDF transaction:

Table 12. Security checking for transactions running under CEDF

CEDF	target transaction	Security checking
CMDSEC(YES)	CMDSEC(YES)	Any access to CICS commands causes a security check.
CMDSEC(YES)	CMDSEC(NO)	Any access to CICS commands causes a security check.
CMDSEC(NO)	CMDSEC(YES)	Any access to CICS commands causes a security check.
CMDSEC(NO)	CMDSEC(NO)	Access to CICS commands does not cause a security check.

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. The IBM-supplied definition of CEDF in the DFHEDF group specifies CMDSEC(YES). Definitions in the IBM-supplied groups cannot be modified, so to change the definitions, copy them to another group.

When CEBR or CECI is invoked from within EDF it is 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 to test a transaction, 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 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 an EXEC CICS SIGNON, EXEC CICS VERIFY PASSWORD, or EXEC CICS CHANGE PASSWORD command is 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 system programming commands shown in Table 10 on page 119. 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 system programming commands in Table 10 on page 119 depend, 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 that protects 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) command. and you should provide READ access to the group profile that protects the commands on which you want them 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)
```

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 in application programs

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, 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 *CICS Application Programming Guide*; for programming information, see the *CICS Application Programming Reference* and *CICS System Programming Reference*.

Chapter 9. Security checking using the QUERY SECURITY command

Use the QUERY SECURITY command in an application program to determine the level of access that the transaction user has to a particular resource. The QUERY SECURITY command does not grant or deny access to a resource. Instead, the application program uses the values returned by the command to determine what action to take.

How QUERY SECURITY works

How QUERY SECURITY works depends on:

- Whether SEC=YES or SEC=NO is specified in the system initialization parameters
- The value specified for the SECPRFX system initialization parameter
- 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

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

SEC system initialization parameter

Table 13 assumes that the relevant resource class is active; for example, that XFCT=YES is specified when issuing QUERY SECURITY RESTYPE('FILE').

Table 13. The effect of the SEC parameter on QUERY SECURITY commands

SEC	RACF Access	Query Security Read	Query Security Update	Query Security Control	Query Security 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

The SECPRFX system initialization parameter determines whether CICS applies a prefix to the RESID option on the QUERY SECURITY command. For example, if you issue the following command:

```
QUERY SECURITY RESTYPE('FILE') RESID('PAYFILE')
```

- When SECPRFX=YES, CICS applies the CICS region userid as a prefix, and calls RACF to check the user's access to *cics_region_userid*.PAYFILE.
- When SECPRFX=*prefix*, CICS applies the prefix supplied, and calls RACF to check the user's access to *prefix*.PAYFILE.
- If SECPRFX=NO is specified, CICS does not apply a prefix, and calls RACF to check the user's access to PAYFILE.

Resource class system initialization parameters

Table 13 on page 129 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, if 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 user's access to the resource, if appropriate. For more information, see “Link security with LU6.2” on page 161, or “Link security with MRO” on page 214, depending upon 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.

The RESTYPE option

Use the QUERY SECURITY command with the RESTYPE option to query access levels to CICS resources (including DB2 resource definitions) contained in the classes activated at initialization by RACLIST. The response to the QUERY SECURITY command indicates the result of a resource check on this resource. If the resource is not defined to RACF, CICS does not grant access and the response is NOTREADABLE. Ensure the length of the resource name passed to RACF with a RESTYPE request is 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 14.

Table 14. QUERY SECURITY RESTYPE values

RESTYPE value	**X*name* parameter**
DB2ENTRY	XDB2
DOCTEMPLATE	XRES
FILE	XFCT
JOURNALNAME	XJCT
PROGRAM	XPPT
PSB	XPSB

Table 14. QUERY SECURITY RESTYPE values (continued)

RESTYPE value	X*name* parameter
SPCOMMAND	XCMD
TDQUEUE	XDCT
TRANSACTION	XPCT
TRANSATTACH	XTRAN
TSQUEUE	XTST
TSQNAME	XTST

The system initialization parameter XHFS, which controls resource security for z/OS UNIX files, does not have a corresponding RESTYPE value on the QUERY SECURITY command. This is because access controls for z/OS UNIX files follow the system of permissions used by z/OS UNIX System Services, so they operate in a different way.

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, 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. The list of possible RESID values for SPCOMMAND is as follows:

| • ASSOCIATION
- AUTINSTMODEL
- AUTOINSTALL
- BRFACILITY
- CFDTPOOL
- CONNECTION
- CORBASERVER
- DB2CONN
- DB2ENTRY
- DB2TRAN
- DISPATCHER
- DJAR
- DOCTEMPLATE
- DSNAME
- DUMP
- DUMPDS
- ENQUEUE
- EXCI
- EXITPROGRAM
- FEPIRESOURCE

- FILE
- HOST
- IPCONN
- IRC
- JOURNALMODEL
- JOURNALNAME
- JVMPOOL
- LIBRARY
- MODENAME
- MONITOR
- MVSTCB
- PARTNER
- PIPELINE
- PROCESS
- PROFILE
- PROGRAM
- REQID
- REQUEST
- RESETTIME
- RRMS
- SECURITY
- SHUTDOWN
- STATISTICS
- STORAGE
- SUBPOOL
- SYSDUMPCODE
- SYSTEM
- TASK
- TCLASS
- TCPIP
- TCPIPSERVICE
- TDQUEUE
- TERMINAL
- TRACEDEST
- TRACEFLAG
- TRACETYPE
- TRANDUMPCODE
- TRANSACTION
- TSQUEUE
- TSMODEL
- TSPOOL
- TYPETERM
- UOW
- UOWDSNFAIL
- UOWENQ

- UOWLINK
- URIMAP
- VOLUME
- VTAM
- WEB
- WEBSERVICE
- WORKREQUEST

QUERY SECURITY RESTYPE enables an application program 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 is 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 is 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 may 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, 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=*prefix*

When SEC=YES, XCMD=$USRCMD, and SECPRFX=*prefix* 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 `prefix.TRN1` in the TCICSTRN class or GCICSTRN group class.

The RESCLASS option

Use the QUERY SECURITY command with the RESCLASS option 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 “RACF classes for protecting system resources” on page 25. 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, nor with a user-specified prefix, 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.

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 activated by RACLIST. For example, if SEC=YES, and XCMD=YES are specified, both CCICSCMD and VCICSCMD are activated by RACLIST 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).

You can also use the RESCLASS option for querying access to DB2ENTRY resources defined in a user-defined resource class, which you specify to CICS on the XDB2 system initialization parameter. The rules about activating classes by means of the RACLIST command also apply to DB2ENTRY resource classes named on the XDB2 system initialization parameter. See “Resource classes for DB2ENTRYs” on page 28 for more information about user-defined DB2ENTRY resource clases.

Issuing QUERY SECURITY RESCLASS('TERMINAL') checks profiles in both TERMINAL and GTERMINL (the terminal grouping class) only if the TERMINAL class has been activated by RACLIST 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 294 for details.

Querying a user's surrogate authority

To query a user's surrogate authority, you can use the QUERY SECURITY command with the RESCLASS('SURROGAT') option. You also need to specify the RESID and RESIDLENGTH options. The RESID value you should provide is described in “RESID values” on page 131. 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

You can control logging on the QUERY SECURITY command by specifying one of the following options:

- LOG
- NOLOG
- LOGMESSAGE(*cvda*), where *cvda* value is 54 for LOG, or 55 for NOLOG

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 *z/OS Security Server RACF Auditor's Guide*.

For programming information about CVDAs, refer to *CICS System Programming Reference*.

Using the QUERY SECURITY command

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

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 each 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 records 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 295.

Chapter 10. Security for CICS-supplied transactions

This topic 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 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.

Categories of CICS-supplied transactions

For the purposes of this discussion, we divide the RACF profile definitions 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 are:

Category 1 transactions
Transactions that are never associated with a terminal—that is, they are for CICS internal use only, and should not be invoked from a user terminal.

Category 2 transactions
Transactions that are initiated by the terminal user, or are associated with a terminal, and for which access should be restricted to specific signed-on users.

Category 3 transactions
Transactions that are either initiated by the terminal user, or are associated with a terminal, and for which access is required by all terminal users, whether signed-on or not.

The three categories contain all 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.

See Chapter 23, “Implementing CICSPlex SM security,” on page 317 for details of CICSPlex SM-related transactions.

By default, all CICS transactions are subject to RACF protection (with the exception of category 3 transactions—see “JES spool protection in a CICS environment” on page 55), 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 each transaction attach, other than a transaction within category 3, to establish whether the user is permitted to run that transaction.

The following 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.

CDBN DBCTL interface connection transaction

CEKL Master terminal transaction for emergency use. This transaction can be used only at an operating system console that has the authority to issue MODIFY commands for the CICS region.

CSXM The transaction used by CICS services to get and free a transaction environment

Category 1 transactions

Category 1 transactions are never associated with a terminal. 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 CICS region userids.

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 *CICSTS32.CICS.SDFHSAMP*.

The following table lists the category 1 transactions supported in CICS Transaction Server for z/OS, Version 3 Release 2.

Note: In addition to the transactions described in the table below, transactions CSGX, CSLG, and CSSX are included in DFH$CAT1 for compatibility reasons.

Table 15. Category 1 transactions

Transaction	Program	CSD group	Security category	Description
CATA	DFHZATA	DFHSPI	1	Defines autoinstall automatic terminal

Table 15. Category 1 transactions (continued)

Transaction	Program	CSD group	Security category	Description
CATD	DFHZATD	DFHSPI	1	Deletes autoinstall terminal
CDBD	DFHDBDI	DFHDBCTL	1	DBCTL disable function
CDBF	DFHD2CM3	DFHDB2	1	CICS DB2 attachment facility shutdown force transaction
CDBO	DFHDBCT	DFHDBCTL	1	DBCTL control function
CDBQ	DFHD2CM2	DFHDB2	1	CICS DB2 attachment facility shutdown quiesce transaction
CDTS	DFHZATS	DFHSPI	1	Provides remote single delete transaction
CESC*	DFHCESC	DFHSIGN	1	Processes timeout and signoff for idle terminals
CEX2	DFHD2EX2	DFHDB2	1	CICS DB2 protected thread purge mechanism and other CICS DB2 services.
CFCL	DFHFCDL	-	1	CFDT Load
CFOR	DFHFCQT	-	1	RLS offsite recovery
CFQR	DFHFCQT	-	1	RLS quiesce receive
CFQS	DFHFCQT	-	1	RLS quiesce send
CFTL	DFHDTLX	-	1	Shared DT load
CFTS	DFHZATS	DFHSPI	1	Provides remote mass flag transaction
CGRP	DFHZCGRP	-	1	Provides VTAM persistent sessions (See note)
CIOD	DFHIIOPA	DFHIIOP	1	Default IIOP interface, started by CIOR
CIOF	DFHIIOPA	DFHIIOP	1	CICS Generic factory, started by CIOR
CIOR	DFHIIOP	DFHIIOP	1	CICS IIOP interface, started by SO_Domain
CIRR	DFHIIRRS	DFHIIOP	1	Default CICS IIOP request receiver
CISC	DFHISCOP	DFHISCIP	1	IPIC acquire IPCONN on the client side of a connection
CISD	DFHISCOP	DFHISCIP	1	IPIC release IPCONN on the client side of a connection
CISE	DFHISCIP	DFHISCIP	1	IPIC error and message program
CISR	DFHISRRP	DFHISCIP	1	IPIC request/response receiver
CISS	DFHISCOP	DFHISCIP	1	IPIC acquire IPCONN on the server side of a connection
CIST	DFHISCOP	DFHISCIP	1	IPIC terminate IPCONN
CISX	DFHISREX	DFHISCIP	1	IPCONN recovery and resynchronization transaction for XA clients
CITS	DFHZATS	DFHSPI	1	Provides remote autoinstall transaction
CJGC	DFHSJGC	DFHJAVA	1	CICS JVM garbage collection transaction
CJMJ	DFHSJJM	DFHJAVA	1	Starts the master JVM
CJPI	DFHSJPI	DFHJAVA	1	Starts JVMs following a PERFORM JVMPOOL command

Table 15. Category 1 transactions (continued)

Transaction	Program	CSD group	Security category	Description
CJTR	DFHOTR	DFHOTS	1	CORBA Object Transaction Services (OTS) resynchronization transaction
CKAM	DFHMQMON	DFHMQ	1	CICS-MQ Adapter alert monitor
CKTI	DFHMQTSK	DFHMQ	1	CICS-MQ Adapter - task initiator transaction
CMTS	DFHZATS	DFHSPI	1	Remote mass delete transaction
COVR	DFHZCOVR	-	1	Provides open VTAM retry transaction
CPIR	DFHPIITL	DFHSTAND	1	Pipeline resolution transaction
CPIS	DFHPIR	DFHPIPE	1	WS-AT transaction that is attached when resynchronization is required
CPLT	DFHSIPLT	-	1	Initializes PLT processing
CRMD	DFHZATMD	DFHSPI	1	Provides remote mass delete transaction
CRMF	DFHZATMF	DFHSPI	1	Provides remote mass flag transaction
CRSQ	DFHCRQ	DFHISC	1	Remote schedule purging (ISC)
CRSY	DFHRMSY	DFHRMI	1	Resource manager resynchronization
CRTP	DFHZRTP	DFHPSSGN	1	Persistent sessions restart timer transaction
CSFR	DFHFCRD	-	1	RLS cleanup
CSFU	DFHFCU	DFHOPCLS	1	File open utility
CSHA	-	-	1	Scheduler services (Autoinstalled by CICS)
CSHQ	DFHSHSY	-	1	Scheduler services domain long running task
CSKP	DFHAKP	DFHCOMP5	1	Writes system log activity keypoint
CSNC	DFHCRNP	DFHISC	1	Interregion control program (MRO)
CSNE	DFHZNAC	DFHVTAM	1	Provides VTAM node error recovery
CSOL	DFHSOL	-	1	TCP/IP listener (Autoinstalled by CICS)
CSPQ	DFHTPQ	DFHBMS	1	Terminal page cleanup (BMS)
CSQC	DFHLGQC	DFHLGQC	1	CICS quiesce after system log failure
CSSY	DFHAPATT	-	1	Provides entry point attach (See note)
CSTE	DFHTACP	DFHSTAND	1	Processes terminal abnormal conditions
CSTP	DFHZCSTP	-	1	Provides terminal control transaction (See note)
CSZI	DFHSZRMP	DFHFEPI	1	Front End Programming Interface (FEPI), only active if FEPI installed.
CTSD	DFHTSDQ	-	1	Temporary storage delete recoverable queue
CWBG	DFHWBGB	DFHWEB	1	CICS Web support cleanup transaction
CWXN	DFHWBXN	DFHWEB	1	CICS Web support attach transaction
CWXU	DFHWBXN	DFHWEB	1	CICS Web support USER protocol attach transaction
CXCU	DFHCXCU	DFHSTAND	1	Performs XRF tracing catchup

Table 15. Category 1 transactions (continued)

Transaction	Program	CSD group	Security category	Description
CXRE	DFHZXRE	DFHSTAND	1	Reconnects terminals following XRF takeover
Note: The transactions that have operator interfaces are marked by an asterisk (*). The remainder therefore have no operator interface.				

Category 2 transactions

Category 2 transactions are initiated by the terminal user or are associated with a terminal. Restrict authorizations to initiate these transactions to user IDs 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, CEST, and CIRP 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. You are **not** recommended to 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 does not have to 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; consider defining them in several subcategories. In the examples that follow, the category 2 transactions are further subdivided into a number of groups. Note that these groups are only examples. You can choose to group CICS transactions in the way that best suits your installations needs.
- SYSADM, containing: CCRL, CDBC, CEDA, CEMN, CEMT, CESD, CETR, CIDP, CIND, and CREA
- DEVELOPER, containing: CADP, CEBR, CECI, CECS, CEDB, CEDF, and CEDX
- INQUIRE, containing CDBI, CEDC, CKBM, CKDL, CKDP, CKQC, CKRT, CLDM, and CREC.
- OPERATOR, containing: CBAM, CEOT, CEST, CIDP, CMSG, CRTE, CSFE, CWTO, and DSNC
- INTERCOM, containing: CDFS, CEHP, CEHS, CPMI, CSHR, CSMI, CSM1, CSM2, CSM3, CSM5, CTIN, 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. In this case, the following conditions apply:
  - The terminal user running the application must be authorized to use the mirror transaction. (See Chapter 5, “Transaction security,” on page 81.)
  - The terminal user must also be authorized to use the data that the mirror transaction accesses. (See Chapter 6, “Resource security,” on page 87.) 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, change the mirror transaction definitions to specify RESSEC(NO). Because the mirror transactions are an IBM-protected resource, first copy these definitions into your own groups and then change them.

- RPCUSER, containing: CRPA, CRPC, and CRPM
- AFFINITIES, containing: CAFB and CAFF
- PIPEUSER, containing: CPIH, CPIL, CPIQ, and CPIA
- WEBUSER, containing: CWBA

  The CICS default user requires access to the CWBA transaction initially, even if a security analyzer is then used to assign another user ID to the task. Ensure that the CICS default user that is specified in the **DFLTUSER** system initialization parameter has access to this transaction. If you use the supplied CLIST DFH$CAT2 to create a WEBUSER RACF profile, the default user must have access to this profile.
- ALLUSER, containing: CMAC, CRTX, and CSGM

  You are recommended to define the CICS “messages and codes” transaction CMAC, and the “good morning” transaction 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. Also include your “goodnight” transaction in this group, if you defined one with the **GNTRAN** system initialization parameter
- IIOPUSER, containing: CIRP
- DBCTL, containing: CDBC, CDBI, CDBM and CDBT

The sample CLIST DFH$CAT2 (in library *CICSTS32.CICS.SDFHSAMP*) can 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 4 on page 145 shows how to use RDEFINE and PERMIT commands to define the example groups for category 2 transactions.

```
RDEFINE GCICSTRN INQUIRE UACC(NONE)
         ADDMEM(CDBI, CEDC, CKBM, CKDL, CKDP, CKQC, CKRT, CLDM, and CREC)
         NOTIFY(security_admin_userid)
         OWNER(userid or groupid)
PERMIT   SYSADM  CLASS(GCICSTRN) ID(sysgrp1,..,sysgrpz) ACCESS(READ)
RDEFINE GCICSTRN DEVELOPER UACC(NONE)
         ADDMEM(CADP,CEDF,CEBR,CECI,CECS,CEDB,CEDX)
         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,CREC)
         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,CIDP,CSFE,DSNC,CBAM)
         NOTIFY(security_admin_userid)
         OWNER(userid or groupid)
PERMIT   OPERATOR  CLASS(GCICSTRN) ID(opsgrp1,..,opsgrpz) ACCESS(READ)
RDEFINE GCICSTRN DBCTL UACC(NONE)
         ADDMEM(CDBC,CDBI,CDBM,CDBT)
         NOTIFY(security_admin_userid)
         OWNER(userid or groupid)
PERMIT   DBCTL  CLASS(GCICSTRN) ID(dbctgrp1,..,dbctgrpz) ACCESS(READ)
RDEFINE GCICSTRN INTERCOM UACC(NONE)
         ADDMEM(CEHP,CEHS,CPMI,CSHR,CSMI,CSM1,CSM2,CSM3,CSM5,CVMI,CDFS,CTIN)
         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 ALLUSER CLASS (GCICSTRN) ID(allrgrp1,..,allrgrpz) ACCESS(READ)
RDEFINE GCICSTRN WEBUSER UACC(NON)
         ADDMEM(CWBA)
         NOTIFY(security_admin_userid)
         OWNER(userid or groupid)
PERMIT WEBUSER CLASS (GCICSTRN) ID(webrgrp1,..,webgrpz) ACCESS(READ)
RDEFINE GCICSTRN RPCUSER UACC(NON)
         ADDMEM(CRPA,CRPC,CRPM)
         NOTIFY(security_admin_userid)
         OWNER(userid or groupid)
PERMIT RPCUSER CLASS (GCICSTRN) ID(rpcrgrp1,..,rpcrgrpz) ACCESS(READ)
RDEFINE GCICSTRN IIOPUSER UACC(NONE)
         ADDMEM(CIRP)
         NOTIFY(security_admin_userid)
         OWNER(userid or groupid)
PERMIT IIOPUSER CLASS (GCICSTRN) ID(iiopgrp1,..,iiopgrpz) ACCESS(READ)
RDEFINE GCICSTRN AFFINITIES UACC(NONE)
         ADDMEM(CAFF,CAFB)
         NOTIFY(security_admin_userid)
         OWNER(userid or groupid)
PERMIT AFFINITIES CLASS(GCICSTRN) ID(affngrp1,..,affngrpz) ACCESS(READ)
RDEFINE GCICSTRN PIPEUSER UACC(NONE)
         ADDMEM(CPIH,CPIL,CPIQ,CPIA)
         NOTIFY(security_admin_userid)
         OWNER(userid or groupid)
PERMIT PIPEUSER CLASS(GCICSTRN) ID(pipeline_access_list)
```

Figure 4. Example of defining groups for category 2 transactions

Note:

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 that the transactions use.
2. If you protect a resource with a resource group profile, you must 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. The merging can have a performance impact and it can be difficult to determine exactly which access authority applies to a particular user. See the *z/OS Security Server RACF Security Administrator's Guide* for further information.

Table 16. Category 2 transactions

Transaction	Program	CSD group	Security category	Description
CADP*	DFHDPLU	DFHDP	2	Application debugging profile manager
CBAM*	DFHECBAM	DFHOPER	2	BTS objects browser
CCRL*	DFHSOCRL	DFHOPER	2	CICS certificate revocation list transaction
CDBC*	DFHDBME	DFHDBCTL	2	DBCTL interface menu transaction
CDBI*	DFHDBIQ	DFHDBCTL	2	DBCTL interface inquiry transaction
CDBM*	DFHDBMP	DFHDBCTL	2	DBCTL operator transaction
CDBT	DFHDBDSC	DFHDBCTL	2	DBCTL interface disconnection transaction
CDFS	DFHDFST	DFHISC	2	Dynamic starts with interval
CEBR*	DFHEDFBR	DFHEDF	2	Browse temporary storage
CECI*	DFHECIP	DFHINTER	2	Command level interpreter
CECS*	DFHECSP	DFHINTER	2	Command level interpreter
CEDA*	DFHEDAP	DFHSPI	2	Resource definition online - full (RDO)
CEDB*	DFHEDAP	DFHSPI	2	Resource definition online - restricted (RDO)
CEDC*	DFHEDAP	DFHSPI	2	Views resource definition online (RDO)
CEDF*	DFHEDFP	DFHEDF	2	Execution diagnostic facility
CEDX*	DFHEDFP	DFHEDF	2	Execution diagnostic facility for non-terminal tasks
CEHP	DFHCHS	DFHISC	2	CICS OS/2 request handler
CEHS	DFHCHS	DFHISC	2	CICS/VM request handler
CEMN	DFHCEMNA	DFHOPER	2	CICS monitoring facility transaction
CEMT*	DFHEMTP	DFHOPER	2	Master terminal
CEOT*	DFHEOTP	DFHOPER	2	Terminal status
CESD	DFHCESD	DFHSDAP	2	Shutdown assist
CEST*	DFHESTP	DFHOPER	2	Supervisory terminal
CETR*	DFHCETRA	DFHOPER	2	Inquire and set trace options
CIDP*	DFHDPIN	DFHDP	2	Inactivate debugging profiles utility
CIND*	DFHINDT	DFHINDT	2	CICS in-doubt testing tool
CIRP	DFHIIRP	DFHIIOP	2	Default CICS IIOP request processor transaction

Table 16. Category 2 transactions (continued)

Transaction	Program	CSD group	Security category	Description
CKBM	DFHMQBAS	DFHMQ	2	CICS-MQ Adapter base panel transaction
CKBP	CSQCBP00	DFHMQ	2	CICS-MQ Bridge DPL bridge task
CKCN	DFHMQQCN	DFHMQ	2	CICS-MQ Adapter start connection transaction
CKDL	DFHMQDSL	DFHMQ	2	CICS-MQ Adapter, display status transaction
CKDP	DFHMQDIS	DFHMQ	2	CICS-MQ Adapter, display transaction
CKRS	DFHMQRS	DFHMQ	2	CICS-MQ Adapter modify transaction
CKRT	DFHMQRET	DFHMQ	2	CICS-MQ Adapter screen return transaction
CKSD	DFHMQDSC	DFHMQ	2	CICS-MQ Adapter stop connection transaction
CKSQ	DFHMQSSQ	DFHMQ	2	CICS-MQ Adapter start/stop CKTI transaction
CKQC	DFHMQCTL	DFHMQ	2	CICS-MQ Adapter control transaction
| CLDM*	DFHLDMAP	DFHOPER	2	CICS load module map
CMAC	DFHCMAC		2	Messages utility
CMSG*	DFHMSP	DFHMSWIT	2	Message switching
CPIH	DFHPIDSH	DFHPIPE	2	CICS pipeline HTTP inbound router
CPIL	DFHPILSQ	DFHPIPE	2	SOAP WMQ inbound listener
CPIQ	DFHPIDSQ	DFHPIPE	2	SOAP WMQ inbound router
CPMI	DFHMIRS	DFHISC	2	CICS LU 6.2 synchronization level 1 mirror
CREA*	DFHADDRM	DFHADST	2	Create REQUESTMODELs for enterprise beans
CREC*	DFHADDRM	DFHADST	2	Create REQUESTMODELs for enterprise beans (read only)
CRPA	DFHRPAS	DFHRPC	2	ONC/RPC Alias transaction
CRPC	DFHRPC00	DFHRPC	2	ONC/RPC Update transaction
CRPM	DFHRPHS	DFHRPC	2	ONC/RPC Server controller
CRTE*	DFHRTE	DFHISC	2	Transaction routing
CRTX	-	DFHISC	2	Dynamic transaction routing transaction definition
CSFE*	DFHFEP	DFHFE	2	Terminal test, trace, storage
CSGM	DFHGMM	DFHVTAM	2	“Good-morning” signon
CSHR	DFHMIRS	DFHISC	2	Scheduler services remote routing
CSM1	DFHMIRS	DFHISC	2	SYSMSG model
CSM2	DFHMIRS	DFHISC	2	Scheduler model
CSM3	DFHMIRS	DFHISC	2	Queue model
CSM5	DFHMIRS	DFHISC	2	DL/I model
CSMI	DFHMIRS	DFHISC	2	Mirror transaction
CTIN	DFHZCT1	DFHCLNT	2	CICS Client

Table 16. Category 2 transactions (continued)

Transaction	Program	CSD group	Security category	Description
CVMI	DFHMIRS	DFHISC	2	CICS LU6.2 synchronization level 1 mirror
CWBA	DFHWBA	DFHWEB	2	CICS Web support alias transaction
CWTO*	DFHCWTO	DFHCONS	2	Write to console operator
DSNC	DFHD2CM1	DFH$DB2	2	DB2 attachment facility transaction
Note: The transactions that have operator interfaces are marked by an asterisk (*). The remainder therefore have no operator interface.				

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; although this definition does not affect task attach-time processing, it is required to support the QUERY SECURITY command.

Table 17. Category 3 transactions

Transaction	Program	CSD group	Security category	Description
CATR	DFHZATR	DFHSPI	3	Deletes autoinstall restart terminal
CCIN	DFHZCN1	DFHCLNT	3	CICS Client
CEGN	DFHCEGN	DFHSIGN	3	Schedules good night transaction
CEJR	DFHEJITL	DFHSTAND	3	Corbaserver resolution
CESF*	DFHSFP	DFHSIGN	3	Signs off terminal user
CESN*	DFHSNP	DFHSIGN	3	Signs on terminal user
CIEP	DFHIEP	DFHIPECI	3	ECI for TCP/IP listener
CLQ2	DFHLUP	DFHISCT	3	Outbound resynchronization for APPC and MRO
CLR1	DFHZLS1	DFHISCT	3	Inbound CNOS for APPC and MRO
CLR2	DFHLUP	DFHISCT	3	Inbound resynchronization for MRO
CLS1	DFHZLS1	DFHISC	3	Provides ISC LU services model
CLS2	DFHLUP	DFHISC	3	Provides ISC LU services model
CLS3	DFHCLS3	DFHISC	3	ISC LU services model
CLS4	DFHCLS4	DFHISC	3	Manages password expiry
CMPX	DFHMXP	DFHISC	3	Ships ISC local queuing
CPCT	DFHZPCT	DFHPSSGN	3	Catalog signed on terminals for persistent session signon retention

Table 17. Category 3 transactions (continued)

Transaction	Program	CSD group	Security category	Description
CPSS	DFHZSGN	DFHPSSGN	3	Persistent sessions signon
CQPI	DFHCLS5	DFHISC	3	Connection quiesce. Architected transaction (inbound).
CQPO	DFHCLS5	DFHISC	3	Connection quiesce. Architected transaction (outbound).
CQRY	DFHQRY	DFHSTAND	3	Provides ATI query support
CRSR	DFHCRS	DFHISC	3	Provides ISC remote scheduler
CSAC	DFHACP	DFHSTAND	3	Provides program abnormal condition
CSCY	DFHCPY	DFHVTAMP	3	Provides 3270 screen print
CSPG*	DFHTPR	DFHBMS	3	Provides BMS terminal paging
CSPK	DFHPRK	DFHVTAMP	3	Provides 3270 screen print support
CSPP	DFHP3270	DFHHARDC	3	Provides 3270 print support
CSPS	DFHTPS	DFHBMS	3	Schedules BMS terminal paging
CSRK	DFHRKB	DFHVTAMP	3	Provides 3270 screen print - release keyboard
CSRS	DFHZRSP	DFHRSEND	3	Synchronizes 3614 message
CSSF	DFHRTC	DFHISC	3	Cancels CRTE transaction routing session
CXRT	DFHCRT	DFHISC	3	Provides Transaction routing relay
Note: The transactions that have operator interfaces are marked by an asterisk (*). The remainder therefore have no operator interface.				

Part 3. Intercommunication security

This part tells you how to plan and implement security in an intersystem communication (ISC) environment, using LU6.2 or LU6.1, or in a multiregion operation (MRO) environment.

Chapter 11. Overview of intercommunication security

This topic gives an overview of how security works when CICS systems are interconnected or connected to other compatible systems.

Introduction to intercommunication security

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 now you 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.

This topic assumes that you are already familiar with setting up security for a single CICS system.

In particular, you should understand the following concepts:

- User signon. (See "The sign-on process" on page 65.)
- 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 65 and Chapter 5, "Transaction security," on page 81.)
- How resource security determines which other resources a user is allowed to access. (See Chapter 6, "Resource security," on page 87.)

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 16 and "RACF group profiles" on page 21 for information on defining these profiles.) That is, you may have to define these profiles in each 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, distributed transaction processing, or external call interface (EXCI). (For descriptions of these methods of intercommunication, see the *CICS Intercommunication Guide* and the *CICS 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 issue 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.

Intercommunication 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**. (For ISC over SNA connections, it is 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 to all of the following:
- The **SNA BIND** command that is used to establish SNA sessions between systems
- The **CICS connection request** that is used to establish an IPIC connection between systems
- The CICS connection request that is used to establish MRO sessions for CICS interregion communication

You can specify bind-time security for APPC (LU6.2), IPIC , and multiregion operation (MRO) links, but **not** for LU6.1 links. For information on defining bind-time security in your system, see “Bind-time security with LU6.2” on page 157, “IPIC bind-time security” on page 205, or “Bind-time security with MRO” on page 211, depending on the environment you are using.

Intercommunication 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 for intercommunication.”

User security for intercommunication

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. For ISC over SNA and MRO links, this is done by specifying the appropriate ATTACHSEC parameters on the CONNECTION definition. For IPIC links, it is done by specifying the ATTACHSEC parameter on both the IPCONN and TCPIPSERVICE definitions.

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 intercommunication.”

For information on defining user security in your system, see “User security with LU6.2” on page 162, “IPIC user security” on page 207, or “User security with MRO” on page 215, 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 for intercommunication

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 81, Chapter 6, “Resource security,” on page 87, and Chapter 8, “CICS command security,” on page 119.

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 168
- “Transaction, resource, and command security with LU6.1” on page 200
- “IPIC transaction, resource, and command security” on page 208
- “Transaction, resource, and command security with MRO” on page 217

Summary of intercommunication security levels

Table 18 on page 156 shows bind-time, transaction, resource, and command security, and how CICS enforces these levels of security under the LU6.1, LU6.2, IPIC, and MRO 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 Intercommunication Guide*.

Table 18 on page 156 shows a summary of intercommunication security.

Table 18. Summary of intersystem and interregion security

Security level	Security checks	LU type 6.1	LU type 6.2	IPIC	MRO
Bind-time security (when BIND is received)	Should the BIND request be accepted?	No check	Session key from RACF	SSL Client Certificate from partner	DFHAPPL profiles in RACF FACILITY class
Bind-time security (when BIND is sent)	Is the remote system the correct one?	No check	Session key from RACF	SSL Client Certificate to partner	DFHAPPL profiles in RACF FACILITY class
Link security	Does the link have authority to attach the transaction?	Link authority is established just after the session is bound, by signon of the user ID specified in the SECURITYNAME attribute of the CONNECTION definition or the USERID attribute of the SESSIONS definition.		Link authority is established just after the connection is established, by signon of a user ID determined according to the IPCONN LINKAUTH attribute.	Link authority is established just after the session is bound, by signon of the user ID specified in the USERID attribute of the SESSIONS definition.
Transaction security	Does the remote user have authority to access this system?	No check	The authority of the remote user is established at signon time		
Transaction security	Does the remote user have authority to attach the transaction?	Link authority	The authority of the remote user is established at this attach request (or possibly at an earlier attach request from the same user) by sign-on		
Resource, command, and surrogate security	Does the session have the authority to access other resources that the transaction uses?	Link authority is established just after the session is bound, by signon of the user ID specified in the SECURITYNAME attribute of the CONNECTION definition or the USERID attribute of the SESSIONS definition.		Link authority is established just after the connection is established, by signon of a user ID determined according to the IPCONN LINKAUTH attribute.	Link authority is established just after the session is bound, by signon of the user ID specified in the USERID attribute of the SESSIONS definition.
Resource, command, and surrogate security	Does the remote user have authority to access other resources that the transaction uses?	Link authority	The authority of the remote user is established at this attach request (or possibly at an earlier attach request from the same user) by sign-on		

Note: Remember to define profiles for your resources and users to RACF, as described for single systems in Chapter 2, “RACF facilities,” on page 13.

Chapter 12. Implementing LU6.2 security

This topic tells you how to implement security for LU6.2.

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 should not specify bind-time security if the remote system does not support it. SNA defines how session security is to be applied, and CICS TS conforms to this architecture. If you want to connect to another system (including CICS systems before CICS/ESA), 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, 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 attribute of the CONNECTION definition
- 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 19 summarizes what happens.

Table 19. 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.

Table 19. APPC bind-time security—relationship to resource definition (continued)

SEC value	XAPPC value	BINDSECURITY value	RACF APPCLU profile	Resulting CICS action
YES	YES	NO	Any value	CICS is unable to validate the bind, and rejects it.
YES	NO	Any value	Any value	CICS is unable to validate the bind, and rejects it.
NO	Any value	Any value	Any value	CICS is unable to validate the bind, and rejects it.

Note:

1. If the RACF APPCLU profile is defined, but the session segment is locked or expired, and 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).

Defining profiles in the APPCLU general resource class

If you use bind-time security with LU6.2, you must define profiles in the APPCLU general resource class: the APPCLU resource class is used to verify the identity of APPC partner logical units (LU type 6.2) during VTAM session establishment.

To do this, take the following steps:

1. Ask your VTAM system programmer for the following information for each session partner:
   - The network ID and the LU identifiers.
2. For each pair of session partners, create two profiles in the APPCLU general resource class.

   On one system, enter the following RDEFINE command:
   ```
   RDEFINE APPCLU netid1.luid1.luid2 UACC(NONE)
           SESSION(SESSKEY(password))
   ```
   On the other system, enter the following RDEFINE command:
   ```
   RDEFINE APPCLU netid2.luid2.luid1 UACC(NONE)
            SESSION(SESSKEY(password))
   ```
   where:

   netid1
   netid2
      are the network IDs (NETID) of the partners. These IDs are specified on the VTAM start option NETID, which is in the ATCSTRxx member of SYS1.VTAMLST.

   luid1

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

session-key
: is the 16-hexadecimal-digit or 8-character password that matches the session key of the remote system. Enclose hexadecimal digits in quotes; for example, `SESSKEY(X'0123456789ABCDEF')`.

  You should specify the same session key in both systems: if the session keys do not match, the session cannot be established.

  Although RACF does not require that you specify a session key, CICS will reject the bind if no session key is specified.

3. Define the attributes of the sessions between the partners of each LU pair. To do this, define a SESSION segment for each APPCLU profile using the SESSION option of the RDEFINE and RALTER commands. You can specify the following information in each SESSION segment:

   CONVSEC
   Specifies the levels of security checking performed for each conversation between the partners of the LU pair. CICS does not use this information; instead it uses the information specified in the ATTACHSEC operand of the CONNECTION definition.

   INTERVAL
   Specifies the maximum number of days the session key is valid before it must be changed.

   You should 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 be written out. See "Specifying bind-time security for LU6.2." (CICS issues message DFHZC4942 to the CSNE destination when the password has expired.) Ensure that you are aware when a password interval is about to expire so that links do not fail for this reason. CICS does not display messages when the password is about to expire, but it does write records to the SMF log.

   LOCK
   Marks the profile as locked. If the profile is locked, the session does not bind, and CICS issues message DFHZC4941.

   NOLOCK
   Marks the profile as unlocked.

For more information about controlling on controlling VTAM LU6.2 binds, see the *z/OS Security Server RACF Security Administrator's Guide*.

Specifying bind-time security for LU6.2

You define bind-time security in the CONNECTION definition, although you must also choose the appropriate system initialization parameters. Figure 5 on page 160 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)
```

Figure 5. Bind-time security

Note: For APPC terminals defined as a TERMINAL-TYPETERM pair, the BINDSECURITY operand is on the TERMINAL definition.

Auditing bind-time security

If security is active (SEC=YES is 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 six 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:

- Message ICH700051 is sent to the userid specified in the profile's notify option. 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 *z/OS Security Server RACF Auditor's Guide*.

Changing RACF profiles that are in use—caution

Take care when changing RACF profiles for APPC connections that are in use. CICS recognizes the change in the profile after a SETROPTS RACLIST(APPCLU) REFRESH command is issued. Bind-time security processing occurs when each session in a connection is acquired. Not all the sessions in a connection are acquired and the APPC profile becomes invalid, then an attempt to establish any of the unacquired sessions causes 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:

- The session key expires
- The session key changes and a SETROPS REFRESH takes place in one system without the corresponding change and refresh occurring in the other system
- The profile is locked while REFRESH takes place.

Sessions that are already acquired still continue to function normally if bind security fails in 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 before the date of expiration, and have remained acquired. 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 is used that will expire shortly. 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 because the connection is unavailable because of an expired session key. For example, you should plan for the changing of the session keys, for security rebuilds (for both CICS systems) and for the possibility of having to reacquire the connection.

You can avoid the problem of APPC profiles becoming invalid while the connection is in use 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.

When link security is in use, each session is given an authority defined by a *link userid*. For LU6.2, all sessions in a connection can have the same link userid, or different groups of sessions within the connection can have different link userids. You can also specify that some groups of sessions should use link security, and that others should not.

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 link userid matches the local region userid:

- If the userids match, 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 the userids do not match, then 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 the security of the local region's default user. This would happen, for example, if the preset session userid had been revoked.

Specifying link security for LU6.2 connections

- To specify that all sessions of a connection should have the same link userid, specify the SECURITYNAME attribute of the CONNECTION definition. If you do not specify a value for this attribute, CICS uses the default user ID.
- To specify different link userids for individual groups of sessions within a connection, specify the USERID attribute of the SESSIONS definition. For each group of sessions, the value specified overrides the SECURITYNAME attribute of the CONNECTION definition.

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 161. After reading the following descriptions, 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 164 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” further describes these types of user checking checking. See “Specifying user security in link definitions” on page 164 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 sign-on mechanism) before letting them use the link. Use IDENTIFY if you want user security for CICS-to-CICS communication (CICS does not support password flows on CICS-to-CICS connection).

Note: This excludes CICS Transaction Server for OS/2.

If the front end does not have a security manager it may not be 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. VERIFY must be used if the request comes from CICS for OS/2, which does not support PERSISTENT.

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 sign-on 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, to what extent 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.
- Does the remote system support persistent verification by keeping track of its user identifiers and passwords? If so, specify PERSISTENT, unless you are using CICS-to-CICS communications, in which case specify MIXIDPE.

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."

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 6.

This topic 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 169. Figure 6 shows an example of defining ATTACHSEC using CEDA.

```
CEDA DEFINE CONNECTION(name)
  GROUP(groupname)

  ATTACHSEC(LOCAL|IDENTIFY|VERIFY|PERSISTENT|MIXIDPE)
```

Figure 6. Defining sign-on level for user security

Note: For APPC terminals defined as a TERMINAL-TYPETERM pair, the ATTACHSEC operand is on the TERMINAL definition.

The ATTACHSEC operand specifies the sign-on 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. 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') user identifier or an unknown user identifier is received, CICS rejects the attach request.

VERIFY
specifies that, in addition to a user identifier, a user password is required for verification against the local RACF database. All remote users of a system must be identified to RACF.

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, CICS rejects the attach request. If the communicating system is CICS for AIX, ATTACHSEC=IDENTIFY should be used.

Note: Products other than CICS can connect to a CICS Transaction Server for z/OS 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 167.

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.

Sign-on 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 sign-on 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 System Definition Guide*. See the *CICS 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 sign-on 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 sign-on 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 revalidated for each attach request. For information on specifying a value for PVDELAY, see the *CICS System Definition Guide*. See the *CICS 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 sign-off 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 user at the remote system is making the request.

This topic 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.

If the user has been added to the front-end system with a group ID 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 where ATTACHSEC(IDENTIFY) has been specified in the CONNECTION definition. If the group ID 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 group ID is passed to the back-end system, the group ID will be used as part of ADD_USER processing on the back-end. That is, the user ID 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 20 shows how CICS decides which userid to send.

Table 20. 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 transient data destination definition 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 "Sign-on status" on page 165.

SNA profiles and attach-time security

Implementation of the LU6.2 attach-time security in CICSTS32.CICS conforms strictly to the architecture. In particular, note the following:
- 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. Any trailing blanks ((X'40') are removed before being passed to the security manager, which either rejects the attach request, or converts the userid and password into 8- character form before proceeding.
- If an attach request does not contain security parameters in the FMH-5, it is rejected, unless USEDFLTUSER(YES) has been specified on the CONNECTION definition. In that case, the security capabilities of the default user apply.
- 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.

Transaction, resource, and command security with LU6.2

As in a single-system environment, users must be authorized to:

- Attach a transaction (**transaction security**)
- Access all the resources that the transaction is programmed to use. These levels are called **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 81.

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 171.

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 7 on page 169.

```
CEDA DEFINE TRANSACTION
  .
  RESSEC(YES)
  CMDSEC(YES)
  .
```

Figure 7. 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 system programming commands shown in Table 10 on page 119 that the attached transaction issues.

For further guidance on specifying resource and command security, see Chapter 6, “Resource security,” on page 87 and Chapter 8, “CICS command security,” on page 119.

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 161.

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 specified in the remote terminal definition, 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 transactions executed under CRTE does not depend on what is specified by ATTACHSEC, or on the userid signed on in the TOR. 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 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 does 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 143.

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 8 on page 172 summarizes what happens.

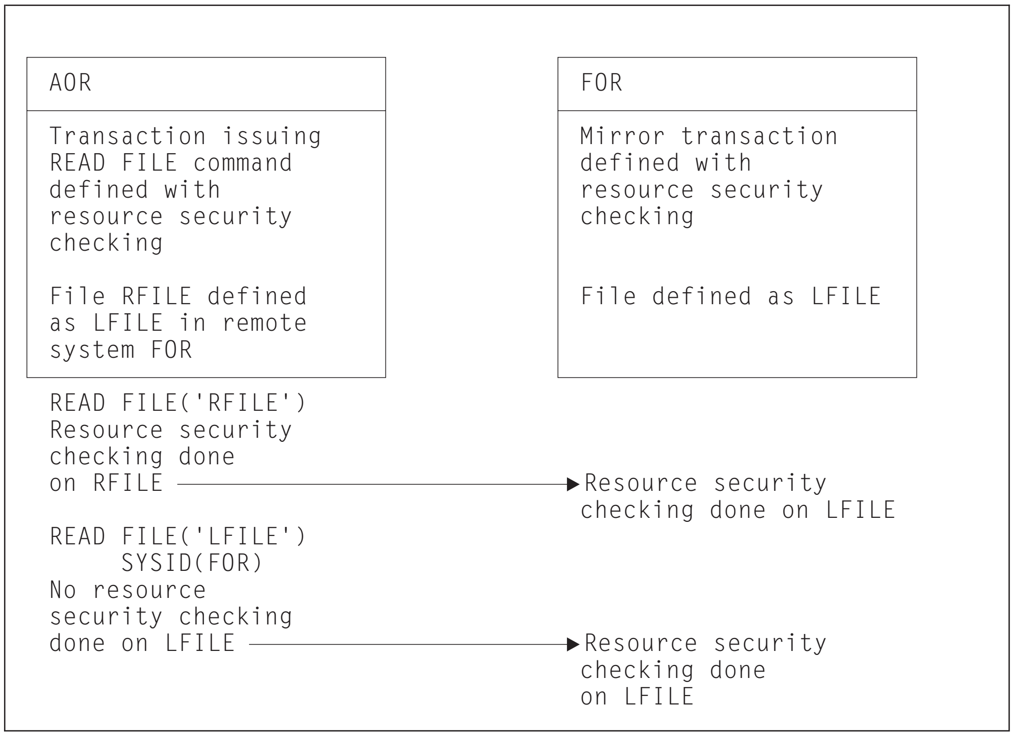

AOR
Transaction issuing READ FILE command defined with resource security checking
File RFILE defined as LFILE in remote system FOR
FOR
Mirror transaction defined with resource security checking
File defined as LFILE
READ FILE('RFILE') Resource security checking done on RFILE
Resource security checking done on LFILE
READ FILE('LFILE') SYSID(FOR) No resource security checking done on LFILE
Resource security checking done on LFILE

Figure 8. Security checking done with and without SYSID. This example illustrates what security checking is done when a transaction in an application-owning region issues a file control request against a remote file.

- *In the application-owning region (AOR), file RFILE is defined as remote, with a name of LFILE in the file-owning region (FOR). Resource security checking is active for the transaction that issues the file control request.*
- *In the FOR, resource security checking is active for the mirror transaction.*

There are two cases. In the first case:

1. *The transaction in the AOR issues EXEC CICS READ FILE('RFILE'). Resource security checking is performed for file RFILE.*
2. *The request is transmitted to the FOR, where resource security checking is performed for file LFILE.*

In the second case:

The transaction in the AOR issues EXEC CICS READ FILE('RFILE') specifying the SYSID option. Resource security checking is not performed for file RFILE.

The request is transmitted to the FOR, where resource security checking is performed for file LFILE.

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 LINK command, or the REMOTESYSTEM option of the PROGRAM resource definition, specifies a remote CICS region.

When the SYSID option on the 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 LINK command 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.

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” on page 174. 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 system programming 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 Application Programming Reference*. For further information about DPL, refer to the *CICS 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 21 and Table 22 on page 175 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, is relevant to security checking in the AOR.

Table 21 shows how checking is done when ATTACHSEC(LOCAL) is specified.

Table 21. 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

Table 21. LU6.2 and ATTACHSEC(LOCAL) (continued)

Region userid for AOR	**SECURITYNAME in connection definition**	**USERID in SESSION definition**	**Checking in AOR**
USERIDA	USERIDB	USERIDC	Check against USERIDC

Table 22 shows how checking is done when the ATTACHSEC parameter IDENTIFY (or PERSISTENT, or MIXIDPE) has been specified.

Table 22. LU6.2 and ATTACHSEC(IDENTIFY), ATTACHSEC(PERSISTENT), and ATTACHSEC(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

Chapter 13. APPC password expiration management

This topic contains information on advanced program-to-program communications (APPC) password expiration management (PEM).

Introduction to APPC password expiration management

This section introduces, and describes the benefits of, APPC password expiration management.

You may find it useful to copy and modify an example program. For your guidance sample programs are now shipped in library *CICSTS32.CICS.SDFHSAMP.* Their names are:

1. DFH$$.SNPW — PEM sample program for Windows NT
2. DFH$$.SNP2 — PEM sample program for OS/2

For examples of PEM requester and CICS PEM server user data produced by a program, see:
- "Sign-on with correct userid and password" on page 193
- "Sign-on with new password" on page 194
- "Response to correct sign-on data" on page 194
- "Response to incorrect data format" on page 196.

Note: In this topic the word 'sign-on' is used in the sense defined in the APPC architecture, which is different from the meaning used elsewhere in this book.

What APPC PEM does

APPC PEM with CICS provides receive support for an APPC architected sign-on transaction that verifies user ID, 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 using a non-APPC link, such as an LU2 3270-emulation session, to update the password.
- It provides APPC users with additional information regarding their sign-on 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 you require to use APPC PEM

To use APPC PEM, you need:

A PEM client
: The PEM client can be any APPC logical unit (LU) or node that is capable of initiating a conversation with the architected sign-on 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. APPC PEM enables users of such LUs or nodes to manage their password values within the ESM used by CICS. The PEM client is linked to a **PEM server**.

A PEM server
: The PEM server can be any APPC LU that supports APPC PEM. This chapter assumes that the PEM server is the one provided by CICS Transaction Server for z/OS, Version 3 Release 2. It is referred to in the rest of this book as the CICS PEM server.

An external security manager
: An external security manager, such as RACF, or an equivalent ESM, must be available to the PEM server.

The PEM client (requester) and the PEM server are linked by an APPC session. This configuration is shown in Figure 9.

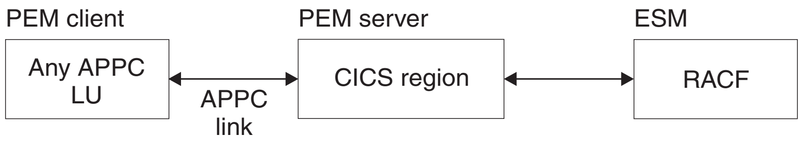


Figure 9. APPC PEM configuration

External security interface

Password expiration management has been enhanced to include the External Security Interface (ESI). The ESI is not part of CICS Transaction server for z/OS, but it allows a non-CICS application to invoke services provided by advanced APPC PEM. ESI provides additional functions which make it easier for a non-CICS application to change and verify a password.

The 2 functions provided by ESI are:

- **CICS_VerifyPassWord** which allows a client application to verify that a password matches the password recorded by RACF, or an equivalent external security manager, for a specified user ID.
- **CICS_ChangePassWord** which allows a client application to change the password recorded by RACF for a specified user ID.

These functions allow a non-CICS application program to act as a PEM requestor without the application programmer having to manage an APPC conversation which implies knowledge of the formats for PEM requests and replies, and of the interface to the local SNA server.

For more information about the ESI password management functions, see the *Client/Server Application Programming* manual.

Roles of PEM client and CICS PEM server

CICS Transaction Server for z/OS, Version 3 Release 2 does not send passwords on APPC conversations. This means that it can **attach**, but not **initiate** the sign-on transaction, and must always act as the PEM server. Therefore, in your configuration always include an LU that is capable of initiating the sign-on transaction to act as the PEM client.

The PEM client collects sign-on information and sends it to the CICS PEM server via a sign-on transaction program. The sign-on transaction program is a SNA service transaction program, as described in *SNA LU 6.2 Peer Protocols* manual.

Note that PEM signon is not to be confused with a CICS signon. In CICS, PEM signon allows the APPC LU to verify and manage user IDs and passwords. Following verification or updating, the user ID or password is intended to be included in 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 user ID will be signed on to CICS. Therefore, a PEM signon does not result in the ESM last—connect, last-access information being updated.

The CICS PEM server then processes the sign-on request, updates the user's password (if necessary), and returns a reply to the PEM client containing responses and other data, such as a password expiry and information about unauthorized attempts to sign on. The PEM client then processes the data, as appropriate.

Note: When you successfully verify your password with CICS Transaction Server for z/OS, Version 2 Release 1, you are not signed on in the target CICS system.

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.

An example of signing on with APPC PEM

This example shows the sequence of events when a user signs on to a PEM client with an expired password. The sequence is illustrated in Figure 10 on page 180.

1. The user attempts to sign on to the PEM client, and supplies a password.
2. The PEM client sends the user ID and password to the CICS PEM server.
3. The CICS PEM server passes the user ID and password to the external security manager.
4. The external security manager checks the password.
5. Because the password has expired, the sign on attempt fails. The external security manager informs the CICS PEM server, which, in turn, informs the PEM client.
6. The PEM client informs the user that the password has expired, and requests a new one.
7. The user enters a new password.
8. The PEM client sends the user ID, the old password, and the new password to the CICS PEM server.

9. The CICS PEM server passes the user ID, the old password, and the new password to the external security manager.
10. The external security manager checks the old and new password.Because a new passowrd is provided, the sign on attempt is succesful. The external security manager informs the CICS PEM server, which, in turn, informs the PEM client.
11. The PEM client informs the user that the sign on request has been successful.

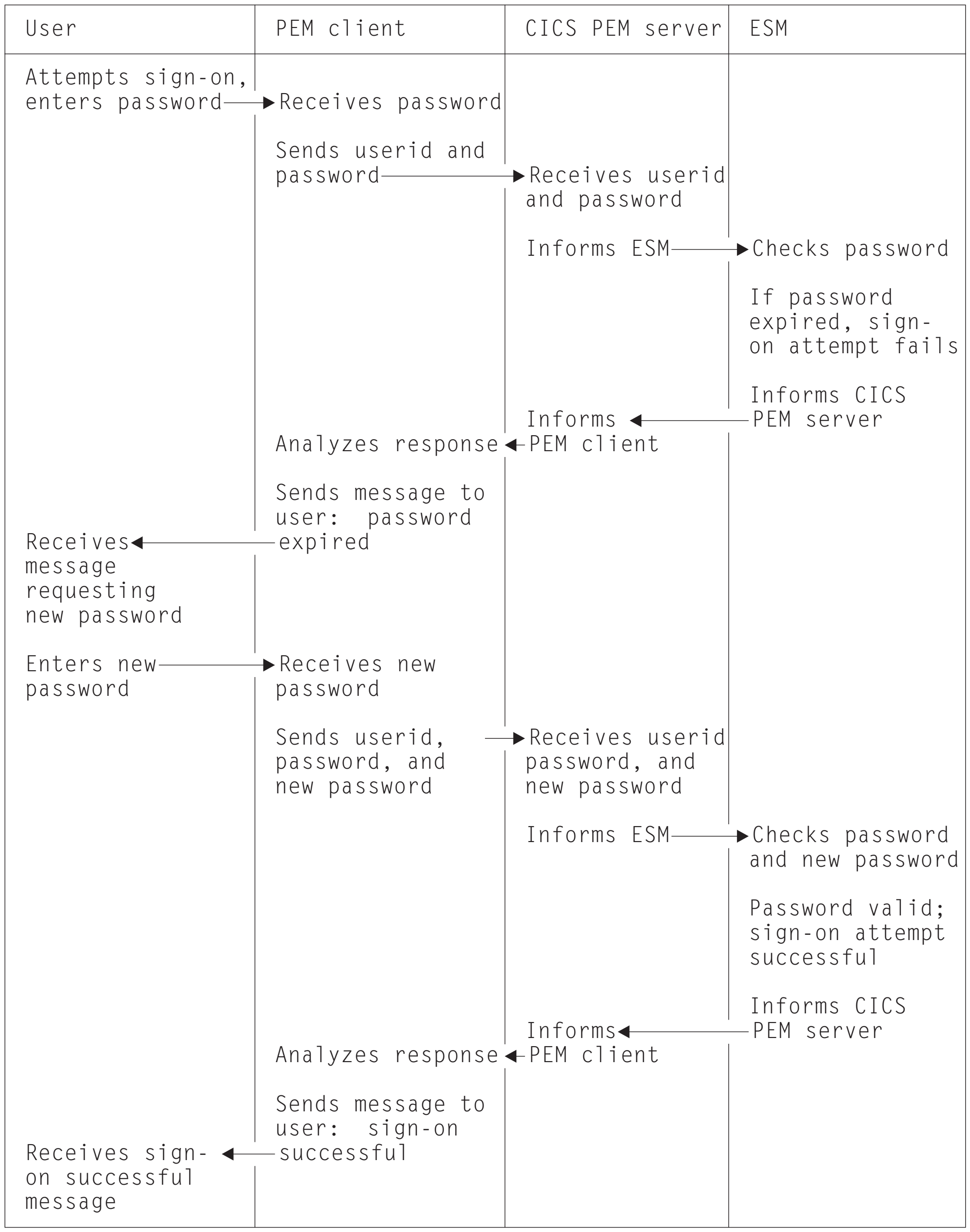

User
PEM client
CICS PEM server
ESM
Attempts sign-on, enters password
Receives password
Sends userid and password
Receives userid and password
Informs ESM
Checks password
If password expired, sign-on attempt fails
Informs CICS PEM server
Informs PEM client
Analyzes response
Sends message to user: password expired
Receives message requesting new password
Enters new password
Receives new password
Sends userid, password, and new password
Receives userid password, and new password
Informs ESM
Checks password and new password
Password valid; sign-on attempt successful
Informs CICS PEM server
Informs PEM client
Analyzes response
Sends message to user: sign-on successful
Receives sign-on successful message

Figure 10. Example of signing on with APPC PEM

Overview of APPC PEM processing

CICS provides the PEM server, the receive side of APPC PEM as a CICS transaction that is started when an ATTACH for the sign-on transaction program is received from the PEM client.

CICS retrieves the sign-on data associated with the request, calls the ESM to perform a sign-on, and retrieves sign-on details for the userid. If the sign-on data includes a new password value, CICS includes this value when it calls the ESM to request a sign-on.

If PV is being used, and sign-on 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 client. The “signed-on-”lists keep track of verified user IDs.

The CICS PEM server builds a reply and returns it to the PEM client, after which the CICS PEM server transaction terminates normally.

PEM client processing

The PEM client sign-on transaction program:

1. Obtains sign-on information, for example by:
   - Displaying a message to the user requesting sign-on 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 sign-on 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 sign-on is successful, 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 sign-on PEM client.
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 sign-on
   - Data and time the password will expire (calculated by data extracted from the ESM by the CICS PEM server itself)
   - Number of unsuccessful sign-on attempts since the last successful sign-on.
3. Returns a response to the PEM client, indicating whether the sign-on was succeeded 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 sign-on
```

```
Expiry-Date-Time = Date and time password will expire
Revoke-Count     = Number of unsuccessful sign-on attempts made with
                   this userid since the previous successful sign-on
```

For detailed information about the response, see “PEM client input and output data” on page 188.

Note: The ESM increments the revoke count whenever it processes an invalid sign-on attempt. The sign-on request may originate from a non-CICS system (for example, a TSO user).

If sign-on is unsuccessful, CICS returns to the PEM client a sign-on completion status value and, if appropriate, a formatting error value. See “PEM client input and output data” on page 188 for more information.

4. If PV is being used (either ATTACHSEC=PERSISTENT or ATTACHSEC=MIXIDPE is specified on the CONNECTION definition), and sign-on is successful, adds the user's name to the PV signed-on-from list.

Expected flows between PEM client and CICS PEM server

Figure 11 on page 183 through Figure 14 on page 186 show expected flows for successful and unsuccessful sign-on attempts with and without PV.

Note: CICS support for the PEM client sign-on transaction assumes that the request for sign-on (or sign-on and change password) is for a single user. Batching of sign-on requests for different userids within a single sign-on 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 sign-on—non-PV connection

Figure 11 on page 183 shows the expected flows between the PEM client and CICS PEM server during a successful sign-on when PV is not being used.

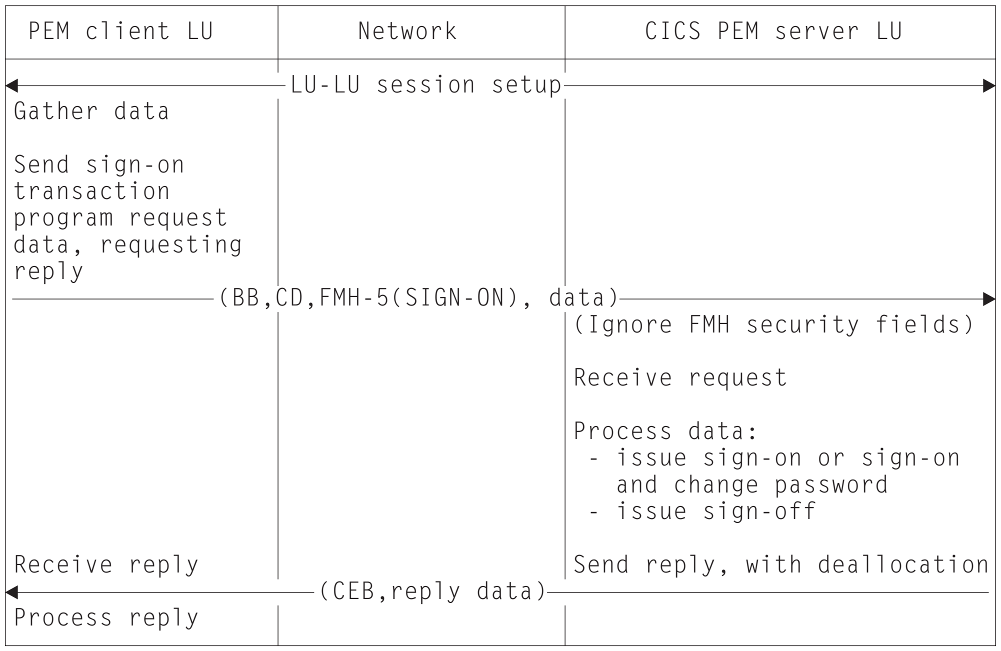

PEM client LU
Network
CICS PEM server LU
LU-LU session setup
Gather data
Send sign-on transaction program request data, requesting reply
(BB,CD,FMH-5(SIGN-ON), data)
(Ignore FMH security fields)
Receive request
Process data:
- issue sign-on or sign-on and change password
- issue sign-off
Receive reply
Send reply, with deallocation
(CEB,reply data)
Process reply

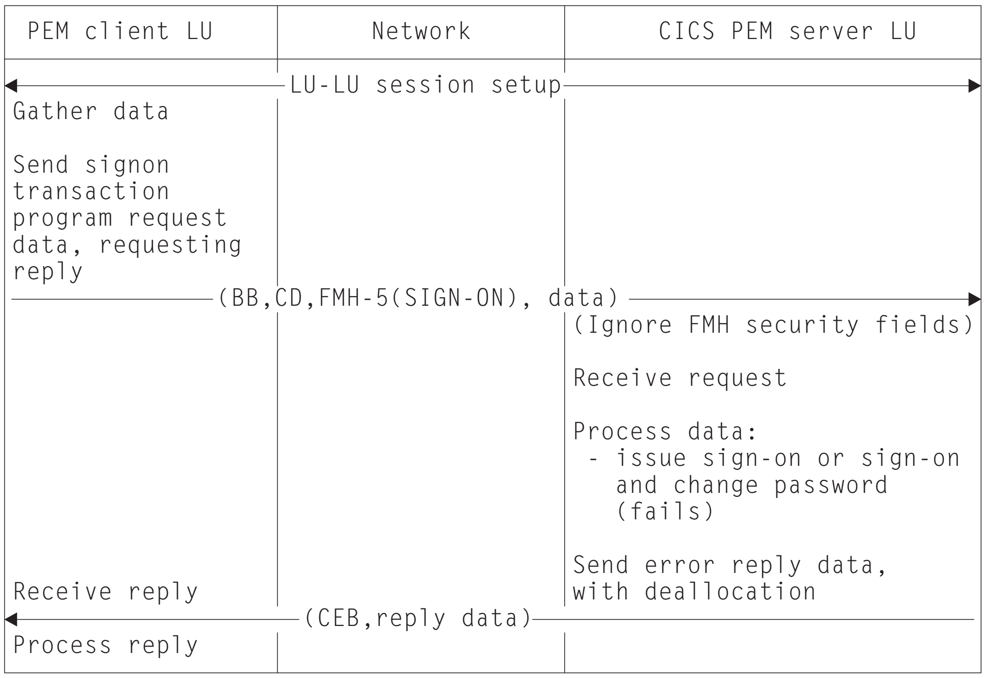
Figure 11. Expected flows for a successful sign-on with a non-PV connection. This is the expected sequence of events:

1. *The session between the PEM client logical unit and the CICS PEM server logical unit is established.*
2. *The PEM client gathers data, and sends data for the sign-on program request to the server, requesting a reply. The request flow contains BB, CD, FMH5, and data.*
3. *The CICS PEM server ignores the security fields in the FMH5. It processes the data to issue the sign on (or sign on with changed password).*
4. *The server sends the reply and deallocates the session. The reply flow contains CEB and data.*
5. *The client receives and processes the reply.*

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 sign-on transaction.

Unsuccessful sign-on—non-PV connection

Figure 12 on page 184 shows the expected flows for an unsuccessful sign-on between a PEM client and CICS PEM server when PV is not being used.



PEM client LU
Network
CICS PEM server LU
LU-LU session setup
Gather data
Send signon transaction program request data, requesting reply
(BB,CD,FMH-5(SIGN-ON), data)
(Ignore FMH security fields)
Receive request
Process data:
- issue sign-on or sign-on and change password (fails)
Send error reply data, with deallocation
Receive reply
(CEB,reply data)
Process reply

Figure 12. Unsuccessful sign-on—non-PV connection. This is the expected sequence of events:

1. *The session between the PEM client logical unit and the CICS PEM server logical unit is established.*
2. *The PEM client gathers data, and sends data for the sign-on program request to the server, requesting a reply. The request flow contains BB, CD, FMH5, and data.*
3. *The CICS PEM server ignores the security fields in the FMH5. It processes the data to issue the sign on (or sign on with changed password). The request fails.*
4. *The server sends the error reply and deallocates the session. The reply flow contains CEB and data.*
5. *The client receives and processes the reply.*

Note: The CICS PEM server schedules sign-off against the PEM client if a sign-on request for a userid fails.

Successful sign-on—PV connection

Figure 13 on page 185 shows the expected flows between the PEM client and CICS PEM server during a successful sign-on on a PV connection.

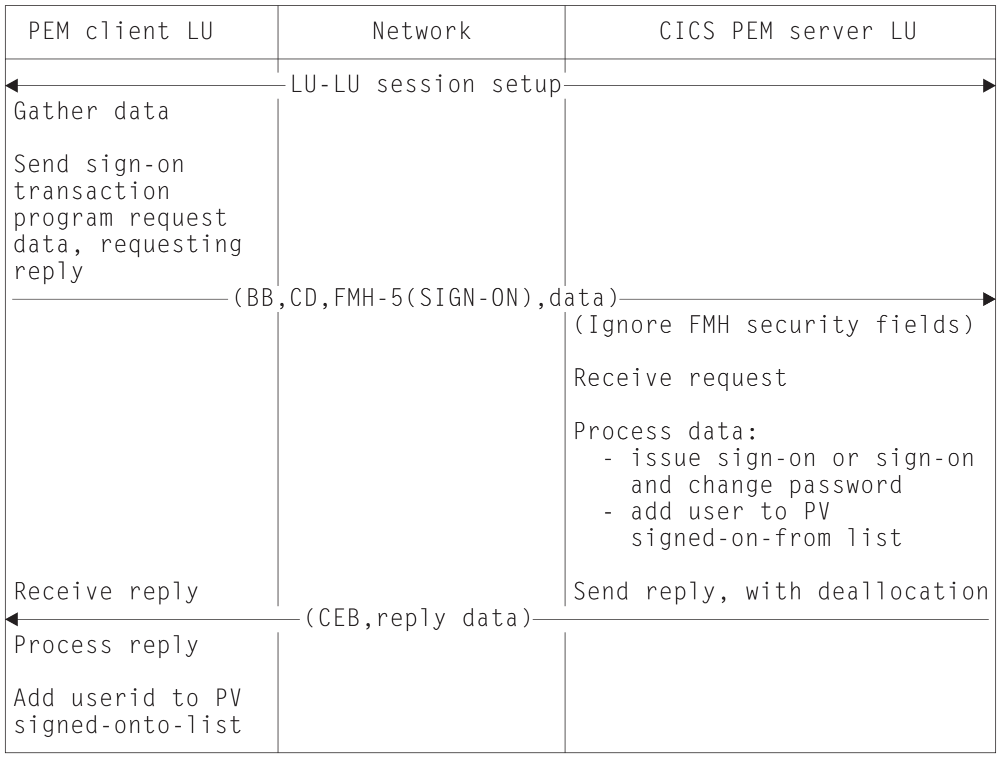

PEM client LU
Network
CICS PEM server LU
LU-LU session setup
Gather data
Send sign-on transaction program request data, requesting reply
(BB,CD,FMH-5(SIGN-ON),data)
(Ignore FMH security fields)
Receive request
Process data:
- issue sign-on or sign-on and change password
- add user to PV signed-on-from list
Receive reply
Send reply, with deallocation
(CEB,reply data)
Process reply
Add userid to PV signed-onto-list

Figure 13. Successful sign-on—PV connection. This is the expected sequence of events:

1. *The session between the PEM client logical unit and the CICS PEM server logical unit is established.*
2. *The PEM client gathers data, and sends data for the sign-on program request to the server, requesting a reply. The request flow contains BB, CD, FMH5, and data.*
3. *The CICS PEM server ignores the security fields in the FMH5. It processes the data to issue the sign on (or sign on with changed password).*
4. *The server adds the user to the PV signed-on-from list.*
5. *The server sends the reply and deallocates the session. The reply flow contains CEB and data.*
6. *The client receives and processes the reply.*
7. *The client adds the userid to the signed-onto list.*

Note:

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 sign-on transaction.
2. The CICS PEM server adds the userid to its PV signed-on-from list only if the sign-on and change password request is successful and either ATTACHSEC=MIXIDPE or ATTACHSEC=PERSISTENT is specified in the CONNECTION definition.
3. The PEM client must add the userid to its PV signed on-to list only if a successful sign-on 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 sign-on—PV connection

Figure 14 shows the expected flows between a PEM client and CICS PEM server during an unsuccessful sign-on on a PV connection.

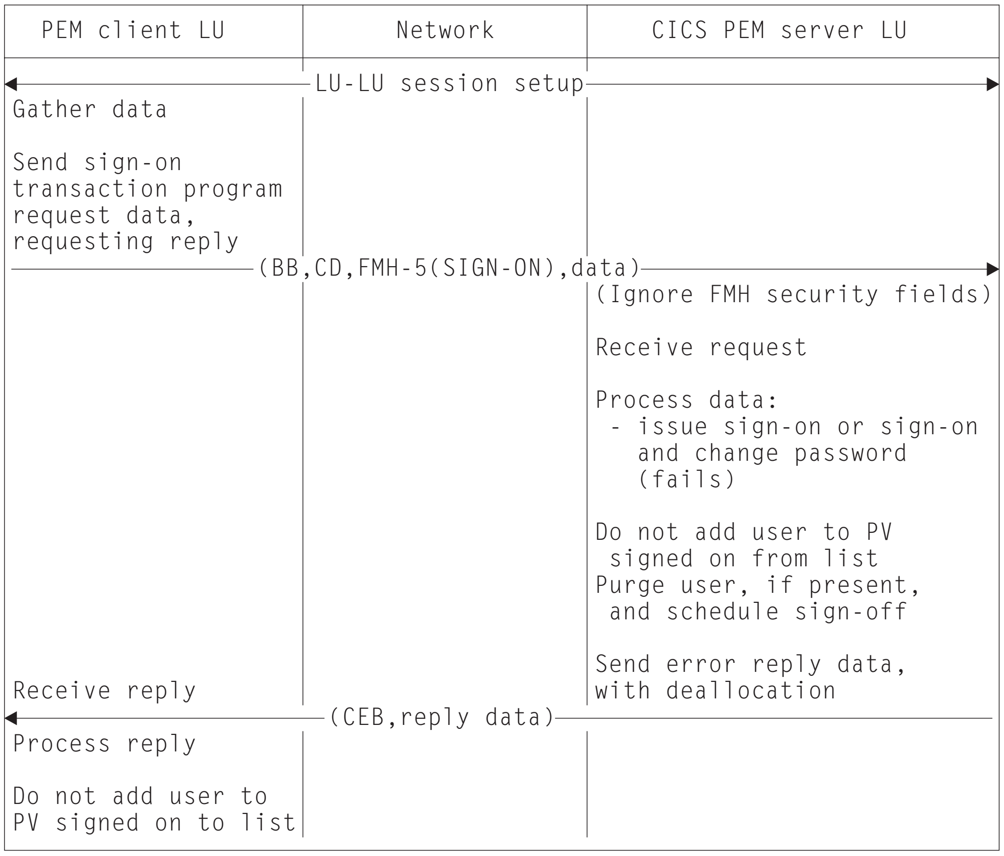

PEM client LU
Network
CICS PEM server LU
LU-LU session setup
Gather data
Send sign-on transaction program request data, requesting reply
(BB,CD,FMH-5(SIGN-ON),data)
(Ignore FMH security fields)
Receive request
Process data:
- issue sign-on or sign-on and change password (fails)
Do not add user to PV signed on from list
Purge user, if present, and schedule sign-off
Send error reply data, with deallocation
Receive reply
(CEB,reply data)
Process reply
Do not add user to PV signed on to list

Figure 14. Unsuccessful sign-on—PV connection. This is the expected sequence of events:

1. *The session between the PEM client logical unit and the CICS PEM server logical unit is established.*
2. *The PEM client gathers data, and sends data for the sign-on program request to the server, requesting a reply. The request flow contains BB, CD, FMH5, and data.*
3. *The CICS PEM server ignores the security fields in the FMH5. It processes the data to issue the sign on (or sign on with changed password). The request fails.*
4. *The server does not add the user to the signed-on-from list. If the user is already on the list, the user is purged from it, and a signoff is scheduled.*
5. *The server sends the error reply and deallocates the session. The reply flow contains CEB and data.*
6. *The client receives and processes the reply.*
7. *The user is not added to the signed-onto list.*

Note: CICS schedules sign-off against the PEM client if a sign-on request for a userid fails, and that user is in the PV signed on from list. In this case, CICS sends the sign-off transaction program output data to the PEM client, where it is processed and the userid removed from the PV signed on to list.

Setting up the PEM client

When setting up your PEM client, note the following:

- Use the basic (also known as **unmapped**) conversation type. In addition to sending the data you want the partner to receive, you must add control bytes (in Assembler language or C) to convert the data into an SNA-defined format called a **generalized data stream** (GDS).
- The SNA service transaction program name for the sign-on transaction program is **X'06F3F0F1'**, which is also the transaction id (XTRANID) that must be used for the CICS transaction CLS4. You specify XTRANID in the CICS TRANSACTION definition.
- Run the CICS PEM server sign-on transaction 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.
- Translate the userid and password into EBCDIC; if they are not in this form, the ESM cannot recognize them and issues an error. See one of the the example programs which is described in "Introduction to APPC password expiration management" on page 177, for an example of converting userids and passwords to EBCDIC.

  If the ESM is RACF, the userid and password must also be in uppercase characters.
- On the ATTACH request for the sign-on transaction program 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 sign-on transaction program. If data identifier (ID) X'00' is provided, CICS returns status value X'06' (incorrect data format) with formatting error X'0002' (precluded structure present). For information about status values and formatting error values, see "PEM client input and output data" on page 188.
- 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', CICS returns status value X'06' (incorrect data format) with formatting error X'0002' (precluded structure present). For information about status values and formatting error values, see "PEM client input and output data" on page 188.
- CICS only supports userids and passwords up to 8 characters long. If the userid or password length (after stripping blanks and nulls) exceeds 8, CICS returns status value X'06' (incorrect data format) with formatting error X'000F' (data value out of range). For information about status values and formatting error values, see "PEM client input and output data" on page 188.
- Program initialization parameter (PIP) data is optional on the ALLOCATE for the sign-on transaction, and is ignored if sent.
- If the sign-on 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. (It does not provide diagnostic information about the type of conversation error.)
- 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, the command 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 each two bytes long), and must follow the attach FMH-5 header. The CICS DFHCLS4 program requires the user input data stream to fit into the format shown in Figure 15; if it does not, CICS rejects the data.

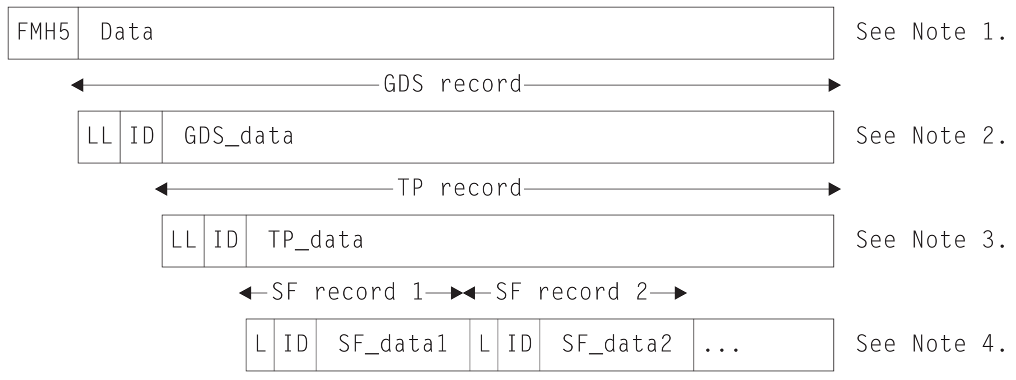


Figure 15. Format of user data.

Note:

1. This is an attach FMH-5 header with its data. Data is passed between the PEM client and the CICS PEM server via GDS variables. (For information on GDS, see the *SNA LU 6.2 Peer Protocols* manual.)
2. The **GDS record** contains GDS data in the format LL-ID-data 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' (sign-on 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' (sign-on) 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 client input and output data

To perform the functions described in “CICS PEM server processing” on page 181, the CICS PEM server takes input data from, and sends output data to, the PEM client sign-on transaction program:

- The PEM client sends data to the CICS PEM server, as described in Table 23 on page 189.
- The CICS PEM server sends data to the PEM client, as described in Table 24 on page 189 through Table 27 on page 192.

Ensure the data conforms to the standards described in “Setting up the PEM client” on page 186, and that its format is as described in “Format of user data” on page 187. See “Sign-on with correct userid and password” on page 193 and “Sign-on with new password” on page 194 for examples of sign-on output data in GDS flows.

Basic conversation information and data are contained in the attach FMH, as described in “Format of user data” on page 187. The sign on request attaches a transaction X'06F3F0F1', which is the SNA service transaction program name for the sign-on transaction program.

Sign-on input data sent by PEM client

Table 23 shows the input data that the CICS PEM server needs from the PEM client sign-on transaction program. See “Sign-on with correct userid and password” on page 193 and “Sign-on with new password” on page 194 for examples of sign-on input data in GDS flows.

Table 23. Sign-on request and data sent to CICS PEM server

Length (bytes)	Value	Meaning
2	X'nnnn'	Length of entire GDS data, including this 2-byte length value.
2	X'1221'	Data ID for sign-on 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 sign-on or sign-on 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.
X'nn'-2	C'xxxxxxxx'	Userid.
1	X'mm'	Length of subfield for password, including this 1-byte length value.
1	X'02'	Data ID of subfield for password.
X'mm'-2	C'xxxxxxxx'	Password.
1	X'pp'	Length of subfield for new password, including this 1-byte length value.
1	X'06'	Data ID of subfield for new password.
X'pp'-2	C'xxxxxxxx'	New password.

Sign-on output data returned by CICS PEM server

Table 24 lists the sign-on output data that the CICS PEM server returns to the PEM client. See “Response to correct sign-on data” on page 194 and “Response to incorrect data format” on page 196 for examples of sign-on output data in GDS flows.

Table 24. Sign-on output data returned to PEM client

Length (bytes)	Value	Required or optional	Meaning
2	X'nnnn'	Required	Length of entire GDS data, including this 2-byte length value.
2	X'1221'	Required	Data ID of subfield for sign-on data.

Table 24. Sign-on output data returned to PEM client (continued)

Length (bytes)	Value	Required or optional	Meaning
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 sign-on reply data.
1	X'03'	Required	Length of subfield for sign-on completion status, including this 1-byte length value.
1	X'00'	Required	Data ID of subfield for sign-on completion status.
1	X'00' through X'06'	Required	Sign-on completion status—see Table 26 on page 192.
1	X'04'	Optional	Length of subfield for sign-on request formatting error, including this 1-byte length value.
1	X'01'	Optional	Data ID of subfield for sign-on request formatting error.
2	X'0000' through X'0003', X'0005' through X'0007', X'000F'	Optional	Sign-on request formatting error—see Table 27 on page 192.
1	X'0A'	Optional	Length of subfield for date and time of current successful sign-on, including this 1-byte length value.
1	X'02'	Optional	Data ID of subfield for date and time of current successful sign-on.
8	See Table 25 on page 191 for format	Optional	Date and time of current successful sign-on. 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 sign-on, including this 1-byte length value.
1	X'03'	Optional	Data ID of subfield for date and time of last successful sign-on.
8	See Table 25 on page 191 for format	Optional	Date and time of last successful sign-on. 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.

Table 24. Sign-on output data returned to PEM client (continued)

Length (bytes)	Value	Required or optional	Meaning
1	X'04'	Optional	Data ID of subfield for date and time password will expire.
8	See Table 25 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 25 lists the format of the date and time subfields that the CICS PEM server can return to the PEM client, as referred to in Table 24 on page 189. See "Response to correct sign-on data" on page 194 for an example of date and time subfields in a GDS flow.

Table 25. Format of date and time subfields using 24-hour clock

Position	Meaning
00	Two-byte year value; for example, 1994=X'07CB'.
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.

Sign-on completion status values returned to PEM client

Table 26 on page 192 describes the sign-on completion status values (referred to in Table 24 on page 189) that the CICS PEM server can return to the PEM client in the status completion subfield in the sign-on reply data. See "Response to correct sign-on data" on page 194 for an example of sign-on completion status values in a GDS flow.

Table 26. Sign-on completion status values returned to PEM client

Status value	Meaning
X'00'	All of the following conditions apply: • Userid valid • Password valid • Password not expired or new valid password specified 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 sign-on request formatting error subfield described in Table 27.

Note: The CICS PEM server never returns either of the following sign-on status values to the PEM client:
- X'07'—general security error
- X'08'—password change completed, but sign-on failed.

Sign-on request formatting errors returned to PEM client

Table 27 lists the sign-on request formatting error values (referred to in Table 24 on page 189) that the CICS PEM server can return to the PEM client. Each is a 2-byte binary value. See "Response to incorrect data format" on page 196 for an example of sign-on request formatting errors in a GDS flow.

Table 27. Sign-on request formatting error values returned to PEM client

Error value	Description
X'0000'	Undefined error not described below.
X'0001'	Required structure absent.
X'0002'	Precluded structure present.
X'0003'	Several occurrences of a nonrepeatable structure.
X'0005'	Unrecognized structure present where precluded.
X'0006'	Length outside specified range. This value assumes that the 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

Design your applications to run the sign-on transaction before any other transaction. This keeps that any password check and any password changing separate from the application's own functions. In multitasking systems, it is possible for more than one sign-on transaction to start on parallel sessions. It is important that the code dealing with application-level ALLOCATE requests, serializes the sign-on process to completion, thus ensuring both flow as signed-on.

To record the date and time of a previous successful sign-on, the CICS PEM server sign-on program extracts password data from the ESM **before** it performs sign-on. 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 client for the current sign-on may not be the same as the timestamp recorded on the ESM database. Remember this if you are writing an application that is to run on multiple systems, and depends on the sign-on time returned to the PEM client. (This situation should not apply on a single system, provided the sign-on 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 sign-on process to be serialized again.

Examples of PEM client and CICS PEM server user data

Data is passed between the PEM client and the CICS PEM server via GDS variables. To help you check the data being sent by your PEM client, the examples that follow show:

- "Sign-on with correct userid and password"
- "Sign-on with new password" on page 194
- "Response to correct sign-on data" on page 194
- "Response to incorrect data format" on page 196.

These examples are produced by the sample PEM client program shown in the library *CICSTS32.CICS.SDFHSAMP*, described in "Introduction to APPC password expiration management" on page 177. That program uses a **partner_LU_alias** of `hostcics`, an **LU_alias** of `ps2lua`, and a **mode_name** of `lu62ss`. When writing your own PEM client program, use the values defined in your communications manager configuration.

Sign-on with correct userid and password

Figure 16 shows a sample flow for a successful sign-on.

```
PEM hostcics ps2lua lu62ss sec2r01 drtnnom
```

Figure 16. Sign-on with correct userid and password, no new password

A valid userid (`sec2r01`) and password (`drtnnom`) are correctly entered. No new password is entered.

The PEM client sends the following hexadecimal user data to the CICS PEM server:

```
001A12210016FF000901E2C5C3F2D9F0F10902C4D9E3D5D5D6D4
```

This contains the following values, as described in Table 23 on page 189:

001A	Length of the entire GDS data, 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 sign-on 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

Sign-on with new password

The following is an example of a successful sign-on using a new password.

```
PEM hostcics ps2lua lu62ss sec2r01 drtnnom hursley
```

A userid, password, and new password are correctly entered.

The PEM client sends the following hexadecimal user data to the CICS PEM server:

```
0231221001FFF010901E2C5C3F2D9F0F10902C4D9E3D5D5D6D40906C8E4D9E2D3C5E8
```

This contains the following values, as described in Table 23 on page 189:

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 sign-on 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 sign-on data

Figure 17 on page 195 shows an example of the response to the correct sign-on data being entered.

```
PEM_OK
GDS LLID
00 2d 12 21
Sign-on Reply LLID
00 29 ff 02
Sign-on Completion Status Subfield
03 00 00
Date & Time of Current Successful Sign-on Subfield
0a 02 07 ca 01 14 0d 24 31 62
Date & Time of Last Successful Sign-on 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 17. Response to correct sign-on data

The first three lines of hexadecimal user data returned to the PEM client show the following *required* values, as described in Table 24 on page 189.

002d Total length of the GDS variable, including this 2-byte length value

1221 Data ID for sign-on data

0029 Length of this second (nested) data structure (length, data ID, and data), including this 2-byte length value

FF02 Data ID for sign-on reply data

03 Length of subfield for sign-on completion status, including this 1-byte length value

00 Data ID for sign-on completion status

00 Sign-on completion status. 00 indicates that the userid and password were valid, and the password had not expired. (See Table 26 on page 192 for a list of sign-on completion status values.)

In Figure 17, the last four lines of hexadecimal user data returned to the PEM client show the following optional values, as described in Table 24 on page 189. (Note that the formatting error subfields shown in Table 24 on page 189 are not included, indicating that there are no errors.)

0A Length of subfield for date and time of current successful sign-on including this 1-byte length value

02 Data ID for date and time of current successful sign-on

Date and time of current successful sign-on, as described in Table 25 on page 191:

07CA	Year (1994)
01	Month (January)
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 sign-on,

03	Data ID for date and time of previous successful sign-on

Date and time of previous successful sign-on, as described in Table 25 on page 191:

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	Length of subfield for 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 25 on page 191:

07ca	Year (1994)
02	Month (February)
03	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 sign-on attempts since the last successful sign-on with this userid.)

Response to incorrect data format

Figure 18 shows an example response to incorrect data being entered.

```
PEM_OK
GDS LLID
00 0F 12 21
Sign-on Reply LLID
00 0B FF 02
Sign-on Completion Status Subfield
03 00 06
Sign-on Request Formatting Error Subfield
04 01 00 0F
```

Figure 18. Response to incorrect data format

The first three lines of hexadecimal user data returned to the PEM client show the following required values, as described in Table 24 on page 189:

000F	Length of entire GDS data, including this 2-byte length value
1221	Data ID for sign-on data

`000B`	Length of this second (nested) data structure (length, data ID, and data), including this 2-byte length value
`FF02`	Data ID for sign-on reply data
`03`	Length of subfield for sign-on completion status, including this 1-byte length value
`00`	Data ID of subfield for sign-on completion status
`06`	Sign-on completion status 06 indicating incorrect data format (see Table 26 on page 192 for a list of signon completion status values.)

The last line of hexadecimal user data returned to the PEM client shows the following **optional** values, which are returned only if there is an error. (The optional values are described in Table 24 on page 189.)

`04`	Length of subfield for sign-on request formatting error, including this 1-byte length value
`01`	Data ID of subfield for sign-on request formatting error
`000F`	Sign-on request formatting error, indicating “data value out of range” (see Table 27 on page 192 for a description of other possible formatting errors).

Chapter 14. Implementing LU6.1 security

This topic tells you how to implement link security for LU6.1.

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 link userid matches the local region's userid:

- If the userids match, the resource check is made against the local region's default user.
- If the userids do not match, the resource check is carried out against the link userid.

If a failure occurs in establishing link security, the link is given the security of the local region's default user. This would happen if, for example, the preset session userid had been revoked.

Specifying link security for LU6.1 connections

- To specify that all sessions of a connection should have the same link userid, specify the SECURITYNAME attribute of the CONNECTION definition. If you do not specify a value for this attribute, CICS uses the default user ID.
- To specify different link userids for individual groups of sessions within a connection, specify the USERID attribute of the SESSIONS definition. For each group of sessions, the value specified overrides the SECURITYNAME attribute of the CONNECTION definition.

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 specify only ATTACHSEC(LOCAL) in the CONNECTION definition. Figure 19 on page 200 shows an example of doing this using CEDA.

```
CEDA DEFINE CONNECTION(name)
  GROUP(groupname)
  .
  ATTACHSEC(LOCAL)
```

Figure 19. Defining sign-on 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 81.

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.

CICS performs resource and command security checking only if the installed TRANSACTION definition specifies that they are required; for example, on the CEDA DEFINE TRANSACTION command, as shown in Figure 20.

```
CEDA DEFINE TRANSACTION
  .
  RESSEC(YES)
  CMDSEC(YES)
  .
```

Figure 20. 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 87 and Chapter 8, “CICS command security,” on page 119.

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 143.

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 21 on page 202 summarizes what happens.

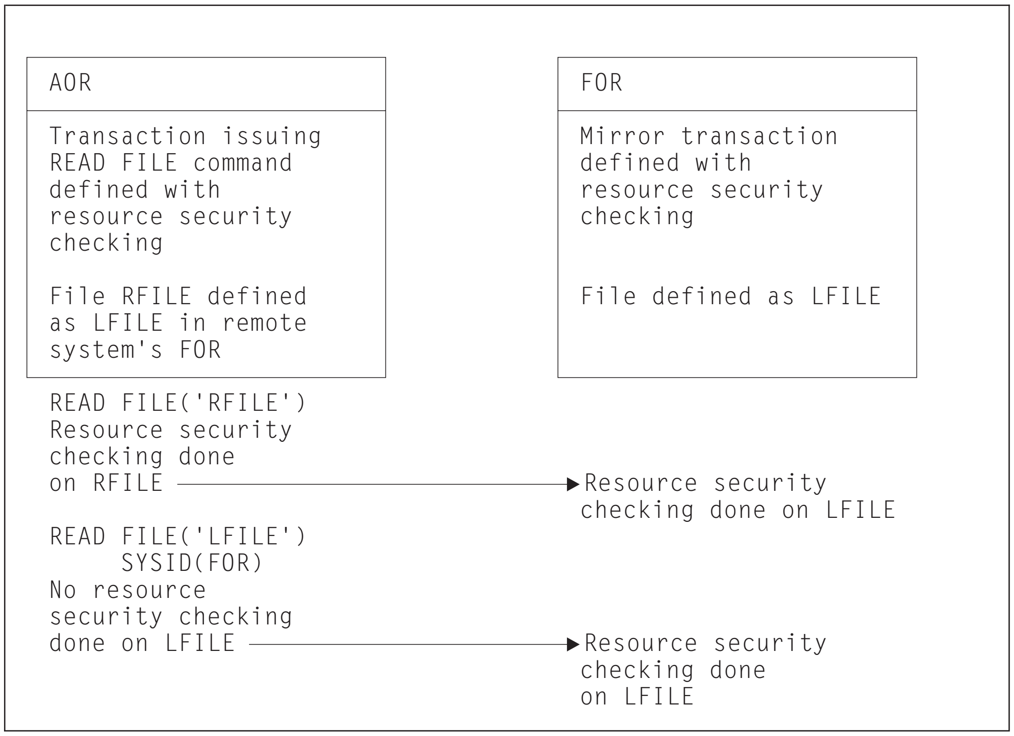

AOR
Transaction issuing READ FILE command defined with resource security checking
File RFILE defined as LFILE in remote system's FOR
FOR
Mirror transaction defined with resource security checking
File defined as LFILE
READ FILE('RFILE') Resource security checking done on RFILE
Resource security checking done on LFILE
READ FILE('LFILE') SYSID(FOR) No resource security checking done on LFILE
Resource security checking done on LFILE

Figure 21. Security checking done with and without SYSID. This example illustrates what security checking is done when a transaction in an application-owning region issues a file control request against a remote file.

- *In the application-owning region (AOR), file RFILE is defined as remote, with a name of LFILE in the file-owning region (FOR). Resource security checking is active for the transaction that issues the file control request.*
- *In the FOR, resource security checking is active for the mirror transaction.*

There are two cases. In the first case:

1. *The transaction in the AOR issues EXEC CICS READ FILE('RFILE'). Resource security checking is performed for file RFILE.*
2. *The request is transmitted to the FOR, where resource security checking is performed for file LFILE.*

In the second case:

The transaction in the AOR issues EXEC CICS READ FILE('RFILE') specifying the SYSID option. Resource security checking is not performed for file RFILE.

The request is transmitted to the FOR, where resource security checking is performed for file LFILE.

For programming information on specifying the SYSID option, see the the *CICS Application Programming Reference*.

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 28 on page 203 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 28 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, is relevant to security checking in the AOR.

Table 28. 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

Chapter 15. Implementing IPIC security

This section tells you how to implement security for IPIC connections.

Related information

Defining IP interconnectivity links

IPIC bind-time security

A security check can be applied when a request to establish a connection is received from, or sent to, a remote system. This is called *bind-time security*. Its purpose is to prevent an unauthorized system from connecting to CICS.

When CICS uses IPIC to communicate with another CICS region, each CICS system must have an IPCONN resource and a TCPIPSERVICE resource defined.

Each CICS system uses the IPCONN to transmit data to the partner system TCPIPSERVICE, which acts as a receiver. The CICS region that starts the communication is the client, the remote system is the server.

For IPIC, bind security is supported by the exchange of Secure Sockets Layer (SSL) client certificates. To allow two CICS regions to connect successfully, and to prevent an unauthorized system from connecting:

- The SEC system initialization parameter must be “YES” on both regions.
- The IPCONN definitions on both the local and remote regions must specify:
  - SSL(YES).
  - CIPHERS(*cipher_suite_code_list*). This is a string of up to 56 hexadecimal digits that is interpreted as a list of up to 28 2-digit cipher suite codes. When you use CEDA to define the resource, CICS automatically initializes the attribute with a default list of acceptable codes, depending on the level of encryption that is specified by the ENCRYPTION system initialization parameter.
  - Optionally, CERTIFICATE(*X.509_certificate_label*). The named certificate is used as the client certificate, during the SSL handshake when the IPCONN is acquired. If CERTIFICATE is not specified, the default certificate defined in the key ring for the CICS region user ID is used.

  The IPCONN defines the *outbound* side of the connection: these settings tell CICS to initiate an SSL handshake. During the SSL handshake, CICS will ask the partner region for the certificate specified on the TCPIPSERVICE. If the remote region TCPIPSERVICE specifies SSL(CLIENTAUTH), the remote system requests the certificate of the originating system as part of the handshake.
- The TCPIPSERVICE definitions on both the local and remote regions specify:
  - PROTOCOL(IPIC).
  - SSL(CLIENTAUTH) or SSL(YES).
  - CIPHERS(*cipher_suite_code_list*).
  - Optionally, CERTIFICATE(*X.509_certificate_label*). The named certificate is used as the server certificate. If CERTIFICATE is not specified, the default certificate defined in the key ring for the CICS region user ID is used.

The TCPIPSERVICE definitions define the *inbound* side of the connection: these settings tell CICS that it must receive a valid SSL client certificate before allowing the client to acquire the IPCONN. These settings also specify that CICS will send the TCPIPSERVICE CERTIFICATE, or the default, when not specified, as a server certificate to the client.

If the TCPIPSERVICE is specified with SSL(YES), the server does not ask for, nor receive, a client certificate during the handshake.

If the TCPIPSERVICEs in both CICS regions are specified with SSL(YES), both CICS regions are authenticated.

If the TCPIPSERVICEs in both CICS regions are specified with SSL(CLIENTAUTH), both CICS regions are authenticated twice.

For most circumstances, an adequate level of security is achieved by specifying TCPIPSERVICE with SSL(YES) in both regions, or SSL(NO) in one region and SSL(CLIENTAUTH) in the other.

Note: If LINKAUTH is specified CERTUSER, the IPCONN must refer to a TCPIPSERVICE that is defined with SSL(CLIENTAUTH).

When the TCPIPSERVICE is specified SSL(NO) on both regions, bind-time security is not possible

If the remote client is trusted by the CICS server, bind time security is not required, however, any user ID and password passed for transaction attach must be valid in the server region's external security manager.

IPIC link security

Link security 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 user ID has no authority.

When link security is in use, all requests are given an authority defined by the *link user ID*. For IPCONNs, all requests for a connection have the same link user ID.

Specifying IPIC link security

To specify the link user ID of an IPCONN, use the LINKAUTH option to specify how the user ID for link security is established in a CICS system with security initialized (SEC=YES). You can specify the following:

CERTUSER
TCP/IP communication with the partner system must be configured for SSL and a certificate must be received from the partner system during SSL handshake.

The IPCONN must refer to a TCPIPSERVICE that is defined with SSL(CLIENTAUTH).

The received certificate must be defined to the external security manager so that it is associated with a user ID, which is used to establish link security.

SECUSER
Specifies that the user ID specified in SECURITYNAME is used to establish link security.

If you do not specify a value for SECURITYNAME, CICS uses the default user ID.

In a CICS system with security initialized (SEC=YES), the link user ID is used to establish the authority of the remote system. The link user ID must be a valid RACF user ID on this region. Access to protected resources on this region is based on the RACF user profile and its group membership.

At least one security check is always performed when a transaction is attached by a remote user, but the security checks are minimized if the specified link user ID matches the local region user ID:

If the user IDs match, only one security check is made. This is either against the default user (for USERAUTH=LOCAL or DEFAULTUSER) or against the user ID that is in the received inbound attach request (USERAUTH=IDENTIFY or VERIFY).

If the user IDs do not match, then for USERAUTH=LOCAL, resource checks are done only against the link user ID. For USERAUTH=IDENTIFY or VERIFY there are always two resource checks. The first against the link user ID, and the second is against the user ID received from the remote user in the attach request. For USERAUTH=DEFAULTUSER there are always two resource checks. The first against the link userid, and the second is against the default user.

See also “IPIC user security.”

If a failure occurs in establishing link security, the link is given the security of the local region's default user. This can happen, for example, when the link user ID has been revoked.

IPIC user security

User security causes a second check to be made against a user signed onto a terminal.

This check is in addition to the link security described in “IPIC link security” on page 206.

For IPIC connections, the level of user security is specified for inbound requests only, on the USERAUTH attribute of the IPCONN resource definition. For IPIC connections, you can specify the following types of user authentication:

LOCAL
CICS does not accept a user ID or password from clients. All requests run under the link user ID, or the default user ID if there is no link user ID.

IDENTIFY
Incoming attach requests must specify a user identifier. Enter IDENTIFY when the connecting system has a security manager; for example, if it is another CICS system.

| Client authentication must be in use or the connecting system must be in
| the same sysplex. For more information see the *CICS RACF Security Guide*

VERIFY
Incoming attach requests must specify a user identifier and a user password. Specify VERIFY when connecting systems are unidentified and cannot be trusted.

DEFAULTUSER
CICS does not accept a user ID and password from the partner system. All requests run under the default user ID.

For outbound requests, the level of user security is specified by the USERAUTH attribute of the IPCONN definition installed in the partner system. CICS sends a user ID when USERAUTH(IDENTIFY) is specified, but not when USERID(LOCAL) is specified. Because CICS does not send passwords to remote systems, USERAUTH(VERIFY) is not supported for communication between CICS TS for z/OS systems.

Establishing a trust relationship when using USERAUTH(IDENTIFY)

Defining an IPCONN resource with USERAUTH(IDENTIFY) means that you are prepared to accept unauthenticated, or asserted, user identifiers on the connection; that is, you are trusting the partner system to transmit only user IDs that have already been authenticated in the partner system. CICS identifies two types of partner, with differing degrees of trust:

The partner is outside the local sysplex
If the partner is outside the local sysplex, CICS does not directly trust the partner to assert unauthenticated user IDs unless it identifies itself with a trusted digital certificate. Connections from such partners are therefore only accepted using a secure socket layer connection with client authentication. This type of connection is specified by the attribute SSL(CLIENTAUTH) on the TCPIPSERVICE definition.

If you do not require encryption over the connection, you can suppress the normal encryption that SSL provides by specifying only non-encrypting cipher suites on the attribute CIPHERS(0102) of the TCPIPSERVICE definition.

The partner is in the local sysplex
If the partner is in the local sysplex, CICS trusts the partner to assert unauthenticated user IDs without requiring a digital certificate. Connections from these partners are permitted without requiring an SSL connection.

Configure the TCP/IP network so that there is no proxy between CICS and the partner system. If there is a proxy, CICS might not be able to correctly detect if a partner is in the same sysplex. It is strongly recommended that you configure the netaccess security zones to control which other systems within the same sysplex are allowed to communicate with CICS.

For further information on network access control, see the *z/OS Communication Server: IP Configuration Guide*. For further information on the `NETACCESS` configuration parameter, see the *z/OS Communication Server: IP Configuration Reference*.

IPIC transaction, resource, and command security

Users must be authorized to attach transactions, and access resources that these transactions use.

As in a single-system environment, users must be authorized to:
- Attach a transaction (**transaction security**)
- Access all the resources that the transaction is programmed to use. These levels are called **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 81.

In an IP environment, two basic security requirements must be met before a transaction can be initiated:

- The link userid must have sufficient authority to initiate the transaction (see “IPIC link security” on page 206).
- If anything other than USERAUTH(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.

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 22.

```
CEDA DEFINE TRANSACTION
  .
  RESSEC(YES)
  CMDSEC(YES)
  .
```

Figure 22. 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 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 have sufficient authority for any of the system programming commands shown in Table 10 on page 119 that the attached transaction issues.

For further guidance on specifying resource and command security, see Chapter 6, “Resource security,” on page 87 and Chapter 8, “CICS command security,” on page 119.

Chapter 16. Implementing MRO security

This topic tells you how to implement CICS multiregion operation (MRO) security.

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 the *CICS Intercommunication Guide*.

Bind-time security with MRO

The CICS interregion communication (IRC) facility supports 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 or not 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.

Logon security checking with MRO

Logon security checking is performed whenever a CICS region logs on to the CICS-supplied interregion communication (IRC) program, 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 uses the IRC access method 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.

When a batch job connects to a CICS region using IRC, the CICS region logs on to IRC, and requires UPDATE access to its own DFHAPPL *applid* profile as described above.

See Figure 23 on page 213 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 Transaction Server for z/OS, Version 3 Release 2 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 not used for MRO 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 23 on page 213.

You cannot specify to CICS whether or not you want connect security checking for MRO connections—CICS always issues the RACROUTE calls.

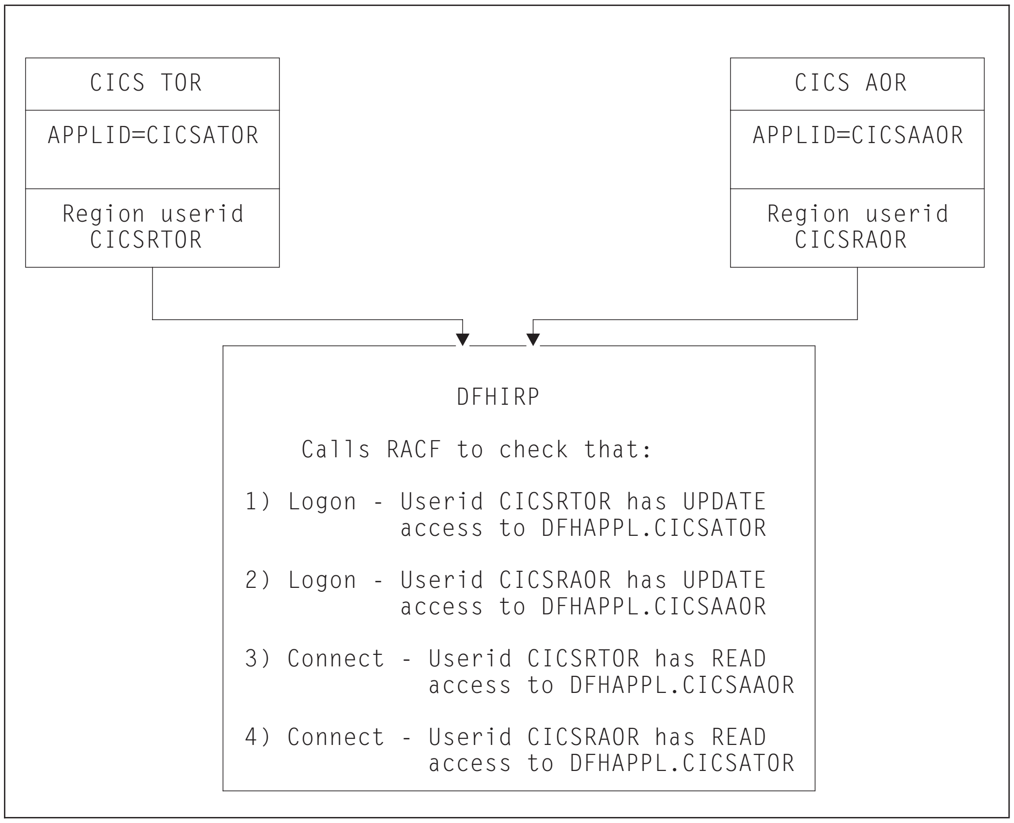

CICS TOR
APPLID=CICSATOR
Region userid CICSRTOR
CICS AOR
APPLID=CICSAAOR
Region userid CICSRAOR
DFHIRP
Calls RACF to check that:
1) Logon - Userid CICSRTOR has UPDATE access to DFHAPPL.CICSATOR
2) Logon - Userid CICSRAOR has UPDATE access to DFHAPPL.CICSAAOR
3) Connect - Userid CICSRTOR has READ access to DFHAPPL.CICSAAOR
4) Connect - Userid CICSRAOR has READ access to DFHAPPL.CICSATOR

Figure 23. Illustration of the DFHIRP logon and connect security checks.

This illustration shows a CICS terminal-owning region (TOR) and CICS application-owning region (AOR) logging on to DFHIRP, and connecting to one another:

- *The TOR has an APPLID of CICSATOR and a region user ID of CICSRTOR.*
- *The AOR has an APPLID of CICSAAOR and a region user ID of CICSRAOR.*

During the logon process, DFHIRP calls RACF to check that:

1. *User ID CICSRTOR has UPDATE access to DFHAPPL.CICSATOR*
2. *User ID CICSRAOR has UPDATE access to DFHAPPL.CICSAAOR*

During the connection process, DFHIRP calls RACF to check that:

1. *User ID CICSRTOR has READ access to DFHAPPL.CICSAAOR*
2. *User ID CICSRAOR has READ access to DFHAPPL.CICSATOR*

The TOR and AOR shown in Figure 23, running under region userids CICSRTOR and CICSRAOR respectively, with APPLIDs CICSATOR and CICSAAOR, require the following RACF definitions to authorize their logon to DFHIRP:

- For 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)
```

- For 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:

IRC rejects the logon or connect request if:

- A security manager was 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.

IRC 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 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)
```

ensures 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 have only one sessions definition per connection, and there can be only 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 link userid matches the local region's userid.

- If the userids match, you will always only have one security check. This will be made either against the local region's default user (for ATTACHSEC=LOCAL) or against the userid in the received FMH-5 attach request (ATTACHSEC=IDENTIFY).
- If the userids do not match, then 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.

Associate the SESSIONS definition with a RACF user profile that has access to any protected resource to which the inbound transaction needs access. See Chapter 2, “RACF facilities,” on page 13 for guidance on defining profiles.

If the sign-on fails, a sign-on 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.

Specifying link security for MRO connections

For MRO connections, all sessions have the same link userid. Specify the link userid in the USERID attribute of the SESSIONS definition. The SECURITYNAME field on the CONNECTION definition is ignored for MRO connections.

User security with MRO

User security causes CICS to make a second check against a user signed on to a terminal, in addition to the link security check described in “Link security with MRO” on page 214. You should consider whether you want the extra level of security checking that user security provides.

You can specify either LOCAL, in which case the user is not checked, or IDENTIFY, in which case a userid is required, but no password is sent.

You specify the sign-on 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 attribute of the CONNECTION definition.

CICS interprets the parameters of the ATTACHSEC attribute as described here. However, special rules apply for CICS transaction routing using CRTE, as described in “Transaction routing security with MRO” on page 218.

The ATTACHSEC attribute specifies the sign-on 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 values of the ATTACHSEC attribute 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.

  Specify ATTACHSEC(IDENTIFY) when you know that CICS can trust the remote system to verify its users, when, for example, the remote system is another CICS.

  The following rules apply to IDENTIFY:

  - If a password is included in an attach request with a user identifier on a link with ATTACHSEC(IDENTIFY), CICS rejects the attach request and unbinds the session.
  - If a null 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 "The CICS default user ID" on page 7.

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.

Sign-on 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 sign-on until either of the following occurs:

- The period specified in the USRDELAY system initialization parameter 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 System Definition 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.

CICS sends userids on ATTACHSEC(IDENTIFY) conversations. Table 29 shows how CICS decides which userid to send.

Table 29. 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 transient data destination definition 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, 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 81.

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 24.

```
CEDA DEFINE TRANSACTION
  .
  RESSEC(YES)
  CMDSEC(YES)
  .
```

Figure 24. 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 10 on page 119) that the attached transaction issues.

For further guidance on specifying resource and command security, see Chapter 6, “Resource security,” on page 87 and Chapter 8, “CICS command security,” on page 119.

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 *CICS 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 214.

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, 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, CICS creates a surrogate terminal 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.

Ensure that the terminal through which CRTE is invoked is defined on the remote system (or defined as “shippable” in the local system). The terminal operator needs RACF authority if the remote system is protected.

Security checking done in the AOR for transactions executed under CRTE does not depend on what is specified on ATTACHSEC, nor on the userid signed on in the TOR. 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 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 does 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 143.

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 25 on page 221 summarizes what happens.

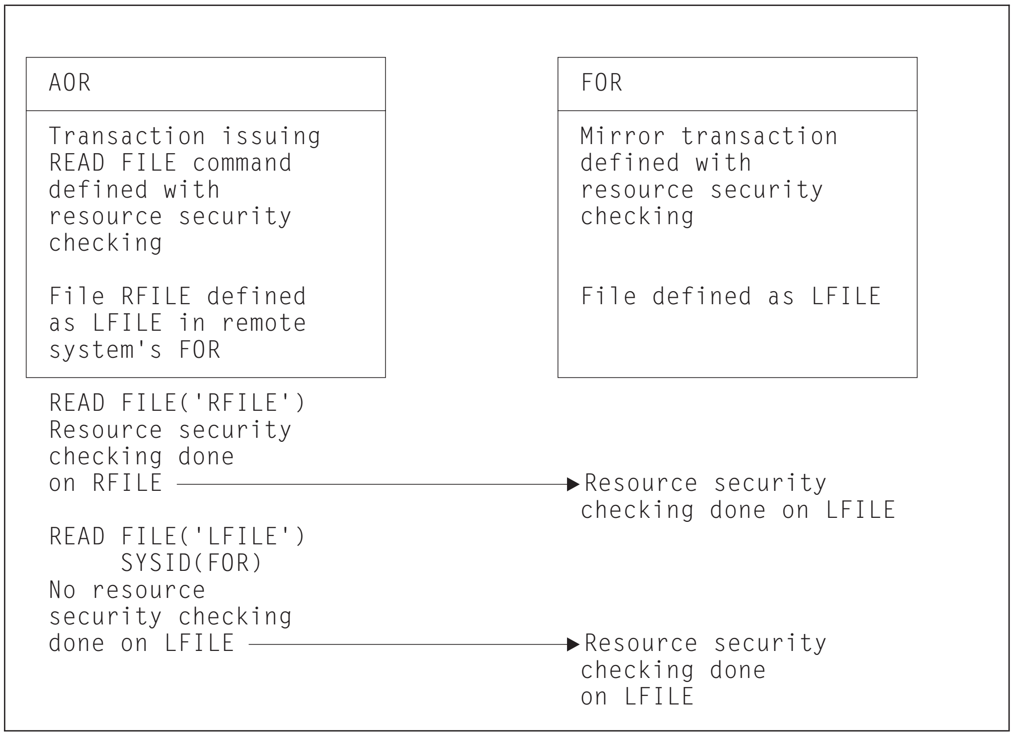

AOR
Transaction issuing READ FILE command defined with resource security checking
File RFILE defined as LFILE in remote system's FOR
FOR
Mirror transaction defined with resource security checking
File defined as LFILE
READ FILE('RFILE') Resource security checking done on RFILE
Resource security checking done on LFILE
READ FILE('LFILE') SYSID(FOR) No resource security checking done on LFILE
Resource security checking done on LFILE

Figure 25. Security checking done with and without SYSID.

This example illustrates what security checking is done when a transaction in an application-owning region issues a file control request against a remote file.

- *In the application-owning region (AOR), file RFILE is defined as remote, with a name of LFILE in the file-owning region (FOR). Resource security checking is active for the transaction that issues the file control request.*
- *In the FOR, resource security checking is active for the mirror transaction.*

There are two cases. In the first case:

1. *The transaction in the AOR issues EXEC CICS READ FILE('RFILE'). Resource security checking is performed for file RFILE.*
2. *The request is transmitted to the FOR, where resource security checking is performed for file LFILE.*

In the second case:

1. *The transaction in the AOR issues EXEC CICS READ FILE('RFILE') specifying the SYSID option. Resource security checking is not performed for file RFILE.*
2. *The request is transmitted to the FOR, where resource security checking is performed for file LFILE.*

For programming information on specifying the SYSID option, see the *CICS 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. Define this transaction 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).

To use the userid parameter in the DFHXCIC call, the client program must have surrogate-user authority to the specified userid. For more information, see “Surrogate user checking for EXCI calls” on page 114.

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.

Authorize 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” on page 223. If you define the mirror transaction with RESSEC(YES) in the server region, authorize these userids to access the server program that is being linked to by the mirror. If the server program accesses any CICS resources, authorize the same userids 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, authorize the same userids 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 Application Programming Reference*. For further information about DPL, refer to CICS distributed program link*CICS 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 30 through Table 31 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 30 shows how checking is done in the AOR when ATTACHSEC(LOCAL) has been specified.

Table 30. 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 31 shows how checking is done in the AOR when ATTACHSEC(IDENTIFY) has been specified.

Table 31. 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

Chapter 17. Security for data tables

This topic describes how to provide security for CICS shared data tables and coupling facility data tables.

Security for CICS shared data tables

To provide security for a shared data table when **cross-memory services** are used, ensure that:

- The file-owning region (FOR) that is acting as the shared data table server cannot be impersonated. See "SDT server authorization security check" for details of how you ensure this.
- An application-owning region (AOR) cannot gain access to data that it is not meant 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 through 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 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, 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 for CICS shared data tables is covered in the following topics:

- "Security checking for data tables"
- "SDT server authorization security check"
- "CONNECT security checks for AORs" on page 226.

Security checking for data tables

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 SDT server APPLIDs as protected resources. An SDT server's APPLID is represented by a DFHAPPL.*applid* profile in the RACF FACILITY resource class.

SDT server authorization security check

When a region attempts to be an SDT server, it calls RACF to check whether its user ID has the required access authority to its APPLID. If the call fails, the region cannot initialize the required SDT support to be a server. This minimizes the risk that an AOR might accept **counterfeit data records** from an FOR that is not properly authorized to act as an SDT server. This check is never bypassed, even when SEC=NO is specified at system initialization.

To act as a server for a protected APPLID, an SDT FOR's userid must have UPDATE (or higher) access to its DFHAPPL.*applid* profile in the FACILITY class. In the following example definitions, the APPLID of the FOR is CICSHF01, and its user ID is CICSSDT1:

```
RDEFINE FACILITY (DFHAPPL.CICSHF01) UACC(NONE)

PERMIT DFHAPPL.CICSHF01 CLASS(FACILITY) ID(CICSSDT1) ACCESS(UPDATE)
```

The above example authorizes one FOR to act as a server with APPLID CICSHF01, running under user ID CICSSDT1. The following example shows how to authorize a group of FORs, with user IDs defined as members of group SDTGRP1, to act as SDT servers using a generic profile in the FACILITY class:

```
RDEFINE FACILITY (DFHAPPL.CICSTST*) UACC(READ)

PERMIT DFHAPPL.CICSTST* CLASS(FACILITY) ID(SDTGRP1) ACCESS(UPDATE)
```

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:

- The request fails 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.
- The request succeeds 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 request is allowed in these cases because there is no evidence that you want to control access to the particular FOR APPLID.

CONNECT security checks for AORs

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.

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. In the following example definitions, three CICS AORs (userids is

CICSAOR1, CICSAOR2, and CICSAOR3) all require SDT access to the FOR represented by the DFHAPPL.CICSHF01 profile:

```
PERMIT DFHAPPL.CICSHF01 CLASS(FACILITY) ID(CICSAOR1 CICSAOR2 CICSAOR3)
       ACCESS(UPDATE)
```

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 226). If the result is a refusal, CICS does not permit shared access by the AOR to the FOR's APPLID.

Note that controlling SDT server authorization security and bind security 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 servers for that APPLID.

SDT bind-time security uses different definitions from those employed by ISC and (if using preset sessions) MRO. Therefore, 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

After the bind-security check, and when file security is in force in the FOR, the FOR checks whether the AOR is authorized to"sign on" to the FOR. This security check is optional, and applies only when the userid of the AOR is different from that of the FOR. It is the equivalent of ATTACHSEC(LOCAL) in an MRO environment (see "User security with MRO" on page 215). The AOR also requires READ authorization to the file it is trying to access in the FOR.

To implement file security checking by the FOR:

- Initialize the FOR with system initialization parameter SEC=YES
- Authorize the AOR with READ access to the FOR's APPLID profile in the APPL general resource class
- Specify the appropriate value on the XFCT system initialization parameter
- Authorize the AOR's region user ID with READ access to the required files in the file resource profiles named on the XFCT system initialization parameter.

For example, define the APPL profile for an FOR with APPLID CICSHF01, and the PERMIT command to enable the AORs with user IDs CICSAOR1 and CICSAOR2 to sign on to CICSHF01, as follows:

```
RDEFINE APPL CICSHF01 UACC(NONE) NOTIFY(sys_admin_userid)
PERMIT CICSHF01 CLASS(APPL) ID(CICSAOR1 CICSAOR2) ACCESS(READ)
```

For information about authorizing access to files, see "Security for files" on page 93.

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 226).

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.

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.

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 24.

Caution: If you use ISC instead of MRO for function shipping, 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.

Security for coupling facility data tables

CICS and MVS use RACF facilities to provide security for coupling facility data tables in the following areas:

1. Authorizing server access to a coupling facility list structure
2. Authorizing the server
3. Authorizing a CICS region's access to a coupling facility data table pool
4. Authorizing a CICS region to a CFDT
5. File resource security checking.

With the exception of items 4 and 5, which are optional, the other security checks are made automatically and are never bypassed. For items 2 and 3 in the above list, in cases when the system authorization facility (SAF) neither grants nor refuses access are resolved in the same way as the LOGON security check for CICS shared data table support (see “SDT server authorization security check” on page 225 for details).

An optional security check, which is controlled by server startup parameters, is provided for controlling access to specific tables within a coupling facility data table pool. This is described under “Authorizing a CICS region to a coupling facility data table” on page 229.

Authorizing server access to a list structure

Each coupling facility data table server requires access to the coupling facility list structure that contains its pool of coupling facility data tables. To permit access, give the server region user ID ALTER access to a FACILITY class general resource profile called IXLSTR.*structure_name*. Structure names for coupling facility data tables take the form DFHCFLS_*poolname*.

For example, if coupling facility data tables are defined in a pool called PRODCFT1, the list structure for this pool is named DFHCFLS_PRODCFT1 in the CFRM policy. To access this list structure, the server user ID for pool PRODCFT1 requires ALTER access to the IXLSTR profile, defined as follows:

```
RDEFINE FACILITY IXLSTR.DFHCFLS_PRODCFT1 UACC(NONE)
PERMIT IXLSTR.DFHCFLS_PRODCFT1 CLASS(FACILITY) ID(server_userid) ACCESS(ALTER)
```

Authorizing the server

When a CFDT server starts up for a given coupling facility data table pool CICS authorized cross-memory (AXM) services calls RACF to establish that it is authorized to act as a server for that pool. To authorize a coupling facility data table server to act as a server for its specified pool, give the server region user ID CONTROL access to a FACILITY class general resource profile called DFHCF.*poolname*.

For example, if the pool is PRODCFT1, define the profile and the required PERMIT statement as follows

```
RDEFINE FACILITY DFHCF.PRODCFT1 UACC(NONE)
PERMIT DFHCF.PRODCFT1 CLASS(FACILITY) ID(server_userid) ACCESS(CONTROL)
```

Authorizing a CICS region to a CFDT pool

Each CICS region requires authorization to connect to a coupling facility data tablepool. To authorize a CICS region to connect to a server and its pool, give the CICS region UPDATE access to the server's FACILITY class profile for the pool.

For example, if the pool is PRODCFT1, define the required PERMIT statement as follows:

```
PERMIT DFHCF.PRODCFT1 CLASS(FACILITY) ID(CICS_region_userid) ACCESS(UPDATE)
```

Authorizing a CICS region to a coupling facility data table

In addition to controlling a CICS region's access to a coupling facility data table pool, you can optionally control access to each CFDT in the pool. This security check, if active, is performed by the server each time a CICS region connects to a coupling facility data table for the first time. The resource security check is done as if for a CICS file owned by the coupling facility data table server region, using a profile defined in the general resource class specified on the SECURITYCLASS server initialization parameter. The default for this is the FCICSFCT class. For the profile name, use the table name as defined in the file resource definition.

You can optionally prefix the profile name using the server region user ID as the prefix by specifying SECURITYPREFIX=YES as a server initialization parameter. You can customize the prefix for this security check using the server initialization parameter SECURITYPREFIXID.

The coupling facility data table server performs the table security check by issuing a cross-memory mode FASTAUTH check, which requires the use of global in-storage security profiles. Access fails if a return code other than zero is received by the server in response to the FASTAUTH check. If the external security manager does not support cross-memory mode FASTAUTH or global in-storage profiles, coupling facility data table security checks are not possible and an error message is issued at server initialization time if table security checking is specified. For information about all server initialization parameters that can be specified, see the *CICS System Definition Guide*

File resource security checking

Normal CICS resource security for files is supported for coupling facility data tables. CICS performs the usual file resource security checks against signed-on users of transactions that access coupling facility data tables, using profiles defined in the general resource class named on the XFCT system initialization parameter.

See “Security for files” on page 93 for details of CICS file security.

Part 4. Security for TCP/IP clients

This part discusses how you can secure your applications when CICS participates in a client-server configuration, using TCP/IP communication protocols.

Chapter 18. About security for TCP/IP clients

TCP/IP connections between clients and servers — especially when they use the internet — are vulnerable to attack by malicious parties. An attacker might attempt to:

- Read confidential information flowing between client and server.

  You can protect your system by encrypting data that flows between the client and server.
- Falsify information flowing between client and server.

  You can protect your system by encrypting data that flows between the client and server, and by detecting data that has been tampered with.
- Impersonate a legitimate user of a client system.

  You can protect your system by authenticating users.

CICS supports several schemes for authenticating users. See “Identification and authentication” on page 236 for more information.

CICS can use the Secure Sockets Layer (SSL) security protocol or the Transport Layer Security (TLS) protocol to support secure TCP/IP connections. See “Support for security protocols” on page 244 for more information.

Message protection

Message protection is a term that describes the techniques used to ensure that a confidential message cannot be observed in transit, and cannot be illicitly changed.

The two aspects of message protection are:

Confidentiality
Protecting the message content from being intercepted

Integrity
Protecting a message from illicit modification

Confidentiality is achieved by encrypting the message (or parts of it) using a public key encryption scheme, so that only the intended recipient of the message can read it.

Integrity is achieved by digitally signing the message, so that the intended recipient can be confident that the message has not been changed illicitly.

Public key encryption

Public key encryption is a cryptographic system that uses two keys - a *public key* that is potentially known to everyone and a related *private key* that is known only to one party in an exchange of information.

The private and public keys used for public key encryption are related to each other in such a way that:

- It is not feasible to deduce the value of the private key from the public key, nor the public key from the private key.

  While the private key must be stored securely, and not made known to anyone but its owner, the public key can be made freely available to any user, with no risk of compromising the security of the private key.

- Information encrypted using the public key can be decrypted only with the private key.

  Information can be encrypted by any user, and sent securely to the holder of the private key: data encrypted with the public key is readable by only the holder of the private key.
- Information encrypted using the private key can be decrypted only with the public key.

  Only the holder of the private key can encrypt information that can be decrypted with the public key. Any party can use the public key to read the encrypted information; however, data that can be decrypted with the public key is guaranteed to originate with the holder of the private key.

Knowledge of a public key does not guarantee the identity of the owner of the corresponding private key, and so encryption of information with a public key cannot, on its own, prevent encrypted information falling into the wrong hands. Before a public key can be safely used to encrypt or decrypt information, the identity of the holder of the private key must be assured. This assurance is provided by a *digital certificate* which binds the public key to the identity of the private key's owner.

Digital signatures

A digital signature is information that is attached to data to assure the recipients of the data that it has not been altered and has originated from the signer of the message. Digital signatures perform an equivalent function to a handwritten signature on a paper document.

A digital signature consists of a *message digest* encrypted with the message sender's private key. The message digest, which is much shorter than the original message, is created from the message using a process known as *hashing*. It is not possible to reconstruct the original message from the message digest. The message, when combined with the signature, is a *signed message*.

The receiver of a signed message attempts to decrypt the signature using the sender's public key, thus changing it back into a message digest. Success indicates that the message was signed by the sender, because only the sender has the private key. The receiver then hashes the document data into a message digest, and compares it with the message digest obtained by decrypting the signature. If both digests are the same, the receiver can be sure that the signed message has not been changed.

A digital signature does not provide confidentiality. In other words, data that is not encrypted data can bear a digital signature.

Knowledge of a public key does not guarantee the identity of the owner of the corresponding private key, and so encryption of information with a public key cannot, on its own, prevent encrypted information falling into the wrong hands. Before a public key can be safely used to encrypt or decrypt information, the identity of the holder of the private key must be assured. This assurance is provided by a *digital certificate* which binds the public key to the identity of the private key's owner.

Digital certificates

A digital certificate is a digitally signed data structure that binds a public key to the identity of the private key's owner. The use of digital certificates ensures that the user of a public key can be confident of the ownership of the corresponding private key.

Digital certificates are issued by a trusted body, known as a Certificate Authority (CA), that is typically independent of the message sender and receiver (although if there is a trust relationship between the sender and the receiver, the certificate can be issued by one or the other). The certificate is encrypted with the CA's private key, and can be decrypted using the CA's public key, which is freely available to anyone who needs to read the certificate.

Following decryption, a valid certificate assures the reader that the certificate was indeed issued by the CA, and that the certificate has not been tampered with or forged.

A digital certificate contains information that identifies the certificate owner, and the certificate owner's public key, and is digitally signed by the CA. The receiver of a message containing a certificate uses the CA's public key to decrypt the certificate, verifies that it was issued by the CA and then obtains the sender's public key and identification information held within the certificate.

X.509 Certificates

ITU-T recommendation X.509 defines a widely used format for digital certificates.

An X.509 certificate contains

- Two *distinguished names*, which uniquely identify the Certificate Authority (CA), that issued the certificate and the *subject* (the individual or organization to whom the certificate was issued). The distinguished names contain several optional components:
  - Common name
  - Organizational unit
  - Organization
  - Locality
  - State or Province
  - Country
- A digital signature. The signature is created by the certificate authority using the public-key encryption technique:
  1. A secure hashing algorithm is used to create a digest of the certificate's contents.
  2. The digest is encrypted with the certificate authority's private key.
  3. The signature is decrypted with the CA's public key.
  4. A new digest of the certificate's contents is made, and compared with the decrypted signature. Any discrepancy indicates that the certificate may have been altered. The digital signature thus assures the receiver that no changes have been made to the certificate since it was issued.
- The subject's domain name. The receiver compares this with the actual sender of the certificate.
- The subject's public key.

Identification and authentication

Identification is the process by which the identity of a user is established, and *authentication* is the process by which a service confirms the claim of a user to use a specific identity by the use of credentials (usually a password or a certificate).

Identification

In CICS, identification can be accomplished in several ways:

- The client can supply a user ID directly. Typically this is done as part of the authentication process.

  You can identify users in this way when you use basic authentication with the ECI and HTTP application protocols.
- The client can supply information other than a user ID (for example, an SSL client certificate) during the authentication process. The information is mapped to a user ID in the security manager.

  You can identify users in this way when you use SSL client certificate authentication with the HTTP and IIOP application protocols.
- An intermediate server can establish the identity of the client, and pass it to the client. You can identify users in this way when you use asserted identity authentication with the IIOP application protocol.
- The user ID can be supplied in a user-replaceable program which is invoked for each inbound request.

  In the HTTP application protocol, the analyzer program can supply a user ID.

  In the IIOP protocol, the program specified in the URM attribute of the TCPIPSERVICE resource definition can supply a user ID.
- The user ID can be supplied in a URIMAP definition for an inbound request. You can identify users in this way when you use URIMAP definitions to handle requests on the HTTP application protocol.

If you do not use one of these methods to supply a user ID, the default user ID is used.

Identifying HTTP users

For the HTTP application protocol, you can identify the user in the following ways:

- A user ID can be obtained from the Web client using HTTP basic authentication.
- If the web browser sends a client certificate, you can use a user ID that is associated with the certificate.

  You can associate a certificate with a RACF userid in two ways:

  – You can use RACF commands to associate a certificate with a user ID.

  – CICS can automatically issue the RACF commands to associate a certificate with a user ID (which is obtained from the Web client using HTTP basic authentication).

  "Associating a RACF user ID with a certificate" on page 259 tells you how to do this.

For application-generated responses only, it is also possible for CICS to supply a user ID on behalf of the Web client:

- In an analyzer program that is used in the processing path for the request.
- In the USERID attribute of the URIMAP definition for a request.
- As the CICS default user ID.

It is important to note that if you use a URIMAP definition or analyzer program to set a user ID that has not been supplied by a client, or allow the CICS default user ID to be used, there is no authentication of the client's identity. You should only do this when communicating with your own client system, which has already authenticated its users, and communicates with the server in a secure environment.

When the HTTP response is to be provided by an application (an application-generated response), the order of precedence of user IDs is:

1. A user ID that you set using an analyzer program. This user ID can override a user ID obtained from the Web client or supplied by a URIMAP definition.
2. A user ID that you obtained from the Web client using basic authentication, or a user ID associated with a client certificate sent by the Web client.
3. A user ID that you specified in the URIMAP definition for the request.
4. The CICS default user ID, if no other can be determined.

When the HTTP response is to be provided by a URIMAP definition that specifies a CICS document template or z/OS UNIX file (a static response), the user ID used for the Web client is a user ID that you obtained from the Web client using basic authentication, or a user ID associated with a client certificate sent by the Web client. For static responses, it is not possible to supply a user ID on behalf of the Web client, nor to override an authenticated user ID obtained from a Web client.

For static responses, CICS only makes use of a supplied user ID if you specify resource security checking for the transaction. No default user ID is required for static responses. If the Web client does not supply a user ID, no resource security checking is carried out, even if resource security is active for the transaction.

The method used to identify the user is determined by the AUTHENTICATE and SSL attributes of the TCPIPSERVICE definition:

Table 32. How the user of an HTTP client is identified

AUTHENTICATE	SSL	How the user is identified
NO	NO or YES	The client does not supply a user ID. It can be supplied by an analyzer program or URIMAP definition, or allowed to default to the CICS default user ID, if applicable.
NO	CLIENTAUTH	If the client sends a certificate that is associated with a user ID, then that user ID applies, unless it is overridden by an analyzer program. If the client sends a certificate that is not associated with a user ID, a user ID can be supplied by an analyzer program or URIMAP definition, or allowed to default to the CICS default user ID, if applicable.
BASIC	all values	A user ID is obtained from the client, using HTTP basic authentication. This can be overridden by an analyzer program.
CERTIFICATE	CLIENTAUTH	If the client sends a certificate that is associated with a user ID, then that user ID applies, unless it is overridden by an analyzer program. If the client sends a certificate that is not associated with a user ID, or does not send a certificate, then the connection is rejected.

Table 32. How the user of an HTTP client is identified (continued)

AUTHENTICATE	SSL	How the user is identified
AUTOREGISTER	CLIENTAUTH	If the client sends a certificate that is associated with a user ID, then that user ID applies, unless it is overridden by an analyzer program. If the client sends a certificate that is not associated with a user ID, then the user ID is obtained from the client, using HTTP basic authentication, and the user ID is registered to the certificate. If the client does not send a certificate, then the connection is rejected.
AUTOMATIC	NO or YES	A user ID is obtained from the client, using HTTP basic authentication. This can be overridden by an analyzer program.
AUTOMATIC	CLIENTAUTH	If the client sends a certificate that is associated with a user ID, then that user ID applies, unless it is overridden by an analyzer program. If the client sends a certificate that is not associated with a user ID, then the user ID is obtained from the client, using HTTP basic authentication, and the user ID is registered to the certificate. If the client does not send a certificate, then the user ID is obtained from the client, using HTTP basic authentication.

Note:

1. This table does not list combinations of values for the AUTHENTICATE and SSL attributes which are invalid, and cannot be specified in the TCPIPSERVICE definition.
2. If HTTP basic authentication is used, CICS verifies the password. If the password is invalid, the connection is rejected.

For more information about CICS Web support architecture, see the *CICS Internet Guide*.

Identifying IIOP users

For IIOP requests, SSL client authentication can be used to identify the user. When the client sends a client certificate, you can identify the user by a user ID that you have previously associated with the certificate. IIOP users cannot register certificates automatically. For more information, see “Associating a RACF user ID with a certificate” on page 259.

If SSL client authentication does not provide a user ID, it is possible for CICS to supply a user ID on behalf of the client. The ID can be supplied by a user-replaceable program specified in the URM attribute of the TCPIPSERVICE resource definition.

If neither of these mechanisms provides a user ID, the user ID can default to the CICS default user ID.

The derived user ID is passed with the IIOP request to the request processor, for authentication of the request execution. If the request processor is executing in a different CICS region, the transmission of the user ID follows CICS rules for CONNECTION authentication.

The method used to identify the user is determined by the AUTHENTICATE and SSL attributes of the TCPIPSERVICE resource definition.

Table 33. How the user of an IIOP client is identified

AUTHENTICATE	SSL	How the user is identified
NO	NO or YES	The user ID can be provided by the user-replaceable program specified in the URM attribute of the TCPIPSERVICE resource definition. Alternatively, it can be allowed to default to the CICS default user ID.
NO	CLIENTAUTH	If the client sends a certificate that is associated with a user ID, then that user ID applies. If the client does not send a certificate, or sends a certificate that is not associated with a user ID, then the user ID can be provided by the user-replaceable program or allowed to default to the CICS default user ID.
CERTIFICATE	CLIENTAUTH	If the client sends a certificate that is associated with a user ID, then that user ID applies. If the client does not send a certificate, or sends a certificate that is not associated with a user ID, then the connection is rejected. The user-replaceable program cannot be used when the CERTIFICATE option is specified.
ASSERTED	CLIENTAUTH	The client in this case is typically an intermediate server. If the client sends a certificate that is associated with a user ID, then it is trusted to identify and authenticate its own clients, and the user ID sent in the IIOP request applies. If the client does not send a certificate, or sends a certificate that is not associated with a user ID, then the connection is rejected. The user-replaceable program cannot be used when the ASSERTED option is specified.
Note: This table does not list combinations of values for the AUTHENTICATE and SSL attributes which are invalid, and cannot be specified in the TCPIPSERVICE definition.		

Identifying ECI users

For the ECI protocol you can use basic authentication to identify the user. Specify ATTACHSEC(VERIFY) in the TCPIPSERVICE definition for the ECI client. Specify ATTACHSEC(LOCAL) if you do not want to identify the user.

Identifying IPIC users

For the IPIC protocol you can use basic authentication to identify the user. Specify USERAUTH(VERIFY) in the IPCONN resource definitions. Specify USERAUTH(LOCAL) if you do not want to identify the user.

Authentication

In many systems, the user's authenticity is verified by checking a password supplied by the user. In a system in which there is no possibility of a password being intercepted, this level of authentication may be sufficient; however, in an insecure network, it is possible that passwords can be intercepted, and used to impersonate legitimate users of the system.

In an environment where your applications may be accessed by users across the internet, and by users who are outside the control of your organization, a more secure method of authentication is required.

On the other hand, there are situations where a limited level of authentication is sufficient. If you have a client system that authenticates its users, and communicates with a server in a secure environment, you may not need to authenticate end users at the server, but rely entirely on the client's authentication mechanisms.

CICS supports the following authentication schemes:

Basic authentication

The client's identity is authenticated by a password. This level of authentication is appropriate in an environment where passwords cannot be intercepted and used to impersonate an end user.

You can use basic authentication with the HTTP and ECI application protocols.

SSL client certificate authentication

The client's identity is authenticated with a client certificate issued by a trusted third party (or Certificate Authority). This level of authentication is appropriate in an environment where information flowing in the network could be intercepted, and used to impersonate an end user.

You can use SSL client certificate authentication with the HTTP and IIOP application protocols.

For more information about SSL, see “Support for security protocols” on page 244.

Asserted identity authentication

Asserted identity authentication can be used when an IIOP client communicates with the target server through an intermediate server, and both servers use the same security manager:

1. The intermediate server's identity is authenticated by the target server using SSL client certificate authentication.
2. Through the security manager, the target server verifies that the intermediate server can be trusted to authenticate its clients.
3. When the intermediate server receives a request, it authenticates the client using whatever authentication protocol is appropriate. If the client is successfully authenticated, the intermediate server passes the request to the target server.
4. Because the target server trusts the intermediate server to authenticate the client, it makes no further checks of the client's authenticity before processing the client's request.

You can use asserted identity authentication with the IIOP application protocols. In CICS, a CorbaServer can be configured as an intermediate or target server.

Authenticating HTTP users

For the HTTP application protocol, you can authenticate the user in the following ways:

- You can use HTTP basic authentication.
- You can use SSL client certificate authentication.

The authentication scheme is specified by the AUTHENTICATE and SSL attributes of the TCPIPSERVICE definition:

Authentication scheme	AUTHENTICATE	SSL	Notes
HTTP with no authentication	NO	NO or YES	
HTTP with basic authentication	BASIC	NO, YES or CLIENTCERT	
HTTP with basic authentication	AUTOMATIC	NO, YES or CLIENTCERT	If SSL(CLIENTCERT) is specified, and the client sends a certificate, then SSL client certificate authentication is used.
HTTP with SSL client certificate authentication	CERTIFICATE or AUTOREGISTER	CLIENTCERT	If the client does not send a certificate, the connection is not established.
HTTP with SSL client certificate authentication	AUTOMATIC	CLIENTCERT	If the client does not send a certificate, then basic authentication is used.

Authenticating IIOP users

For the IIOP application protocol, you can authenticate the user using SSL client certificate authentication or asserted identity authentication.

The authentication scheme used by a port is specified by the AUTHENTICATE and SSL attributes of the TCPIPSERVICE resource definition which defines the characteristics of the port.

Authentication method	AUTHENTICATE	SSL	Associated CORBASERVER attribute
IIOP with no authentication	NO	NO	UNAUTH
IIOP with no authentication	NO	YES	SSLUNAUTH
IIOP with SSL client certificate authentication	CERTIFICATE	CLIENTCERT	CLIENTCERT
IIOP with asserted identity authentication	ASSERTED	CLIENTCERT	ASSERTED

A CorbaServer can support more than one authentication scheme. Each CORBASERVER resource definition is associated with one or more TCPIPSERVICE resource definitions, and each TCPIPSERVICE resource definition supports a different mechanism for authentication and identification.

- The UNAUTH attribute of the CORBASERVER resource definition names a TCPIPSERVICE resource definition for a port which is used for inbound IIOP with no authentication.

  Note: You must specify a value for the UNAUTH attribute when you create a CORBASERVER resource definition, even if you intend that all inbound requests to this CorbaServer should be authenticated. This is because the PORTNUMBER attribute of the TCPIPSERVICE is required in order to construct IORs that are exported from this logical server.
- The SSLUNAUTH attribute of the CORBASERVER resource definition names a TCPIPSERVICE resource definition for a port which is used for inbound IIOP with SSL encryption but no client authentication.
- The CLIENTCERT attribute of the CORBASERVER resource definition names a TCPIPSERVICE resource definition for a port which is used for inbound IIOP with SSL client certificate authentication.
- The ASSERTED attribute of the CORBASERVER resource definition names a TCPIPSERVICE resource definition for a port which is used for inbound IIOP with asserted identity authentication.

To change the association between an installed CORBASERVER definition and its TCPIPSERVICE definitions, discard and reinstall the CORBASERVER definition.

The authentication protocols supported by an object are made known to clients in the IOR for the object:

- When CICS is the target server, the authentication protocols are specified in CORBASERVER resource definitions. When the Generic Factory Interoperable Object Reference (GenFacIOR) of the CorbaServer is published, the authentication protocols supported by each object are made known to clients in the GenFacIOR.
- When CICS is the intermediate server, it examines the IOR for the server object to determine which authentication protocols the object supports, and selects the protocol to use. If more than one protocol is supported, CICS selects the first supported protocol from:
  1. Asserted identity authentication
  2. SSL client certificate authentication

  If these authentication protocols are not supported, no authentication is used.

| CICS can use asserted identity authentication when communicating with WebSphere® Application Server for z/OS or with other CICS regions. Both the target server and the intermediate server must use the same security manager.

| The protocol that a CICS CorbaServer uses for asserted identity authentication is the z/OS Secure Authentication Service (z/SAS) protocol. In WebSphere Application Server for z/OS Version 6.1 or later, this protocol is no longer available, but CICS is able to communicate with these product versions using a limited implementation of the Common Secure Interoperability Version 2 (CSIv2) protocol for identity assertion. (Release 6.1.0.13 or later of WebSphere Application Server for z/OS is required to support this function.) To enable this function in CICS, apply APARs

PK59219 and PK64022 to CICS, then specify the system property **-Dcom.ibm.cics.iiop.CSIv2Enabled=true** in all of the JVM properties files used in the CICS region.

CICS only uses the CSIv2 protocol if WebSphere Application Server for z/OS requires that protocol; if the protocol is optional, CICS does not use it. When CICS is communicating with earlier versions of WebSphere Application Server for z/OS or with other CICS regions, CICS chooses the z/SAS protocol instead of the CSIv2 protocol whenever support for the z/SAS protocol is available.

Authenticating ECI users

For the ECI protocol you can use basic authentication to authenticate the user. Specify ATTACHSEC(VERIFY) in the TCPIPSERVICE definition for the ECI client. Specify ATTACHSEC(LOCAL) if you do not want to authenticate the user.

Authenticating IPIC users

For IPIC connections, you can use basic user authentication. Specify USERAUTH(VERIFY) in the ../dfha4/resources/ipconn/dfha4_overview.dita . Specify USERAUTH(LOCAL) if you do not want to authenticate the user.

Configuring CICS to use asserted identity authentication

You can configure each CorbaServer in your CICS region as:

An intermediate server
The intermediate server authenticates a client using whatever authentication protocol is appropriate.

A target server
The target server trusts the intermediate server to authenticate the client, and does not perform its own authentication.

In either case, you must establish a trust relationship between the intermediate and target servers.

If a CICS CorbaServer needs to support asserted identity authentication for IIOP messages sent from WebSphere Application Server for z/OS Version 6.1 or later, you need to enable a suitable authentication protocol before completing the steps in these tasks. To do this, apply APARs PK59219 and PK64022 to CICS, then specify the system property **-Dcom.ibm.cics.iiop.CSIv2Enabled=true** in all of the JVM properties files used in the CICS region. Release 6.1.0.13 or later of WebSphere Application Server for z/OS is required to support this function.

Establishing a trust relationship between the servers

To establish a trust relationship between the intermediate and target servers, where the target server is a CICS CorbaServer, perform the following steps:

1. Configure your CICS region to use SSL authentication. See Chapter 19, “Configuring CICS to use SSL,” on page 253 for more information.
2. Associate the intermediate server's client certificate with a RACF userid. For more information, see “Associating a RACF user ID with a certificate” on page 259.
3. Create a profile named DFH.*applid*.*corbaserver*.ASSERTID in the SERVAUTH general resource class, where

   applid
   is the APPLID of the CICS region

corbaserver
is the name of the target CorbaServer

For example, use the following RACF command:

```
RDEFINE SERVAUTH DFH.applid.corbaserver.ASSERTID UACC(NONE)
```

4. Give the intermediate server's userid (established in step 2 on page 243) READ authority to the profile. For example, use the following RACF command:

```
PERMIT DFH.applid.corbaserver.ASSERTID CLASS(SERVAUTH)
      ID(server_userid) ACCESS(READ)
```

For information about establishing a trust relationship where another product is the target server, see the documentation for the target server.

Configuring a CorbaServer as an intermediate server

To configure a CICS CorbaServer as an intermediate server, perform the following steps:

1. Configure your CICS region to use SSL authentication. See Chapter 19, “Configuring CICS to use SSL,” on page 253 for more information.
2. Specify the certficate that will authenticate this server in the CERTIFICATE attribute of the CORBASERVER definition.

Configuring a CorbaServer as the target server

To configure a CICS Corbaserver as the target server, perform the following steps:

1. Configure your CICS region to use SSL authentication. See Chapter 19, “Configuring CICS to use SSL,” on page 253 for more information.
2. Define and install a TCPIPSERVICE to define the characteristics of the port which is used for inbound IIOP with asserted identity authentication. Specify AUTHENTICATE(ASSERTED) when you define the TCPIPSERVICE.
3. Define and install a CORBASERVER. Specify the name of the TCPIPSERVICE you defined in step 2 in the ASSERTED attribute.

Support for security protocols

CICS supports two security protocols that can be used to provide secure communication over the Internet. The first is the Secure Sockets Layer (SSL) 3.0 protocol. The second is the Transport Layer Security (TLS) 1.0 protocol, which is the latest industry standard SSL protocol and is based on SSL 3.0. The TLS 1.0 specification is documented in RFC2246 and is available on the Internet at www.rfc-editor.org/rfcsearch.html. Any connections that require encryption will automatically use the TLS protocol, unless the client specifically requires SSL 3.0.

Note: For clarity, the term SSL is used to refer to both protocols in the documentation, except where a specific point about either protocol is required.

The primary aim of TLS is to make the Secure Sockets Layer more secure and to make the specification of the protocol more precise and complete. TLS provides the following enhancements over SSL 3.0:

Key-Hashing for Message Authentication
TLS uses Key-Hashing for Message Authentication Code (HMAC), which ensures that a record cannot be altered while travelling over an open network such as the Internet. SSL Version 3.0 also provides keyed message authentication, but HMAC is considered more secure than the (Message Authentication Code) MAC function that SSL Version 3.0 uses.

Enhanced Pseudorandom Function (PRF)
: PRF is used for generating key data. In TLS, the PRF is defined with the HMAC. The PRF uses two hash algorithms in a way that guarantees its security. If either algorithm is exposed then the data remains secure as long as the second algorithm is not exposed.

Improved finished message verification
: Both TLS 1.0 and SSL 3.0 provide a finished message to both endpoints that authenticates that the exchanged messages were not altered. However, TLS bases this finished message on the PRF and HMAC values, which is more secure than SSL Version 3.0.

Consistent certificate handling
: Unlike SSL 3.0, TLS attempts specify the type of certificate which must be exchanged between TLS implementations.

Specific alert messages
: TLS provides more specific and additional alerts to indicate problems that either session endpoint detects. TLS also documents when certain alerts should be sent.

The main features of the security protocols are:

Privacy
: The data to be exchanged between the client and the server is encrypted. See “SSL encryption” for more information.

Integrity
: Data which is transmitted using the SSL protocols is protected against tampering by a **message authentication code** (MAC). The MAC is computed from the data contents using a secure hashing algorithm and transmitted with the data. It is computed again by the receiver, and compared with the value transmitted by the sender. A mismatch between the two values of the MAC indicates that the data may have been tampered with.

Authentication
: SSL uses digital certificates to authenticate servers to clients, and — optionally — clients to servers. See “SSL authentication” on page 246 for more information.

SSL encryption

The SSL protocol operates between the application layer and the TCP/IP layer. This allows it to encrypt the data stream itself, which can then be transmitted securely, using any of the application layer protocols. Two encryption techniques are used:

- Public key cryptography standard (PKCS), which encrypts and decrypts certificates during the SSL handshake. Encryption keys are created in pairs, a public key and its associated private key. Data encrypted with a given public key can be decrypted only with the associated private key; this means that data is readable by only the intended recipient. Data encrypted with a given private key can be decrypted only with the associated public key; this means that authentication data is assured to originate from the owner of the private key.
- A mutually agreed symmetric encryption technique, such as DES (data encryption standard), or triple DES, is used in the data transfer following the handshake.

PKCS, as used by SSL, works briefly as follows:

1. When a certificate is created, an algorithm based on two random numbers is used to create a private key and public key for the certificate owner. The private and public keys which result are related to each other such that:
   - **It is not feasible to deduce the value of the private key from the public key, nor the public key from the private key**

     The private key is stored securely, and is not made known to anyone but its owner. The public key can be made freely available to any user, with no risk of compromising the security of the private key.
   - **Information encrypted using the public key can be decrypted only with the private key**

     Information can be encrypted by any user, and sent securely to the holder of the private key. A third party cannot use the public key to read the information.
   - **Information encrypted using the private key can be decrypted only with the public key**

     Only the holder of the private key can encrypt information that can be decrypted with the public key. A third party cannot pose as the sender of the information.

SSL authentication

To make an environment secure, you must be sure that any communication is with "trusted" sites whose identity you can be sure of. SSL uses certificates for authentication — these are digitally signed documents which bind the public key to the identity of the private key owner. Authentication happens at connection time, and is independent of the application or the application protocol. Authentication involves making sure that sites with which you communicate are who they claim to be. With SSL, authentication is performed by an exchange of certificates, which are blocks of data in a format described in ITU-T standard X.509. The X.509 certificates are issued, and digitally signed by an external authority known as a certificate authority.

A certificate contains

- Two **distinguished names**, which uniquely identify the **issuer** (the certificate authority that issued the certificate) and the **subject** (the individual or organization to whom the certificate was issued). The distinguished names contain several optional components:
  – Common name
  – Organizational unit
  – Organization
  – Locality
  – State or Province
  – Country
- A digital signature. The signature is created by the certificate authority using the public-key encryption technique:
  1. A secure hashing algorithm is used to create a digest of the certificate's contents.
  2. The digest is encrypted with the certificate authority's private key.

  The digital signature assures the receiver that no changes have been made to the certificate since it was issued:
  1. The signature is decrypted with the certificate authority's public key.

2. A new digest of the certificate's contents is made, and compared with the decrypted signature. Any discrepancy indicates that the certificate may have been altered.

- The subject's domain name. The receiver compares this with the actual sender of the certificate.
- The subject's public key.

Certificates are used to authenticate clients to servers, and servers to clients; the mechanism used is essentially the same in both cases. However, the server certificate is mandatory — that is, the server must send its certificate to the client — but the client certificate is optional: some clients may not support client certicates; other may not have certificates installed. Servers can decide whether to require client authentication for a connection.

Certificate authorities

In order that one system can be assured that a certificate received from another system is genuine, a trusted third party that can vouch for the certificate is needed.

Certificate authorities are independent bodies who act as the trusted third parties, by issuing certificates for use by others. Before issuing a certificate, a certificate authority will examine the credentials of the person or organization that has requested the certificate. When the certificate has been issued, information about it is held on a publicly accessible repository. Users can consult the repository to check the status and validity of any certificates received.

Certificate authorities issue several levels of security certificates for different purposes. For example:

- Secure e-mail
- Client authentication
- Server authentication

CICS can check every certificate it receives from a client for a revoked status by using certificate revocation lists. A certificate revocation list details all the revoked certificates for a particular certificate authority. These lists are freely available to download from the Internet. If the certificate has a revoked status, CICS closes the SSL connection immediately. To find out how to set up certificate revocation lists, see “Using certificate revocation lists (CRLs)” on page 264.

Cipher suites

There are many different algorithms which can be used for encrypting data, and for computing the message authentication code. Some provide the highest levels of security, but require a large amount of computation for encryption and decryption; others are less secure, but provide rapid encryption and decryption. The length of the key used for encryption affects the level of security - the longer the key, the more secure the data.

To allow users to select the level of security that suits their needs, and to enable communication with others who may have different needs, SSL defines *cipher suites*, or sets of ciphers. When an SSL connection is established, the client and server exchange information about which cipher suites they have in common. They then communicate using the common cipher suite that offers the highest level of security. If they do not have a cipher suite in common, then secure communication is not possible and CICS closes the connection.

Use the ENCRYPTION system initialization parameter to specify the level of encryption that CICS should use. The default is STRONG, which means that CICS can use all of the cipher suites to negotiate with clients. You can set a minimum as well as a maximum encryption level by editing the list of cipher suites in the CIPHERS attribute on the appropriate resource definition.

To specify the level of encryption required:

For inbound HTTP and IIOP
Use the CIPHERS attribute of the TCPIPSERVICE resource. This automatically defines the PRIVACY attribute.

For outbound HTTP and web service requests
Use the CIPHERS attribute of the URIMAP resource definitions.

For outbound IIOP
Use the CIPHERS attribute of the CORBASERVER resource. This automatically defines the OUTPRIVACY attribute.

For inbound IPIC
Use the CIPHERS attribute of the TCPIPSERVICE resource definition.

For outbound IPIC
Use the CIPHERS attribute of the IPCONN resource definition.

For inbound CICSPlex SM Web User Interface requests
Use the **TCPIPSSLCIPHERS** Web User Interface server initialization parameter. This value has the same syntax as the CIPHERS attribute of the TCPIPSERVICE resource, but it is limited to a maximum of 22 cipher suites.

The PRIVACY and OUTPRIVACY attributes are no longer supported, except in compatibility mode. The values are determined by the list of ciphers in the CIPHERS attribute as follows:

NOTSUPPORTED
Specified when the list of ciphers in the CIPHERS attribute only includes ciphers with no encryption. For example, cipher suites 01 and 02.

REQUIRED
Specified when the list of ciphers in the CIPHERS attribute only includes cipher suites with encryption. For example, if ENCRYPTION=STRONG is specified, the full list of cipher suites are listed in the CIPHERS attribute. If you remove 01 and 02, the PRIVACY attribute changes to REQUIRED.

SUPPORTED
Specified when the list of ciphers in the CIPHERS attribute includes 01 and 02 in combination with any other cipher suites.

The cipher suites supported by z/OS Versions 1.6 and 1.7 and CICS are shown in Table 34. More cipher suites are supported by z/OS Version 1.8.

Table 34. Cipher suites supported by z/OS and CICS

Cipher suite	Encryption algorithm	Key length	Hash algorithm	Key exchange	Certificate
00		None	None	RSA	RSA
01		None	MD5	RSA	RSA
02		None	SHA-1	RSA	RSA
03	RC4	40 bits	MD5	RSA	RSA

Table 34. Cipher suites supported by z/OS and CICS (continued)

Cipher suite	Encryption algorithm	Key length	Hash algorithm	Key exchange	Certificate
04	RC4	128 bits	MD5	RSA	RSA
05	RC4	128 bits	SHA-1	RSA	RSA
06	RC2	40 bits	MD5	RSA	RSA
09	DES	56 bits	SHA-1	RSA	RSA
0A	Triple DES	168 bits	SHA-1	RSA	RSA
0C	DES	56 bits	SHA-1	fixed Diffie-Hellman	DSS
0D	Triple DES	168 bits	SHA-1	fixed Diffie-Hellman	DSS
0F	DES	56 bits	SHA-1	fixed Diffie-Hellman	RSA
10	Triple DES	168 bits	SHA-1	fixed Diffie-Hellman	RSA
12	DES	56 bits	SHA-1	ephemeral Diffie-Hellman	DSS
13	Triple DES	168 bits	SHA-1	ephemeral Diffie-Hellman	DSS
15	DES	56 bits	SHA-1	ephemeral Diffie-Hellman	RSA
16	Triple DES	168 bits	SHA-1	ephemeral Diffie-Hellman	RSA
2F	AES	128 bits	SHA-1	RSA	RSA
30	AES	128 bits	SHA-1	fixed Diffie-Hellman	DSS
31	AES	128 bits	SHA-1	fixed Diffie-Hellman	RSA
32	AES	128 bits	SHA-1	ephemeral Diffie-Hellman	DSS
33	AES	128 bits	SHA-1	ephemeral Diffie-Hellman	RSA
35	AES	256 bits	SHA-1	RSA	RSA
36	AES	256 bits	SHA-1	fixed Diffie-Hellman	DSS
37	AES	256 bits	SHA-1	fixed Diffie-Hellman	RSA
38	AES	256 bits	SHA-1	ephemeral Diffie-Hellman	DSS
39	AES	256 bits	SHA-1	ephemeral Diffie-Hellman	RSA

Table 34. Cipher suites supported by z/OS and CICS (continued)

Cipher suite	Encryption algorithm	Key length	Hash algorithm	Key exchange	Certificate
Terms used in this table are: **AES** Advanced encryption Standard **DES** Data Encryption Standard **Triple DES** DES applied three times. **DSS** Digital Signature Standard **MD5** Message Digest algorithm **RC** Rivest encryption **RSA** Rivest, Shamir, Adleman public key. **SHA** Secure Hash algorithm					

The SSL handshake

The SSL handshake is an exchange of information that takes place between the client and the server when a connection is established. It is during the handshake that client and server negotiate the encryption algorithms that they will use, and authenticate one another. The main features of the SSL handshake are:

- The client and server exchange information about the SSL version number and the cipher suites that they both support.
- The server sends its certificate and other information to the client. Some of the information is encrypted with the server's private key. If the client can successfully decrypt the information with the server's public key, it is assured of the server's identity.
- If client authentication is required, the client sends its certificate and other information to the server. Some of the information is encrypted with the client's private key. If the server can successfully decrypt the information with the client's public key, it is assured of the client's identity.
- The client and server exchange random information which each generates and which is used to establish session keys: these are symmetric keys which are used to encrypt and decrypt information during the SSL session. The keys are also used to verify the integrity of the data.

The SSL cache

The SSL cache is used to store session ids from the negotiation between clients and CICS.

The SSL cache allows CICS to perform partial handshakes with clients that it has previously authenticated. In a local CICS region, the SSL cache is part of the enclave for the S8 TCBs. You have the option of sharing the SSL cache across a sysplex if this is appropriate for your CICS system. You can use sysplex caching if you have multiple CICS socket-owning regions that accept SSL connections at the same IP address.

If you want to share the cache between regions, activate the system SSL started task GSKSRVR and use the system initialization parameter SSLCACHE. The default is to use the local region cache, but you can change this by specifying the option SYSPLEX. CICS will use the SSL cache in the coupling facility instead to store session ids.

The SSL pool

CICS uses the open transaction environment (OTE) to manage SSL connections and requests to LDAP using the DFHDDAPX XPI interface.

To improve the number and performance of SSL connections in CICS, each SSL connection uses an S8 TCB from the SSL pool. The S8 TCBs run as UNIX pthreads and are owned by an open mode TCB called SP. The SP TCB owns a single enclave, in which all the S8 TCBs run. This enclave also includes the SSL cache. This provides the benefit of saving storage below the line, allowing many more simultaneous SSL connections in CICS.

The S8 TCBs are contained in an SSL pool, which is managed by the CICS dispatcher. The S8 TCBS are allocated from the new SSL pool, but are only locked to a transaction for the period that it needs to perform SSL or LDAP functions. After the SSL or LDAP request is complete, the TCB is released back into the SSL pool to be reused. The MAXSSLTCBS system initialization parameter specifies the maximum number of S8 open TCBs in the SSL pool. The default value is 8, but you can specify up to 1024.

You can monitor the performance of the SSL pool and the S8 TCBs using the dispatcher reports from DFH0STAT and DFHSTUP. The statistics include information on how often the maximum number of S8 TCBs are reached, the delay before a TCB is allocated and the actual number of TCBs in the SSL pool.

Chapter 19. Configuring CICS to use SSL

CICS can use the Secure Sockets Layer (SSL) or the Transport Layer Security (TLS) security protocols to support secure TCP/IP connections. To authenticate servers to clients, create certificates and key rings in RACF and ensure that the CICS region and resources are correctly configured to support security.

Before you begin to configure CICS, decide which type of certificates to use in SSL handshakes.

You can use RACF to create certificates, but you must configure your clients to ensure that they can recognize the RACF server certificate. If you cannot configure your clients in this way, for example when clients are external to your organization, use a certificate signed by an external certificate authority.

Complete the following tasks to configure CICS to use SSL:

1. Set the correct authorizations in RACF to create a key ring, create a signing certificate (certificate authority certificate), and to add certificates to the key ring.
2. Optional: If you decide to use a certificate from a certificate authority, create a certificate request using RACF and send it to the certificate authority. You might have to wait a number of days to receive a signing certificate from the certificate authority. If your chosen certificate authority does not have its certificate built in to RACF, you might have to import it.
3. Create a key ring. You must create a key ring in the RACF database. The key ring contains:
   - Your public and private keys
   - Your server certificates
   - Signing certificates for the server certificates
   - Signing certificates for any client certificates owned by clients with which you expect CICS to communicate using client authentication.
4. Create the certificates and add them to the key ring.
5. Ensure that the CICS region has access to the z/OS system SSL library SIEALNKE. You can use STEPLIB or JOBLIB statements, or use the system link library.
6. Define the CICS system initialization parameters that are related to security. In particular, specify the name of the key ring that you created in the **KEYRING** system initialization parameter.
7. Define TCPIPSERVICE resources.

CICS supplies a sample REXX program, DFH$RING, that contains all of the RACF commands to create a key ring, create a signing certificate, create additional certificates, and add them to the key ring. DFH$RING contains sample values which are suitable for building a test key ring only. You must edit all the values if you want to create a key ring that is suitable for a production environment.

Authorizing users to administer certificates in the key ring

You must grant access to the appropriate profiles in the FACILITY class in RACF to build a key ring in the RACF database, create a signing certificate, and administer certificates in the key ring.

You must grant this access only to users who administer CICS regions and not to general CICS users. The following profiles are available:

Access level	Profile	Description
CONTROL	IRR.DIGTCERT.GENCERT	Allow certificates to be signed by a CERTAUTH certificate.
	IRR.DIGTCERT.ADD	Allow a CERTAUTH certificate to be generated on first execution.
	IRR.DIGTCERT.CONNECT	Connect CERTAUTH certificates for other users.
UPDATE	IRR.DIGTCERT.CONNECT	Connect CERTAUTH certificates to your key ring.
	IRR.DIGTCERT.*	Manage certificates for other users.
READ	IRR.DIGTCERT.*	Manage certificates for your own user ID.

IRR.DIGTCERT.* contains the wildcard asterisk and is intended as a generic profile.

CICS provides a sample REXX program, DFH$RING, which you can edit to create a key ring, signing certificate, and other certificates as appropriate. The default values are suitable only for a test key ring and not for production. To run DFH$RING, you must have the required authorization access in RACF.

1. To allow generic profiles to be created in the FACILITY class, issue the following command:
   ```
   SETROPTS GENERIC(FACILITY)
   ```
2. Issue the following commands:
   ```
   RDEFINE FACILITY(IRR.DIGTCERT.*)
   RDEFINE FACILITY(IRR.DIGTCERT.ADD)
   RDEFINE FACILITY(IRR.DIGTCERT.CONNECT)
   RDEFINE FACILITY(IRR.DIGTCERT.GENCERT)
   ```
3. Depending upon whether the FACILITY class is RACLISTed or not, issue one of the following commands:
   ```
   SETROPTS RACLIST(FACILITY) REFRESH
   SETROPTS GENERIC(FACILITY) REFRESH
   ```
4. Give a user ID or group, *ringuser*, authority to use the RACF commands to create and manage certificates. Issue the following commands:
   ```
   PERMIT IRR.DIGTCERT.*       CLASS(FACILITY) ID(ringuser) ACCESS(READ)
   PERMIT IRR.DIGTCERT.CONNECT CLASS(FACILITY) ID(ringuser) ACCESS(UPDATE)  (for self)
   PERMIT IRR.DIGTCERT.CONNECT CLASS(FACILITY) ID(ringuser) ACCESS(CONTROL)  (for another user)
   PERMIT IRR.DIGTCERT.GENCERT CLASS(FACILITY) ID(ringuser) ACCESS(CONTROL)
   ```
5. Give a user ID, *certauser*, authority to create a signing certificate (certificate authority certificate). RACF uses this certificate to sign all the other certificates. Issue the following command:
   ```
     PERMIT IRR.DIGTCERT.ADD CLASS(FACILITY) ID(certauser) ACCESS(CONTROL)
   ```

   If you plan to use the DFH$RING program, your user ID must have CONTROL access to create the signing certificate the first time you run the program. After you have created the signing certificate, you require only UPDATE access.

You can add certificate information for your own user ID if you have READ access to the IRR.DIGTCERT.ADD profile in the FACILITY class. You can add certificate information for other user IDs if you have UPDATE access to the IRR.DIGTCERT.ADD profile in the FACILITY class. If you have SPECIAL authority, you can run RACDCERT ADD for any user ID. You can also generate a digital certificate for any RACF-defined user or for any certificate authority or site certificate with SPECIAL authority.

If you want to use a certificate from a certificate authority to authenticate with a client, create a certificate request using RACF and send it to the certificate authority. If you want to use RACF certificates only, you can build a key ring.

Requesting a certificate from a certificate authority

You can use RACF to request a signing certificate (certificate authority certificate) from a certificate authority such as Verisign. Use an external certificate to authenticate your server to clients that cannot recognize RACF certificates.

You must have authorization to use the **RACDCERT** command. This command installs and maintains digital certificates, key rings, and digital certificate mappings in RACF.

1. Create a self-signed certificate in RACF as a placeholder:
   ```
   RACDCERT ID(foruser) GENCERT,
    SUBJECTSDN(CN('username')
              T ('username''s certificate')
              OU('department')
              O ('organization')
              L ('city')
              SP('state')
              C ('country'))
    NOTBEFORE(DATE(start) TIME(00:00:00))
    NOTAFTER (DATE(finish) TIME(23:59:59))
    WITHLABEL(self-signed-certlabel)
    SIZE     (1024)
   ```
2. Generate a certificate request, based on the placeholder certificate, to send to your external certificate authority. Use the **RACDCERT GENREQ** command:
   ```
   RACDCERT ID(cics-region-userid) GENREQ(LABEL('label'))
            DSN('request.dataset')
   ```
   where *label* is the placeholder self-signed certificate. RACF saves the certificate request in the data set specified in the **DSN** parameter.
3. Send the certificate request to the certificate authority, using a method that the certificate authority accepts.
4. When you receive the certificate, save it in a new data set.
5. Optional: If you are using a certificate authority that is not one of the default certificate authorities, for which certificates are already stored in the key database, you must import the certificate authority's certificate into your RACF database. See Importing a certificate from a file as a trusted CA certificate in the z/OS information center.
6. Replace the self-signed certificate with your new CA-signed certificate:
   ```
   RACDCERT ID(cics-region-userid) ADD('response.dataset') TRUST
   ```

Create the key ring in the RACF database and add your CA-signed certificate.

Building a key ring manually

In CICS, the required server certificate and related information about certificate authorities are held in a *key ring* in the RACF database. The key ring contains your system's private and public key pair, together with your server certificate and the certificates for all the certificate authorities that might have signed the certificates you receive from your clients.

Before you can use SSL with CICS, you must create a key ring that contains a private and public key pair and a server certificate. To create a key ring you must have UPDATE authority to the IRR.DIGTCERT.ADDRING resource in the FACILITY class. If you want to share certificates in a key ring between CICS regions, the CICS regions must have the same user ID and the user ID must own the key ring.

The **RACDCERT** command installs and maintains public key infrastructure (PKI) private keys and certificates in RACF. You can either manually issue the **RACDCERT** command to create a new key ring or you can use the DFH$RING sample program.

To create a key ring manually, follow these steps:

Issue the following **RACDCERT** command:

```
RACDCERT ID(cics-region-userid) ADDRING(ringname)
```

The key ring must be associated with the CICS region user ID.

RACF creates the key ring in the RACF database. If there is a key ring of the same name already in the RACF database, it is replaced with the new key ring.

Create a signing certificate (certificate authority certificate) and add it to the key ring.

Building a key ring with certificates using DFH$RING

DFH$RING is a sample REXX program that builds a key ring, creates a signing certificate (certificate authority certificate), creates additional certificates, and adds the certificates to the key ring.

You must have the required authorization to run the RACF commands. Your user ID must have CONTROL access to create the signing certificate the first time your run the program. If you run the program again, you require only UPDATE access.

DFH$RING is in library CICSTS32.CICS.SDFHSAMP. Edit the values in DFH$RING to create a suitable key ring and certificates:

1. Enter values for the *firstname*, *lastname*, and *hostname* variables. The *firstname* and *lastname* values are concatenated together to form the name of the key ring. Enter the host name of your Web server for the *hostname* variable.
2. Optional: Enter a value for the *FORUSER* variable if you are building a key ring for a different user ID, such as a CICS region user ID.
3. If you have a signing certificate (certificate authority certificate), enter the label in the *certifier* variable.
4. If you do not have a signing certificate, replace the variables for the **RACDCERT CERTAUTH GENCERT** command with suitable values and RACF can create it for you:

```
"RACDCERT CERTAUTH GENCERT",
" SUBJECTSDN(CN('CICS Sample Certification Authority' ) ",
           "OU('"department"'                        ) ",
           "O ('"organization"'                      ) ",
           "L ('"city"'                              ) ",
           "SP('"state"'                             ) ",
           "C ('"country"'                          ))",
" NOTBEFORE(DATE("start")  TIME(00:00:00)             )",
" NOTAFTER (DATE("finish") TIME(23:59:59)             )",
" WITHLABEL("certifier"                               )",
" SIZE     (768                                       )"
```

These values define appropriate fields in the distinguished names of the generated certificates. *start* and *finish* determine the validity of the certificate. *certifier* is the label of the self-signed Certificate Authority certificate that is used to sign the other certificates. The **SIZE** parameter specifies the size, in bits, of the private key that is associated with the certificate. The larger the size, the more secure the key. The most common size is 1024.

DFH$RING creates the signing certificate only if it does not already exist.

5. Edit the variables for the **RADCERT GENCERT** RACF commands to create appropriate certificates to add to your key ring. DFH$RING has four examples that you can edit, add to, or remove. Ensure that the *certifier* variable on the **SIGNWITH** parameter matches the label of your signing certificate.
6. Edit the labels for the **RACDCERT CONNECT** RACF commands to match your certificates. Ensure that the signing certificate is added to the key ring first, because it signs all the other certificates.
7. Run DFH$RING to create the key ring and certificates as follows:

   ```
   EXEC 'CICSTS32.CICS.SDFHSAMP(DFH$RING)' 'firstname lastname webservername  [ FORUSER(use
   ```

   where *userid* is the CICS region user ID.

The DFH$RING program creates a key ring with name *firstname.lastname* which is owned by the *userid* user ID. Any existing key ring with that name is replaced. If you omit the **FORUSER** parameter, the key ring is owned by the user ID that you used to run the program. DFH$RING creates a signing certificate if required and adds it to the key ring, followed by the other certificates.

If you run DFH$RING with the default values, DFH$RING creates certificates with the following labels:

*lastname***-Web-Server**
: This certificate can be used in the CERTIFICATE attribute of TCPIPSERVICEs with PROTOCOL(HTTP). The distinguished name within the certificate has a common name of *webservername*, which must be the same as the host name associated with the connection. Web browsers usually check that the common name in the certificate matches the host name of the server from which it is received.

*lastname***-EJB-Container**
: This certificate can be used in the CERTIFICATE attribute of a CORBASERVER definition.

*lastname***-IP-CONNECTION**
: This certificate can be used for IP interconnectivity (IPIC). It can be used in CERTIFICATE attributes of resource definitions that are required for a CICS region to use IPIC. This sample certificate is for a CICS region to use as a client certificate and as a server certificate during an SSL handshake that occurs when an IPCONN is acquired. It can be used in the CERTIFICATE

attribute of an IPCONN definition for a client certificate and the CERTIFICATE attribute of a TCPIPSERVICE definition with PROTOCOL(IPIC) for a server certificate.

lastname-**Default-Certificate**
This certificate is marked as the default certificate for the key ring and is the one that is used for all TCPIPSERVICEs and CORBASERVERs that do not specify a CERTIFICATE attribute. This certificate also contains a common name of *webservername*.

Verisign Class 1 Primary CA

Verisign Class 2 Primary CA

IBM World Registry CA
These certificates are required to validate client certificates that you might receive that have been signed by these Certificate Authorities. If you intend to accept client certificates signed by other Certificate Authorities, or certificates that you have created yourself, you will have to add their certificates to the key ring manually, using the **RACDCERT CONNECT** command. When you add a certificate to the key ring in this way, you must specify USAGE(PERSONAL).

You can create and add further certificates to the key ring.

Creating new RACF certificates

Use the **RACDCERT** command to create and add new certificates to a key ring.

The certificates in the key ring must be associated with the CICS region user ID. The key ring must be owned by the CICS region user ID that is making use of it.

1. Create a certificate, specifying the CICS region user ID. Enter the **RACDCERT GENCERT** command as follows:

   ```
   RACDCERT ID(foruser) GENCERT
    SUBJECTSDN(CN('username')
             T ('username''s certificate')
             OU('department')
             O ('organization')
             L ('city')
             SP('state')
             C ('country'))
    NOTBEFORE(DATE(start) TIME(00:00:00))
    NOTAFTER (DATE(finish) TIME(23:59:59))
    SIGNWITH (CERTAUTH LABEL('certifier'))
    WITHLABEL('certlabel')
    SIZE     (1024)
   ```

   Provide values for the variables. The country code for the *country* variable must be an ISO 3166-1 code. For a list of valid codes, see http://www.iso.org/iso/country_codes/iso_3166_code_lists.htm. The value of *certifier* is the label of the signing certificate in the key ring.
2. Add the certificate to the key ring using the RACDCERT CONNECT command.
   a. If you want to share the certificate across multiple CICS regions, add it to the key ring specified in the **KEYRING** system initialization parameter for that CICS region and specify USAGE(PERSONAL). Any CICS region that has the same region user ID and is using the same key ring can access the certificate.

      ```
      RACDCERT ID(foruser) CONNECT( RING(ringname) LABEL('label') USAGE('PERSONAL'))
      ```

   b. If you want to add a certificate to the key ring as the default certificate, add it to the key ring specified in the **KEYRING** system initialization parameter for that CICS region and specify DEFAULT.

      ```
      RACDCERT ID(foruser) CONNECT( RING(ringname) LABEL('label') DEFAULT)
      ```

      When a client or server requests a certificate from CICS, the default certificate is used unless you have specified otherwise:

      - For inbound HTTP and IIOP requests, specify the certificate in the TCPIPSERVICE resource definition.
      - For outbound IIOP requests, specify the certificate in the CORBASERVER resource definition.

3. After running any of the RACDCERT commands that update certificates or key rings, if the DIGTCERT and DIGTRING classes are RACLISTed, you must issue the following command:

   ```
   SETROPTS RACLIST(DIGTCERT DIGTRING) REFRESH
   ```

4. After you have performed a key ring update, restart the CICS region to pick up the changes.

Associating a RACF user ID with a certificate

The client certificate can only be used to determine the user ID for the CICS transaction if the certificate is associated with a RACF user ID.

You can associate a certificate with a RACF user ID in two ways:

- Users can register their certificates online through their browser program. You enable clients to register their certificates themselves by specifying AUTHENTICATE(AUTOREGISTER) on the TCPIPSERVICE definition. Users connecting to CICS through such a TCPIPSERVICE must have a client certificate. If that certificate is already registered to a user ID, then that user ID is used; if not, the client is prompted for a user ID and password with HTTP basic authentication. If the client then enters a valid user ID and password, that user ID is registered to the certificate, and the client will not be prompted for a password again. The rules are summarized in “Identifying HTTP users” on page 236.

  Once a certificate has been registered in this way, it can be used for all inbound TCP/IP connections. For example, the same certificate can be used to authenticate and identify users of IIOP requests as well as HTTP requests.
- You can use the RACDCERT command. If you do not wish to allow your clients to register their own certificates, you must register them with the RACDCERT command. Before executing RACDCERT, you must download the certificate that you wish to process into an MVS sequential file with RECFM=VB that is accessible from TSO. The syntax of RACDCERT is:

  ```
  RACDCERT ADD('datasetname') TRUST [ ID(userid) ]
  ```

  where *datasetname* is the name of the dataset containing the client certificate, and *userid* is the user ID that is to be associated with the certificate. If the optional ID(userid) parameter is omitted, the certificate is associated with the user issuing the RACDCERT command.

You can add certificate information for your own user ID if you have READ access to the `IRR.DIGTCERT.ADD` profile in the FACILITY class. You can add certificate information for other user IDs if you have UPDATE access to the `IRR.DIGTCERT.ADD` profile in the FACILITY class or if you have RACF SPECIAL authority.

For further information on the RACDCERT command, including the format of data allowed in the downloaded certificate dataset, see *z/OS Security Server RACF Command Language Reference*

Marking a certificate untrusted

If a certificate has been registered in the RACF database, but you do not want it to be used by clients, you can mark it as UNTRUSTED using the RACDCERT command. To do this, first issue:

```
RACDCERT ID(userid) LIST
```

to find the label associated with the certificate you wish to change, and then issue:

```
RACDCERT ID (userid) ALTER(LABEL(label)) NOTRUST
```

to mark the certificate as untrusted. Clients are then prevented from establishing CLIENTAUTH connections with this certificate.

System initialization parameters for SSL

The following system initialization parameters relate to SSL:

CRLPROFILE
: Specifies the name of the profile that authorizes CICS to access certificate revocation lists that are stored in an LDAP server. For more information about certificate revocation lists and setting up this profile, see "Configuring an LDAP server for CRLs" on page 264.

ENCRYPTION
: Specifies the cipher suites that CICS uses for secure TCP/IP connections. When a secure connection is established between a pair of processes, the most secure cipher suite supported by both is used. For more information about cipher suites, see "Cipher suites" on page 247.

KEYRING
: Specifies the name of a key ring in the RACF database that contains keys and certificates used by CICS. It must be owned by the CICS region userid. You can create an initial key ring with the DFH$RING exec in CICS2T1.CICS.SDFHSAMP.

MAXSSLTCBS
: Specifies the maximum number of S8 TCBs that are available to CICS to
| process secure sockets layer connections and requests to LDAP using the
| DFHDDAPX XPI interface. This value is a number in the range 0 through 999, and has a default value of 8. The S8 TCBs are created and managed in the
| SSL pool. An S8 TCB is used by a task only for the duration of the SSL or
| LDAP processing. This parameter replaces the obsolete SSLTCBS system initialization parameter.

SSLCACHE
: Specifies whether CICS should use the local SSL cache in the CICS region, or share the cache across multiple CICS regions by using the coupling facility. Caching across a sysplex can only take place when the regions accept SSL connections at the same IP address.

SSLDELAY
: Specifies the length of time in seconds for which CICS retains session IDs for secure socket connections in a local CICS region. Session IDs are tokens that represent a secure connection between a client and an SSL server. While the

session ID is retained by CICS within the SSLDELAY period, CICS can continue to communicate with the client without the significant overhead of an SSL handshake. The value is a number of seconds in the range 0 through 86400. The default value is 600.

See *CICS System Definition Guide* for details of the system initialization parameters.

Defining TCPIPSERVICE resources for SSL

The following attributes of the TCPIPSERVICE resource relate to SSL:

AUTHENTICATE
specifies the authentication and identification scheme to be used for inbound TCP/IP connections for the HTTP and IIOP protocols. Each protocol supports a different set of authentication schemes.

When PROTOCOL(HTTP) is specified:

NO The client is not required to send authentication or identification information.

BASIC
HTTP basic authentication is used to obtain a user ID and password from the client.

CERTIFICATE
SSL client certificate authentication is used to authenticate and identify the client.

AUTOREGISTER
SSL client certificate authentication is used to authenticate the client. If the client sends a valid certificate that is not registered to the security manager, then CICS will register the certificate.

AUTOMATIC
If the client sends a certificate, SSL client certificate authentication is used to authenticate the client. If the client sends a valid certificate that is not registered to the security manager, then CICS will register the certificate. If the client does not send a certificate, then HTTP Basic authentication is used to obtain a user ID and password from the client.

When PROTOCOL(IIOP) is specified:

NO The client is not required to send authentication or identification information.

CERTIFICATE
SSL client certificate authentication is used to authenticate and identify the client.

ASSERTED
Asserted identity authentication is used to authenticate and identify the client.

CERTIFICATE
specifies the label of the server certificate used during the SSL handshake. If this attribute is omitted, the default certificate defined in the key ring for the CICS region user ID is used.

CIPHERS
specifies a string of up to 56 hexadecimal digits that is interpreted as a list of up

to 28 2-digit cipher suite codes. When you use CEDA is to define the resource, CICS automatically initializes the attribute with a default list of acceptable codes, depending on the level of encryption that is specified by the ENCRYPTION system initialization parameter.

- For ENCRYPTION=WEAK, the default value is `03060102`.
- For ENCRYPTION=MEDIUM, the initial value is `0903060102`.
- For ENCRYPTION=STRONG, the initial value is `050435363738392F303132330A1613100D0915120F0C03060201`.

You can reorder the cipher codes or remove them from the initial list. However, you cannot add cipher codes that are not in the default list for the specified encryption level. To reset the value to the default list of codes, delete all of the cipher suite codes: the field is automatically repopulated with the default list.

See the *CICS RACF Security Guide* for more information.

PORTNUMBER
specifies the number of the port on which CICS is to listen for incoming client requests. The well known ports for SSL services supported by CICS are:

443 HTTP with SSL

684 IIOP with SSL

SSL
specifies whether the TCP/IP service is to use SSL for encryption and authentication:

NO SSL is not to be used.

YES An SSL session is to be used; CICS will send a server certificate to the client.

CLIENTAUTH
An SSL session is to be used; CICS will send a server certificate to the client, and the client must send a client certificate to CICS.

Customizing encryption negotiations

You can select the cipher suites that are used in the encryption negotiation process to set a minimum level as well as a maximum level of encryption.

The CIPHERS attribute on the resource definitions TCPIPSERVICE, CORBASERVER and URIMAP specifies the cipher suites that can be used for each encryption level. The default value of the attribute is the list of 2-digit cipher codes that are used in encryption negotiations. You have the option of customizing this list of cipher suites to include your order of preference for the encryption levels at which CICS should negotiate with clients. You can also choose to remove cipher suites from the list. This is particularly useful if you want to ensure that only a very high level of encryption is used. You can do this as follows:

1. Select the resource definition that you want to change.
2. The CIPHERS attribute will display the default value. For example, if the system initialization parameter ENCRYPTION=STRONG, the default value is `0504352F0A0903060201`.
3. Edit the attribute value to remove and reorder the cipher suites. For example, you could specify `352F0A0504`.
4. Save the resource definition.

Specifying `352F0A0504` means that CICS will not negotiate below 128-bit encryption for connections using this resource. Each of the 2-digit codes in the attribute, for example 35, 2F, 0A and so on, refer to cipher suites that have at least a 128-bit encryption. CICS will start by trying to negotiate using the AES cipher suites 35 and 2F, because these are first in the list of cipher codes. If the client does not have this level of encryption, CICS will close the connection.

Note that you cannot include cipher suites that are not in the default values for that level of encryption. For example, if you have a MEDIUM level of encryption specified, you cannot add the AES cipher suites 35 and 2F to the CIPHERS attribute. For a complete list of cipher suites for each level of encryption, see The ENCRYPTION system initialization parameter.

Configuring LDAP for CICS use

You can use LDAP for storing CRLs (certificate revocation lists) or Basic Authentication credentials. When certificate revocation lists or credentials are stored in the LDAP server, you need to authorize CICS to access them.

Certificate revocation lists and passwords are stored in the LDAP server with an access class of *critical*, and can only be accessed by a user who has provided authentication credentials at LDAP bind time. These credentials are a user's distinguished name and an associated password. These details can be saved in a specialized profile in the LDAPBIND RACF class. To set up the profile, follow these steps:

1. The password that is used in the profile must be encrypted before it is stored in the RACF database. To do this, you need to store a password encryption key in the KEYSMSTR RACF class by issuing one of the following RACF commands:

   -
     ```
     RDEFINE KEYSMSTR LDAP.BINDPW.KEY OWNER(userid)
             SSIGNON(KEYENCRYPTED(keyvalue))
     ```

     Use this command when the password encryption key is stored by the integrated cryptographic service facility (ICSF).

   -
     ```
     RDEFINE KEYSMSTR LDAP.BINDPW.KEY OWNER(userid)
             SSIGNON(KEYMASKED(keymask))
     ```

     Use this command when ICSF is not active.
2. Create the profile using the following RACF command:

   ```
   RDEFINE LDAPBIND profile-name
                    PROXY(LDAPHOST(ldap-url)
                          BINDDN('ldap-distinguished-name')
                          BINDPW(password))
                    UACC(NONE)
   ```

   where:

   profile-name
   is the name of the RACF profile whose PROXY segment contains the following LDAP bind parameters

   ldap-url
   is a fully qualified URL of the LDAP server to be accessed. For example, `LDAP://WINMVS28.HURSLEY.IBM.COM:3389`.

   ldap-distinguished-name
   is the distinguished name of an LDAP user authorized to inquire on certificate revocation list attributes from the server. For example, `CN=LDAPADMIN`

password
is the password that authenticates the LDAP user. The password is case-sensitive.

3. Authorize each CICS region user ID to access appropriate bind credentials in the LDAPBIND class by issuing one or more commands of the following form:

```
PERMIT profile-name CLASS(LDAPBIND)
                    ACCESS(READ)
                    ID(region-userid)
```

Using certificate revocation lists (CRLs)

You can configure CICS to use certificate revocation lists (CRLs) to check the validity of client certificates being used in SSL negotiations.

To use certificate revocation lists, you must install and configure an LDAP server. Details on how to perform these tasks can be found in *z/OS V1R4.0 Security Server LDAP Server Admin and Use*.

A certificate revocation list details the revoked certificates from a certificate authority. Certificate authorities keep these lists in CRL repositories that are available on the World Wide Web and can be downloaded and stored in an LDAP server. To populate the LDAP server and update certificate revocation lists, use the CICS-supplied transaction CCRL. You also need to authorize CICS to access the LDAP server. To use CRLs, follow these steps:

Configuring an LDAP server for CRLs

To use certificate revocation lists (CRLs), you must have an LDAP server running. You will also need to perform some configuration steps before you download the CRLs.

If you need to install and configure an LDAP server, read the *z/OS V1R4 Security Server LDAP Server Admin and Use* manual.

1. Ensure that the LDAP server is running. The default started task name is LDAPSRV.
2. In the file system in `etc/ldap`, edit the configuration file slapd.conf as follows:
   a. Create an administrator distinguished name and password, by providing values for adminDN and adminPW. The CICS-supplied CCRL transaction requires this information to update the LDAP server with the certificate revocation lists.
   b. Create a suffix entry for every certificate authority that you want to download CRLs from using CCRL. For each suffix, use the syntax "O=*certificate authority*". The suffix is comprised of the Certificate Authority's distinguished name that contains the organization or "O=" keyword, together with any other keywords to the right of this. If the suffix contains any of the special characters `<,+;>\"` you must escape them by using **two** backslash characters. If you are using the z/OS LDAP server and the suffix contains any characters that are not in the required 1047 code page, the characters should be escaped by encoding them as the 3-digit octal number of their Unicode representation, preceded by an ampersand.

For example you could specify the following suffixes in the file slapd.conf:

```
suffix "O=CompanyName"
suffix "O=CompanyName plc"
suffix "O=CompanyName,L=CompanyLocation,ST=CompanyArea,C=CompanyCountry"
suffix "O=CompanyName\\, Inc."
suffix "O=CompanyName\\, Inc.,C=CompanyCountry"
```

When you have configured the LDAP server to include all of your certificate authorities, run the CCRL transaction. For details, see “Running the CCRL transaction.”

Running the CCRL transaction

The CICS-supplied transaction CCRL allows you to download and store certificate revocation lists (CRLs) that can be used in the SSL handshake to determine if client certificates are valid.

You need to configure an LDAP server to specify which certificate authorities you want to use and to create an administrator id and password. See “Configuring an LDAP server for CRLs” on page 264 for detailed instructions.

Certificate revocation lists are available from certificate authorities such as Verisign. They are kept in CRL repositories that are available on the World Wide Web and can be downloaded and stored in an LDAP server. To populate the LDAP server and update certificate revocation lists, use the CICS-supplied transaction CCRL. You can run the CCRL transaction from a terminal or using a **START** command. Use the **START** command to schedule regular updates.

1. Specify the name of the profile that authorizes CICS to access the LDAP server in the **CRLPROFILE** system initialization parameter.
2. Run the CCRL transaction using either of the following methods:
   - Run the transaction from a terminal. See “Running CCRL from a terminal”.
   - Run the transaction using a command. See “Running CCRL from a START command” on page 266.

Running CCRL from a terminal

You can run the CICS-supplied transaction CCRL using a terminal to download certificate revocation lists (CRLs).

Read “Running the CCRL transaction” to find out about the prerequisites before running this transaction from a terminal.

1. From a terminal, enter the command `CEOT TRANIDONLY` so that you can enter the list of URLs in mixed case.
2. Enter `CCRL` *`url-list`*, where *url-list* is the URL that specifies the location of the certificate revocation list file that you want to download. You can specify more than one URL by leaving a space between each URL in the list. For example, you could specify: `CCRL http://crl.verisign.com/ATTClass1Individual.crl http://crl.verisign.com/ATTClass2Individual.crl`.
3. You are prompted to enter the administrator distinguished name and password for the LDAP server. This allows CICS to update the LDAP server with the CRLs that it downloads. The administrator name and password are specified in the file slapd.conf. For more information about configuring this file, see “Configuring an LDAP server for CRLs” on page 264

CICS downloads the CRLs from the URLs that you have specified and store them in the LDAP server. You will receive confirmation that all of the lists were downloaded. If CICS experiences a problem, for example the URL is not valid, you will receive an error message.

To set up regular updates, you can use a START command. See “Running CCRL from a START command.”

Running CCRL from a START command

You can schedule the CCRL transaction to run at regular intervals using a START command.

Read “Running the CCRL transaction” on page 265 to find out about the prerequisites before running this transaction from a terminal.

To use a START command, enter `EXEC CICS START TRANSID(CCRL) FROM (admin://`*`adminDN:adminPW url-list`*`) LENGTH (`*`url-list-length`*`) [INTERVAL(`*`hhmmss`*`)|TIME(`*`hhmmss`*`)]` where *url-list* is a space-delimited list of URLs from where certificate revocation lists can be downloaded, *url-list-length* is the length of the URL list (including `admin://`), and *hhmmss* is the interval or expiration time at which the CCRL transaction is to be scheduled. For example, you could specify:

```
EXEC CICS START TRANSID(CCRL)
          FROM('admin://cn=ldapadmin:cics31ldap http://crl.verisign.com/ATTClass1Individual.crl htt
         LENGTH(124) INTERVAL(960000)
```

This would schedule the CCRL transaction to run in 96 hours.

Part 5. Security for enterprise beans

This describes how you can use RACF to provide security for enterprise beans.

Chapter 20. Managing security for enterprise beans

The following security mechanisms can be used with enterprise beans. You can implement any combination of these.

Java2 security
: This form of security control is implemented by the Java Virtual Machine (JVM) and can be used with any Java program that executes under JVM control. See "Protecting Java applications in CICS by using the Java 2 security policy mechanism" for guidance on using this type of security control.

Secure Sockets Layer (SSL) security
: The Secure Sockets Layer (SSL) is a security protocol that provides privacy and authentication between clients and servers communicating using TCP/IP. For more information about SSL, see Support for security protocols, in the *CICS RACF Security Guide*.

MRO security
: After the request receiver has established a CICS USERID to be associated with the request, it may need to be routed to an application-owning-region (AOR). If the routing mechanism uses a multiple region operation (MRO) connection, the transmission of the userid is subject to MRO security rules. See Link security with MRO, in the *CICS RACF Security Guide*.

Security roles
: A security role represents a type of user of an application in terms of the permissions that the user must have to successfully use the application. See "Security roles" on page 277.

Protecting Java applications in CICS by using the Java 2 security policy mechanism

The security of the enterprise beans container environment is protected by the Java 2 security policy mechanism and is independent of CICS security. The security policy mechanism is one of the components that make up the Java 2 security model. The security policy mechanism is used to enforce the restrictions in the EJB specification concerning Java functions that may not be issued by enterprise beans.

By default, Java applications have no security restrictions placed on activities requested of the Java API; the Java API will do whatever it is asked. If you want to use Java 2 security to protect a Java application or enterprise bean from performing potentially unsafe actions, you need to enable a security manager for the Java virtual machine (JVM) in which the application or enterprise bean executes. If no security manager is enabled, then by default, the JVM runs without Java 2 security. A default security manager is supplied with the Java 2 platform. To prevent unauthorized access to system resources by enterprise beans, you are recommended to enable the default security manager.

The security manager enforces a security policy, which is a set of permissions (system access privileges) which are assigned to code sources. Every time the JVM executes code within a class, the JVM determines the code source for the class and consults the security policy before granting the class the appropriate permissions. Thus, if a piece of code requests access to a particular system resource while a security manager is active, the JVM grants the code access to that resource only if such an access is a privilege associated with that class.

When a JVM starts up, its security manager determines the security policy for the JVM by looking at one or more **policy files** that you have specified. The policy files contain details of the permissions that are granted to particular code sources. A default policy file is supplied with the Java 2 platform. If you enable the default security manager for a JVM, but do not specify any policy files, the security manager determines a security policy using the permissions given in the default policy file. You can specify one or more additional policy files containing permissions that you want to grant, and the security manager adds these permissions to the security policy. So although only one security policy is in effect for the JVM at any given time, this security policy can be the result of processing one or more policy files.

To enable Java applications and enterprise beans to run successfully in CICS when Java 2 security is active, you need to specify, as a minimum, an additional policy file that gives CICS the permissions it needs to run the enterprise beans container, and gives applications the permissions outlined in the *Enterprise JavaBeans* specification, Version 1. The CICS-supplied enterprise beans policy file, `dfjejbpl.policy`, contains the permissions that you need for this purpose. You need to specify this additional policy file for each kind of JVM that has a security manager enabled.

You enable the security manager for a JVM, and specify additional policy files, using the JVM properties file for the JVM. “Enabling a Java security manager and specifying policy files for a JVM” tells you how to do this.

If you need more information about Java 2 security than is provided here, refer to the Java 2 documentation.

Note: Java 2 security with JDBC or SQLJ

To use JDBC or SQLJ from enterprise beans that execute in a JVM with a Java 2 security policy mechanism active, you must use the JDBC 2.0 driver provided by DB2 Version 7 or later. The JDBC 1.2 driver provided by DB2 does not support Java 2 security, and will fail with a security exception unless you disable the mechanism (by deactivating the security manager for the JVM). You will also need to modify your additional policy file to grant permissions to the JDBC driver. “Enabling a Java security manager and specifying policy files for a JVM” tells you more about this.

Enabling a Java security manager and specifying policy files for a JVM

To enable a Java security manager for a JVM and specify additional policy files that you want the security manager to use, you need to customize the JVM properties file for the JVM.

The JVM properties file specifies the system properties for a JVM, including the security manager and policy files. It is associated with the JVM profile for a JVM. Setting up JVM profiles and JVM properties files"Setting up JVM profiles and JVM properties files" in *Java Applications in CICS* explains what JVM profiles and JVM properties files are, and how to choose and customize JVM profiles and JVM properties files for a JVM.

For each JVM profile that your Java applications and enterprise beans request, if you want JVMs with that profile to run with Java 2 security, you need to modify the JVM properties file that is associated with the JVM profile, to enable the default security manager and specify a suitable policy file. When you have located the relevant JVM properties file for each JVM profile that you want to use Java 2 security, customize the following system properties in the JVM properties file:

-Djava.security.manager
This system property indicates the Java security manager to be enabled for the JVM. To enable the default Java 2 security manager, include this system property in one of the following formats:

```
-Djava.security.manager=default
```

or

```
-Djava.security.manager=""
```

or

```
-Djava.security.manager=
```

All these statements have the effect of enabling the default security manager. If you do not include the **-Djava.security.manager** system property in your JVM properties file, then the JVM runs without Java 2 security enabled. If you need to disable Java 2 security for a JVM, comment out this system property.

-Djava.security.policy
This system property describes the location of additional policy files that you want the security manager to use to determine the security policy for the JVM. A default policy file is provided with the JVM in `/usr/lpp/java142/J1.4/lib/security/java.policy`, where `java142/J1.4` is your install location for the IBM SDK for z/OS, Java 2 Technology Edition. The default security manager always uses this default policy file to determine the security policy for the JVM, and you can use the **-Djava.security.policy** system property to specify any additional policy files that you want the security manager to take into account as well as the default policy file.

To enable CICS Java applications and enterprise beans to run successfully when Java 2 security is active, you need to specify, as a minimum, an additional policy file that gives CICS the permissions it needs to run the enterprise beans container, and gives applications the permissions outlined in the *Enterprise JavaBeans* specification, Version 1. If you do not provide these permissions, then the container code may become inaccessible, preventing CorbaServers from being initialized. The CICS -supplied enterprise beans policy file, `dfjejbpl.policy`, contains the permissions that you need. To specify this policy file, include the system property:

```
-Djava.security.policy=/usr/lpp/cicsts/cicsts32/lib/security/dfjejbpl.policy
```

where *cicsts32* is your chosen value for the USSDIR installation parameter that you defined when you installed CICS TS. “The CICS-supplied enterprise beans policy file, dfjejbpl.policy” on page 273 has more information about `dfjejbpl.policy`.

If you need to give any of your applications further permissions, you can modify the CICS-supplied enterprise beans policy file, or create and specify your own additional policy file. Policy files are stored in text format, so you can display or modify them using any standard text editing tool. In particular, if you want to use JDBC or SQLJ from enterprise beans, you need to modify the enterprise beans policy file that you have specified, to grant permissions to the JDBC driver. Requirements to support Java programs in the CICS DB2 environment, in the *CICS DB2 Guide*, tells you how to do this.

It is recommended that policy files are made secure, with update authority restricted to system administrators.

When you specify a policy file in the JVM properties file, the policy file is used for JVMs that are built using JVM profiles which reference that JVM properties file. As an alternative, you can specify a policy file to be used for **all** the JVMs in your system for which you have enabled a Java security manager, whatever JVM properties file they have. For example, you could specify the CICS-supplied enterprise beans policy file, `dfjejbpl.policy`, to be used for all your JVMs. To do this, instead of including the **-Djava.security.policy** system property in the JVM properties file, use the alternative method described in “Specifying policy files to apply to all JVMs.” If you specify a policy file to be used for all JVMs, remember that to activate Java 2 security for your JVMs, you still need to add the **-Djava.security.manager** system property to your JVM properties files to enable a Java security manager.

Specifying policy files to apply to all JVMs

As an alternative to using the **-Djava.security.policy** system property in a JVM properties file to specify additional policy files, you can name the additional policy files in the JVM default **security properties file**, which applies to all JVMs. This file is where the default Java 2 security manager looks for the name of the default policy file, which it always uses to determine the security policy for a JVM.

The default security properties file is called `java.security`. It is provided by CICS in:

```
/usr/lpp/java142/J1.4/lib/security/java.security
```

where the `java142/J1.4` subdirectory names are your install location for the IBM SDK for z/OS, Java 2 Technology Edition on z/OS UNIX.

The default security properties file already includes the name of the default policy file, `/usr/lpp/java142/J1.4/lib/security/java.policy`. You can add the names of additional policy files, and the security manager will then use these files, as well as the default policy file, to determine the security policy for **all** JVMs. The security manager will also refer to any policy files that you have specified in the JVM properties file for a particular type of JVM.

In the default security properties file `java.security`, policy files are specified in the form:

```
policy.url.n=URL
```

where `n` represents the precedence number for the order in which the policies should be loaded. The location of a policy file is specified as a URL, so policy files do not need to be stored in the local file system.

Note that the precedence numbers must be serial and continuous. For example, if `policy.url.1` and `policy.url.3`, are present, but `policy.url.2` is missing, then `policy.url.3` is ignored and only `policy.url.1` is considered.

The default security properties file `java.security` contains these two entries:

```
policy.url.1=file:${java.home}/lib/security/java.policy
policy.url.2=file:${user.home}/.java.policy
```

To specify the CICS-supplied enterprise beans policy file, `dfjejbpl.policy`, as an additional policy file to be used for all JVMs, add the entry:

```
policy.url.3=file:/usr/lpp/cicsts/cicsts32/lib/security/dfejbpl.policy
```

where *cicsts32* is your chosen value for the USSDIR installation parameter that you defined when you installed CICS TS. It is specified as `policy.url.3` because two other policy files are already specified. You can substitute the path to your own policy file in place of `dfjejbpl.policy`, or add further entries to specify additional policy files.

It is possible to bypass the default security properties file `java.security` for a JVM. You can do this by specifying your own policy file on the **-Djava.security.policy** system property in the JVM properties file for the JVM, and inserting a double equals sign (= =). For example, if you include the system property:

```
-Djava.security.policy==/usr/lpp/cicsts/cicsts32/lib/security/dfejbpl.policy
```

then the security manager ignores any policy files that are specified in the `java.security` file, and uses only `dfjejbpl.policy` to determine the security policy for the JVM. However, you should bear in mind that if you bypass the default security properties file, the security manager will not grant any permissions that are specified in that file; it will only grant the permissions that are specified in your own policy file.

The CICS-supplied enterprise beans policy file, dfjejbpl.policy

The CICS-supplied enterprise beans policy file, `dfjejbpl.policy`, is based on the security policy recommended in the Sun Microsystems *Enterprise JavaBeans Specification, Version 1.1*, which is available at `http://www.javasoft.com/products/ejb`. The sample policy file is shown in Figure 26 on page 274.

In Java 2, the security policy is defined in terms of protection domains which map permissions to code sources. A protection domain contains a code source with a set of associated permissions.

The CICS-supplied enterprise beans policy file defines two protection domains, which do the following:

1. Grants the required permissions to the CICS enterprise beans Container code source for execution. See the 'grant codeBase' block in Figure 26 on page 274.
2. Grants any code source only the permissions outlined in the *Enterprise JavaBeans* specification, Version 1. See the default 'grant' block in Figure 26 on page 274:
   - To allow anyone to initiate a print job request.
   - To allow outbound connection on any TCP/IP ports.
   - To allow all system properties to be read.

Remember that if you want to use JDBC or SQLJ from enterprise beans, you need to amend the CICS-supplied enterprise beans policy file to grant permissions to the JDBC driver. Requirements to support Java programs in the CICS DB2 environment, in the *CICS DB2 Guide*, tells you how to do this.

```
 // permissions granted to CICS enterprise beans Container codesource protection
 //domain
    grant codeBase "file:usr/lpp/cicsts/cicsts32//-" {
      permission java.security.AllPermission;
    };

// default EJB 1.1 permissions granted to all protection domains
    grant {
      // allows anyone to initiate a print job request
      permission java.lang.RuntimePermission "queuePrintJob";

      // allows outbound connection on any TCP/IP ports
      permission java.net.SocketPermission "*:0-65535", "connect";

      // allows anyone to read properties
      permission java.util.PropertyPermission "*", "read";
    };
```

Figure 26. Sample CICS enterprise beans security policy

Using enterprise bean security

The EJB 1.1 specification defines the following security APIs to allow enterprise beans to make application decisions based on their callers' security details.

java.security.Principal getCallerPrincipal()
This method is used to determine who invoked the current bean method. The getCallerPrincipal method is fully supported in CICS. Details of the way that the identity of the current caller is determined are shown in "Deriving distinguished names" on page 276.

boolean isCallerInRole(String SecurityRoleReference)
This method is used to test whether the current caller is assigned to a security role that is linked to the security role reference specified on the method call.

CICS will throw a runtime exception (which conforms to the EJB 1.1 specification) if the following deprecated EJB 1.0 security APIs are used.

- java.security.Identity getCallerIdentity()
- boolean isCallerInRole(java.security.Identity role)

Note: Note that enterprise beans developed to the Enterprise JavaBeans (EJB) 1.0 specification need to be migrated to the Enterprise JavaBeans 1.1 specification level, using the supplied development tools.
- See The deployment tools for enterprise beans in a CICS system for information about deployment tools.
- See Writing enterprise beans for information about writing enterprise beans.
- See *Java Applications in CICS* for more information.
- See The deployment tools for enterprise beans in a CICS system for information about deployment tools.
- See Writing enterprise beans for information about writing enterprise beans.

Defining file access permissions for enterprise beans

To successfully run enterprise beans in CICS, the CICS region userid must be permitted to access the files used by the enterprise logic. These file permissions are required to run enterprise beans, regardless of the level of security implemented. See also the *CICS Transaction Server for z/OS Installation Guide*.

Access to z/OS UNIX files used by enterprise beans

Table 35. File access permissions required for CICS enterprise beans

File/Directory structure	Minimum permission	Comments
CORBASERVER Shelf directory (for example, `/var/cicsts/`)	Read, write and execute	The shelf is accessed during CORBASERVER and DJAR installation, and each CICS needs to create unique subdirectories (see note 1).
`/usr/lpp/cicsts/`*`cicsts32`* directory structure and classes	Read and execute	Contains the CICS-supplied Java code (see note 2).
`/usr/lpp/java142/J1.4/bin` and `/usr/lpp/java142/J1.4/bin/classic` directories	Read and execute	Contain the IBM JVM code (see note 3).
CICS working directory	Read, write and execute	Used to create `stdin` files (see note 4).
Deployed jar file	Read	Used during DJAR installation by the deployment process.
Security policy file (if required)	Read	Required if the `-Djava.security.policy` property is specified in the JVM system properties file.
System properties file	Read	Required by CICS when creating a JVM (see note 5).

Note:

1. `/var/cicsts/` is the default SHELF directory name when you define a CORBASERVER resource definition. Each CICS region creates a unique subdirectory in this shelf when it installs the resource definition
2. *cicsts32* is your chosen value for the USSDIR installation parameter that you defined when you installed CICS TS.
3. `java142/J1.4` is your install location for the IBM SDK for z/OS, Java 2 Technology Edition.
4. The CICS working directory is defined by the WORK_DIR parameter in the JVM profile.
5. The system properties directory and file name are named on the JVMPROPS option in the JVM profile.

File ownership and permissions may be defined using the **chmod** and **chown** commands. For more information, see *z/OS UNIX System Services Command Reference*.

Access to data sets used by enterprise beans

Before CORBASERVERs can be installed in a CICS region, the following two data sets must be created with UPDATE access, defined to CICS and installed. These files can be VSAM data sets or coupling facility data tables.

Figure 27 on page 276 shows an example of RACF commands to access data sets with the necessary authorization.

Note: These files are used internally by CICS, so no users should be given resource level security access to them. This will prevent VSAM applications from accessing the data in these files.

DFHEJDIR
This data set contains a request streams directory which is shared by the listener regions and AORs comprising a CICS IIOP server. The file must be recoverable.

DFHEJOS
DFHEJOS is a data set containing passivated stateful session beans. It is shared by all the AORs comprising a CICS IIOP server. This file must not be recoverable.

```
ADDSD  'CICSTS32.CICS.CICS.DFHEJDIR' NOTIFY(cics_sys_admin_id) UACC(NONE)
PERMIT 'CICSTS32.CICS.CICS.DFHEJDIR' ID(cics_id1,...,cics_group1,..,cics_groupn)
       ACCESS(UPDATE)
ADDSD  'CICSTS32.CICS.CICS.DFHEJOS'  NOTIFY(cics_sys_admin_id) UACC(NONE)
PERMIT 'CICSTS32.CICS.CICS.DFHEJOS'  ID(cics_id1,...,cics_group1,..,cics_groupn)
       ACCESS(UPDATE)
```

Figure 27. An example of RACF commands used to authorize access to CICS data sets

See Authorizing access to CICS data sets, in the *CICS RACF Security Guide*, for more information about authorizing access to CICS data sets.

Deriving distinguished names

Enterprise beans can identify their end-user, or client, by means of a *Principal* object. The *getCallerPrincipal* method returns a *Principal* object representing the client, and that *Principal* object contains methods that can be invoked to return information about the client. In particular, the *getName* method of the *Principal* object returns a *String* that contains the "distinguished name" of the client. The distinguished name, or DN, is a sequence of keyword and value pairs, known as relative distinguished names, or RDNs, and forms part of the X.500 recommendation (Standard ISO/IEC 9594). The string representation of a distinguished name is suggested by RFC2253, *LDAP V3: UTF-8 String Representation of Distinguished Names.*

Note: CICS Transaction Server for z/OS, Version 3 Release 2 does not verify that a stateful session bean instance is used only by the same principal that created it. Therefore the principal's userid and distinguished name may be different after a bean instance has been reactivated.

If the bean's client has been identified and authenticated by means of a client certificate using the secure sockets layer protocol, the distinguished name is always obtained from that certificate. However, if the bean's client has not provided a certificate, the distinguished name is obtained by invoking the DFHEJDNX user-replaceable module. The inputs to the DFHEJDNX module are the title, organizational unit, organization, locality, state, and country, obtained from the server certificate whose label is specified in the CERTIFICATE option of the CORBASERVER definition, and the userid and common name associated with the user ID of the user executing the bean, but if SEC=NO is specified, the CICS region userid is used. The common name is derived by transforming the username for that user to a mixed-case string.) The certificate label specifies a certificate within the key ring identified by the KEYRING system initialization parameter. If the CERTIFICATE option is omitted, information is obtained from the default certificate in the key ring. If the KEYRING parameter is omitted, no certificate information is passed to DFHEJDNX, and only the common name RDN is available.

The CICS-supplied version of DFHEJDNX accepts the inputs derived from the CORBASERVER certificate and the username, and formats them into a distinguished name in the following style:

T=CICS EJB Container,CN=Louise Peters,OU=CICS/390 Development, O=IBM,L=Hursley,ST=Hampshire,C=GB

CICS-supplied samples of DFHEJDNX are located in the SDFHSAMP library, *CICSTS32.CICS.CICS.SDFHSAMP*, as:
- DFHEJDN1 for Assembler language
- DFHEJDN2 for C language

Security roles

Access to enterprise bean methods is based on the concept of **security roles**. A security role represents a type of user of an application in terms of the permissions that the user must have to successfully use the application. For example, in a payroll application:
- A `manager` role could represent users who are permitted to use all parts of the application
- A `team_leader` role could represent users who are permitted to use the administration functions of the application
- A `data_entry` role could represent users who are permitted to use the data entry functions of the application

The security roles for an application are defined by the application assembler, and are specified in the bean's deployment descriptor. For more information, see “Security roles in the deployment descriptor” on page 281

The security roles that are permitted to execute a bean method are also specified in the bean's deployment descriptor, again by the application assembler. In the example, methods which update the hours worked by employees each week might be assigned to the `data_entry` role, while methods which delete an employee from the payroll might be assigned to the `team_leader` role.

To distinguish similarly named security roles in different applications, or in different systems, the security roles specified in the bean's deployment descriptor can be given a one- or two-part qualifier when the bean is deployed in a CICS system. For example:
- Security role with no qualifiers:
  `team_leader`
- Security role with one qualifier:
  `payroll.team_leader`
- Security role with two qualifiers:
  `test.payroll.team_leader`

A security role with its qualifiers is known as a **deployed security role**. For more information, see “Deployed security roles” on page 278.

The mapping of security roles to individual users is done in the external security manager. The mapping is not necessarily one-to-one. For example, several users might be assigned to the `data_entry` role, while a some users might be assigned to both the `team_leader` role and the `data_entry` role. For more information, see "Implementing security roles" on page 283.

The security role and display name in the deployment descriptor can contain any ASCII or Unicode character. This is not so for names used in RACF, which are restricted to characters in EBCDIC code page 037. In addition, some characters — the asterisk (*) for example — have special meaning when used in RACF commands. Therefore, when CICS constructs the deployed security role from its components, some characters are replaced with a different character, and others are replaced with an escape sequence. For details, see "Character substitution in deployed security roles" on page 279.

Deployed security roles

A direct mapping between the security roles specified in a bean's deployment descriptor and individual users may not adequately control access to bean methods. For example

- Two applications, provided by different suppliers, might use similar names for security roles. In your enterprise, the users of each application might be different.
- A bean could be used in more than one application. A user may be entitled to use a particular method in one application, but not in the other.
- An application could be deployed in a test system and a production system. Members of the test department may be permitted to use all bean methods in the test system, but not in the production system.

To provide the degree of control that is needed in these and other cases, you can qualify the security roles at the application level and the system level. A security role with its qualifiers is known as a **deployed security role**. Here is an example of a role name which is qualified at both levels:

```
test.payroll.team_leader
```

- `payroll` qualifies the security role at the application level, and is used to distinguish between the `team_leader` role in the payroll application and the `team_leader` role in other applications.
- `test` qualifies the security role at the system level, and is used to distinguish between the `payroll.team_leader` role in the test system and the `payroll.team_leader` role in other systems.

At the application level, security roles are qualified by the **display name**, if one is specified in the deployment descriptor. If a display name is not specified, the security roles are not qualified at the application level. If an application level qualifier is used, a period (.) is used as the delimiter; if no qualifier is used, there is no delimiter.

At the system level, security roles are optionally qualified with a prefix which is specified in the EJBROLEPRFX system initialization parameter. If EJBROLEPRFX is not specified, the security roles are not qualified at the system level. If a system level qualifier is used, a period (.) is used as the delimiter; if no qualifier is used, there is no delimiter.

This example shows how security roles defined in a bean's deployment descriptor can be qualified:

- A bean contains three security roles: `manager`, `team_leader`, and `data_entry`
- The bean is used in a payroll application, with a display name of `payroll`. The bean is also part of a test application, which does not have a display name.

- The payroll application is used on two production systems: the first does not specify a prefix, while the second specifies a prefix of `executive`.
- The test application is used on a test system with a prefix of `test1`.

When the two levels of qualification are applied to the security roles specified in the deployment descriptor, the deployed security roles are:

```
payroll.manager       executive.payroll.manager       test1.manager
payroll.team_leader   executive.payroll.team_leader   test1.team_leader
payroll.data_entry    executive.payroll.data_entry    test1.data_entry
```

Each of these deployed roles can be mapped to individual users (or groups of users) to suit the security need of the enterprise.

If a security role is not qualified at the application level, or at the system level, then the deployed security role is the same as the security role defined in the deployment descriptor. For example, if the bean in the previous example is used in an application which does not have a display name, and the application is used in a system that does not specify EJBROLEPRFX, then the deployed security roles are:

```
manager
team_leader
data_entry
```

Enabling and disabling support for security roles

By default, CICS support for security roles is enabled. You can use the XEJB system initialization parameter to disable (or explicitly enable) support for security roles. If you disable the support:

- CICS does not perform method authorization checks: all users are permitted to use all bean methods.
- The `isCallerInRole()` method returns `true` for all users.

Security role references

Within an application, the `isCallerInRole()` method can be used to determine if the user of the application is defined to a given role. The method takes a **security role reference** as an argument, rather than a security role. The security role references coded in the bean are defined by the bean provider, and declared in the bean's deployment descriptor.

For more information, see “Security roles in the deployment descriptor” on page 281

Each security role reference is linked to a security role by the application assembler; the linkage is declared in the deployment descriptor for the bean. For example, the security role reference of `administrator` used within the bean's code might be linked, in the deployment descriptor, to the `team_leader` role.

For more information, see “Security roles in the deployment descriptor” on page 281

Character substitution in deployed security roles

The security role and display name in the deployment descriptor can contain any ASCII or Unicode character. The character set which can be used in deployed security roles is more restricted:

- Profile names used in RACF are restricted to characters in EBCDIC code page 037.

- Some characters — the asterisk (`*`) for example — have special meaning when used in RACF commands, and cannot be used in a profile name.

When Unicode characters in the security role and display name cannot be used directly in the deployed security role, they are replaced by the escape sequences shown in Table 36. Substitution occurs:

- when the EJBROLE generator utility (`dfhreg`) processes the deployment descriptor to generate RACF commands
- when CICS maps a security role to a RACF user ID

Table 36. Escape sequences used in security roles

Character	Description	ASCII/Unicode	EBCDIC code page 037	Escape sequence
ASCII and Unicode values whose equivalent EBCDIC value cannot be used in a deployed security role name are replaced with a three-character escape sequence as follows:				
	blank	X'20'	X'40'	¢
¢	cent	X'A2'	X'4A'	\A2
\	back slash	X'5C'	X'E0'	\5C
*	asterisk	X'2A'	X'5C'	\2A
&	ampersand	X'26'	X'50'	\26
%	per cent	X'25'	X'6C'	\25
,	comma	X'2C'	X'6B'	\2C
(	left parenthesis	X'28'	X'4D'	\28
)	right parenthesis	X'29'	X'5D'	\29
;	semicolon	X'3B'	X'5E'	\3B
Unicode values which do not have an equivalent in EBCDIC code page 037 are replaced with the Unicode escape sequence: a character with a Unicode representation of X'yyyy' is replaced by `\uyyyy`. For example:				
€	Euro symbol	X'20AC'	not supported	\u20AC
	Hiragana Ki	X'304D'	not supported	\u304D
α	alpha	X'03B1'	not supported	\u03B1

Here are two examples that illustrate the way that characters are substituted:

Example 1

- The EJBROLEPRFX has a value of `test`
- The display name in the deployment descriptor has a value of `year.end.processing`
- The security role in the deployment descriptor has a value of `auditor 1`

In this example, when the deployed security role is constructed:

1. Each space is replaced with ¢
2. The deployed security role is composed from the EJBROLEPRFX value, the display name, and the security role; a period is used as the delimiter.

The resulting deployed security role is:

```
test.year.end.processing.auditor¢1
```

Example 2

- The EJBROLEPRFX has a value of `test`

- The display name in the deployment descriptor has a value of `αβ32`. The Unicode encoding is X'03B1 03B2 0033 0034'.
- The security role in the deployment descriptor has a value of `auditor 1`

In this example, when the deployed security role is constructed:

1. Each Unicode character that has an equivalent in EBCDIC code page 037 is replaced accordingly: In the display name, X'0033 0034' is replaced by `34`.
2. Each Unicode character that does ***not*** have an equivalent in EBCDIC code page 037 is replaced with the corresponding escape sequence. In the display name, X'03B1 03B2' is replaced by `\u03B1\u03B2`
3. Each space is replaced with `¢`
4. The deployed security role is composed from the EJBROLEPRFX value, the display name, and the security role; a period is used as the delimiter.

The resulting deployed security role is:

```
test.\u03B1\u03B234.auditor¢1
```

Security roles in the deployment descriptor

Figure 28 on page 282 shows a fragment of a deployment descriptor that includes security role information. It contains:

- 1 A display name of `payroll`.
- 2 The security role reference of `administrator` which is linked to the `team_leader` role.
- 3 A security role of `team_leader`.
- 4 A method permission that allows a user defined in the `team_leader` role to invoke the `create()` method.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE ejb-jar PUBLIC
"-//Sun Microsystems, Inc.//DTD Enterprise JavaBeans 1.1//EN"
"http://java.sun.com/j2ee/dtds/ejb-jar_1_1.dtd">
  <ejb-jar id="ejb-jar_ID">
    <display-name>payroll</display-name>         1
      <enterprise-beans>
        <session id="Session_1">
          .
          .
          <security-role-ref id="SecurityRoleRef_1">
            <role-name>administrator</role-name> 2
            <role-link>team_leader</role-link>
          </security-role-ref>
          .
          .
        </session>
      </enterprise-beans>
      <assembly-descriptor id="AssemblyDescriptor_1">
        <security-role id="SecurityRole_1">
          <role-name>team_leader</role-name>     3
        </security-role>
        .
        .
        <method-permission id="MethodPermission_1">
          <description>team_leader:+:</description>
          <role-name>team_leader</role-name>     4
          <method id="MethodElement_01">
            <ejb-name>Managed</ejb-name>
            <method-intf>Home</method-intf>
            <method-name>create</method-name>
            <method-params>
            </method-params>
          </method>
           .
           .
        </method-permission>
         .
         .
      </assembly-descriptor>
       .
       .
     </ejb-jar>
```

Figure 28. Example of a deployment descriptor containing security roles

If an application with this deployment descriptor is used in a CICS system with the following system initialisation parameters:

```
SEC=YES
XEJB=YES
EJBROLEPRFX='test'
```

- The deployed security role of `test.payroll.team_leader` must be defined to RACF.
- Users that have READ access to that deployed security role will be permitted to invoke the `create()` method.
- `isCallerInRole('administrator')` will return `true` for users defined in the deployed security role of `test.payroll.team_leader`, and `false` for other users.

For detailed information about the contents of the deployment descriptor, refer to *Enterprise JavaBeans Specification, Version 1.1*.

To view the contents of a deployment descriptor, you can use the Assembly Toolkit (ATK). For more information about ATK, see The enterprise bean deployment tool, ATK, in the *CICS Operations and Utilities Guide*.

Implementing security roles

Access to enterprise bean methods is based on the concept of **security roles**. These are described in “Security roles” on page 277.

To implement the use of security roles in a CICS enterprise bean environment, you must:

1. Determine which security roles are defined in the application's deployment descriptor.
2. Determine the display names associated with the security roles in the application's deployment descriptor. The display name qualifies the security role at the application level.
3. Decide whether you need to qualify the security role name at the system level, and — if you do — the value of the prefix which you will use in each system where the application executes.
4. Using the information gathered in steps 1 through 3, determine the names of the deployed security roles used by the application in each system. Characters in the security role and display name that do not have a direct equivalent in EBCDIC code page 37 (and some other characters) must be replaced with a different character or an escape sequence when constructing the deployed security role. See “Character substitution in deployed security roles” on page 279 for more information.
5. Using the information gathered in steps 1 through 3, define RACF profiles for the deployed security roles. See the *Java Applications in CICS* manual“Defining security roles to RACF” on page 285 for more information.
6. Associate individual users or groups of users with each deployed security role in RACF. See the *Java Applications in CICS* manual“Defining security roles to RACF” on page 285 for more information.
7. Specify these system initialization parameters:
   - `SEC=YES`
   - `XEJB=YES`. This is the default value, so you do not need to specify it explicitly.
8. For those systems where the deployed security roles contain a system level qualifier (see step 3), specify the EJBROLEPRFXEJBROLEPRFX system initialization parameter.

Using the RACF EJBROLE generator utility

The RACF EJBROLE generator utility (`dfhreg`) is a Java application program that extracts security role information from deployment descriptors, and generates a REXX program which can be used to define security roles to RACF .

The REXX program that `dfhreg` generates contains the RACF commands that define security roles as members of a profile in the GEJBROLE class. Before you run the REXX program, you will need to modify it, in order to change the name of the profile that is defined.

The `dfhreg` invocation scripts for USS (`dfhreg`) and for Windows (`dfhreg.bat`) are in the `CICS_HOME/lib/security` directory. The implementation of `dfhreg` (`dfhreg.jar`) is also in this directory. The other JAR files required to run `dfhreg` (`dfjcsi.jar`, `dfjejbdd.jar`, and `dfjorb.jar`) are in the `CICS_HOME/lib` directory. `CICS_HOME` is the z/OS UNIX directory in which you have installed the USS components of CICS.

You can execute `dfhreg` on any platform that supports Java; however, you must execute the resulting REXX program against the RACF database on the z/OS system where you wish to define the security roles. When you run `dfhreg`:

1. Your classpath must contain:
   ```
   dfhreg.jar
   dfjcsi.jar
   dfjejbdd.jar
   dfjorb.jar
   ```
2. You must be using a 1.4 or later version of the Java 2 SDK.

The REXX program which the utility generates is in the code page of the platform where the utility executes. If you run the utility on a platform that uses an ASCII code page, you must convert the REXX program to the EBCDIC code page used on the target z/OS system.

Executing the utility

To execute the utility enter the following on the command line:

```
dfhreg [options] inputfiledesc
```

The full syntax is

```
dfhreg [-secprfx secprfx]
       [-out outputfiledesc]
       [-f | -force]
       [-v | -verbose]
       [-? | -help]
       inputfiledesc
```

where

–secprfx *secprfx*
: Specifies the name used to qualify the security role name at system level. The value you specify must match the value of the EJBROLEPRFXEJBROLEPRFX system initialization parameter for the CICS system where the security roles will be used

out *outputfiledesc*
: Specifies the file which to which the utility writes its output. If you do not specify a file, output will be written to standard output.

inputfiledesc
: Specifies the input file containing the deployment descriptor. The file must be a Java archive file (file type `jar`).

–f | –force
: Specifies that the utility will overwrite an existing output file.

–v | –verbose
: Specifies that processing messages will be written to standard output.

–? | –help
: Displays a summary of the syntax for the utility.

All options are case sensitive; the keywords (`-secprfx`, `-out`, `-force`, `-f`, `-verbose`, `-v`, `-help`) must be entered in lower case.

If the utility encounters an error, it generates one or more messages. These are described in the *CICS Messages and Codes* manual.

Defining security roles to RACF

In RACF, deployed security roles are managed as general resources. To define the deployed security roles, define profiles in the GEJBROLE or EJBROLE resource classes, with appropriate access lists.

For example, to use the following commands to define deployed security roles `deployed_security_role_1`and `deployed_securityrole_2` as members of the `securityrole_group` profile in the GEJBROLE class, and give `READ` access to `user1` and `user2`:

```
RDEFINE GEJBROLE securityrole_group UACC(NONE)
                 ADDMEM(deployed_security_role_1, deployed_securityrole_2, ...)
                 NOTIFY(sys_admin_userid)
PERMIT securityrole_group CLASS(GEJBROLE) ID(user1, user2) ACCESS(READ)
```

Alternatively, use the following commands to define deployed security roles in the EJBROLE class, and to give users `READ` access to each deployed security role:

```
RDEFINE EJBROLE (deployed_security_role1, deployed_security_role2, ...) UACC(NONE)
                 NOTIFY(sys_admin_userid)
PERMIT deployed_security_role1 CLASS(EJBROLE) ID(user1, user2) ACCESS(READ)
PERMIT deployed_security_role2 CLASS(EJBROLE) ID(user1, user2) ACCESS(READ)
```

Note:

1. The security role you specify is the deployed security role, and not the unqualified security role which is defined in the deployment descriptor.
2. To execute a bean method, or to receive a `true` response from the `isCallerInRole()` method, a user requires `READ` access.

Part 6. Customization

This part tells you how to customize the interface between CICS and the External Security Manager.

Chapter 21. Customizing security processing

This topic describes the interface between CICS and RACF interface, and how you can customize security processing using RACF exit programs.

This topic contains Product-sensitive Programming Interface and Associated Guidance Information.

Overview of the CICS-RACF interface

In CICS Transaction Server for z/OS, Version 3 Release 2, the only form of security CICS supports is that provided by an external security manager (ESM), such as RACF. CICS uses, by means of the RACROUTE macro, the MVS system authorization facility (SAF) interface to route authorization requests to RACF.

Figure 29 shows how the MVS router exit and RACF user exits are invoked when CICS issues a RACROUTE macro.
See the *z/OS Security Server RACROUTE Macro Reference* for information on how

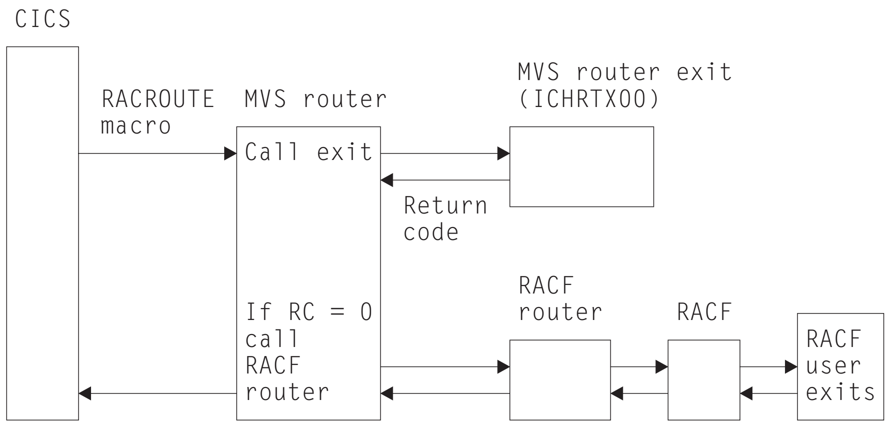


Figure 29. MVS router exit.
1. *CICS issues the RACROUTE macro, which invokes the MVS router.*
2. *The MVS router calls the MVS router exit (ICHRTX00)*
3. *If the MVS router exit's return code is zero, the MVS router calls the RACF router; the RACF router calls the external security manager (RACF in this example). The security manager may call its own user exits.*
4. *The MVS router returns control to CICS.*

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 291.

The 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 *z/OS Security Server RACF Messages and Codes*.

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 37 apply only to RACF. For programming information on non-RACF interfaces, see the *CICS Customization Guide*.

The 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 37.

Table 37. 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

Note:

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 *z/OS Security Server RACF Security Administrator's Guide*. For full descriptions of the RACF exit parameter lists, see the *z/OS Security Server RACF System Programmer's Guide*.

The 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
  - 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 Customization Guide*.

For programming information about user-written ESMs, see the *CICS Customization Guide*.

CICS security control points

This section 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 sign-on (with the parameter ENVIR=CREATE), and at signoff (with the parameter ENVIR=DELETE). It 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 sign-on through EXEC CICS SIGNON, but this call is ignored by RACF):

Each of the following control points relates to ENVIR=CREATE:
- Normal sign-on through EXEC CICS SIGNON
- Sign-on of the default userid DFLTUSER
- Sign-on of preset-security terminal
- Sign-on of MRO session

- Sign-on of LU6.1 session
- Sign-on of LU6.2 session
- Sign-on for XRF tracking of any of the above
- Sign-on associated with the userid on an attach request (for all operands of ATTACHSEC except LOCAL).

Each of the following control points relates to ENVIR=DELETE:
- Normal sign-off through EXEC CICS SIGNOFF
- Sign-off when deleting a terminal
- Sign-off when TIMEOUT expires
- Signoff when USRDELAY expires
- Sign-off of MRO session
- Sign-off of LU6.1 session
- Sign-off of LU6.2 session
- Sign-off for XRF tracking of any of the above.
- Sign-off associated with the userid on an attach request (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:

- Sign-on, as an alternative to VERIFY, when an optimized sign-on 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 a local transaction
- When checking link security for transaction attach
- Transaction validation for an MRO task
- 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
- Remote DL/I PSB scheduling resource check
- When checking a surrogate user authority
- QUERY SECURITY with the RESTYPE option.
- When an enterprise bean invokes the `isCallerInRole()` method.
- When checking the authority of a user to invoke an enterprise bean method.

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 macro 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 sign-on through EXEC CICS SIGNON
- Sign-on of the default userid DFLTUSER
- Sign-on of preset security terminal
- Sign-on of MRO session
- Sign-on of LU6.1 session
- Sign-on of LU6.2 session
- Sign-on for XRF tracking of any of the above
- Sign-on associated with the userid on an attach request (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 CICS control point for bind of an 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 *z/OS Security Server RACROUTE Macro Reference*.

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 “Security-related system initialization parameters” on page 56.
- As the user to be checked for category 1 transactions. For more information, see “Category 1 transactions” on page 140.
- 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).
- For MRO bind security. For more information, see Chapter 16, “Implementing MRO security,” on page 211.

CICS obtains the region userid by invoking the external security manager, which extracts it from the RACF control blocks relevant for the job. The security domain and MRO-bind security each obtain the region userid by issuing a RACROUTE REQUEST=EXTRACT macro. To customize the response from this macro, and thus the security identification of a CICS region, use the MVS security router exit, ICHRTX00.

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, 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, 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.

| Specify the length of the resource name in the MAXLNTH. Specify a length sufficient to contain:
- An eight-character prefix
- A period (.)
- The name of the resource type.

| Thus for RDO resources, with resource type names of up to 12 characters, specify MAXLNTH=23.

| **Remember:** For RDO resources, the name of the resource type can be up to 12 characters long, even though the name of the resource itself is shorter. For example, “PARTITIONSET” contains 12 characters, but the names of PARTITIONSET resources are restricted to eight characters in length. Thus for RDO resources, with resource type names of up to 12 characters, specify MAXLNTH=23.

For example, to add to the CDT a new class $FILEREC, and a corresponding (optional) 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
```

Add the same classes 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 *z/OS Security Server RACF Macros and Interfaces* manual.

When you have recreated the two modules ICHRRCDE and ICHRFR01, re-IPL your MVS system to bring them into use.

Activating the user-defined resource classes

Once you have installed the new classes in your 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)
```

You need to issue the SETROPTS commands 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, refresh the profiles in virtual storage by issuing the command:

```
SETROPTS RACLIST($FILEREC) REFRESH
```

Designing applications to use the user-defined resources

This topic gives 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.

Suppressing 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. 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 need only 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, specify the ESMEXITS=INSTLN system initialization parameter 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 290. 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, 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 sign-on or sign-off, but notify you when the userid associated with a terminal changes. The exits are further described in the *CICS Customization Guide*.

Part 7. Problem determination

This part explains how to determine the cause of security-related problems in CICS.

Chapter 22. Problem determination in a CICS-RACF security environment

This topic 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 CICS is using for the authorization check.

For each security violation, up to three messages are issued:
- CICS issues an authority message to the terminal user (or returns a “not authorized” return code 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 305. For complete descriptions of this and all other RACF messages, see *z/OS Security Server RACF Messages and Codes*.

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 RACF sends the ICH408I message 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 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.
- When you have determined which RACF profile is denying access, see the *z/OS Security Server RACF Security Administrator's Guide*for a

description of authorization checking for RACF-protected resources. The following 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**=*prefix* or **SECPRFX**=YES is specified. The name of the resource in message ICH408I includes the prefix if one of these parameter values is specified in the system initialization parameters.
  - Make sure that an installation-written SAF exit is not changing the effective userid under which the CICS region is running.
- Is CICS using current copies of the RACF resource profiles?

  If 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:
  - SETROPTS GENERIC(*class-name*) REFRESH (a generic profile has been changed).
  - SETROPTS RACLIST(*classname*) REFRESH

  For more information about refreshing resource profiles, see “Refreshing resource profiles in main storage” on page 24.
- 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?” on page 303.

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 transient data destination definition 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 “CICS users” on page 4.

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 access.

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 30 on page 304.
- A slightly longer listing showing the member profile as well as the resource group profile. See Figure 31 on page 304.

- A “profile not found” message, if no profile is found that protects the resource. See Figure 32.
- A “not authorized” message, if If a profile exists, but you are not authorized to list the profile. See Figure 33 on page 305.

```
rlist tcicstrn cemt resgroup
CLASS      NAME
-----      ----
TCICSTRN   CEMT
GROUP CLASS NAME
----- ----- ----
GCICSTRN
RESOURCE GROUPS
-------- ------
CAT2
```

Figure 30. 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 31. Output of RLIST command with RESGROUP: several profiles

Note: When you are using resource group profiles, more than one profile might be used at the same 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 32. 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 33. 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 *z/OS Security Server RACF Command Language Reference*.

RACF message ICH408I

For a complete description of RACF message ICH408I, see the *z/OS Security Server RACF Messages and Codes* manual.

The first line of message ICH408I identifies a user who had an authorization problem. The other lines of the message describe the request the user was issuing and the reason for the failure.

Consider 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 in 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 2 on page 24. 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 26/09/95 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 displays the access list, as shown in Figure 34.

```
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 34. 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. To list the groups to which JONES is connected, issue LISTUSER JONES Other specifications in the profile (such as security level or security category) might cause access to be denied. For a complete description, see the *z/OS Security Server 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, even when you think that the user should **not** have that access. Here are some 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 148.)
- 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 you are using TSMODEL resource definition, check the SECURITY option of the model.
- 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?
  - Is link security being bypassed because the link userid matches the local region userid?
  - 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 *z/OS Security Server RACF Security Administrator's Guide*.

CICS initialization failures related to security

If SEC=YES is specified, external security is *required*. If external security cannot be provided, CICS cannot be initialized.

Figure 35 on page 309 shows an example of a failure to initialize.

If security initialization fails:

- Examine the DFHXS1106 message return codes. In the example shown in Figure 35 on page 309, 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 291.
- 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 *z/OS Security Server RACF Macros and Interfaces* includes a listing of the IBM-supplied module, ICHRFR0X.)

Figure 35 on page 309 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 to the RACF CDT.

RACF abends

If a RACF abend occurs, see *z/OS Security Server RACF Messages and Codes* and *z/OS Security Server 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 36 on page 310 shows an example of a CICS job log when the DFLTUSER fails to sign on. CICS is started with SEC=YES and DFLTUSER=ORMAN. User profile ORMAN has not been defined to RACF.

This CICS region cannot be initialized because, with SEC=YES specified, external security is required and the default user must be defined to RACF.

```
DFHPA1927  IYCTZCCA AKPREQ=0
DFHPA1927  IYCTZCCA APPLID=IYCTZCCA
DFHPA1927  IYCTZCCA CSDFRLOG=NO
DFHPA1927  IYCTZCCA CSDRECOV=NONE
DFHPA1927  IYCTZCCA FCT=NO
DFHPA1927  IYCTZCCA SIT=6$
DFHPA1927  IYCTZCCA START=INITIAL
DFHPA1927  IYCTZCCA GMTEXT='SSYS NOOR CIC5100M SYSTEM with RACF'
DFHPA1927  IYCTZCCA GRPLIST=USERLIST
DFHPA1927  IYCTZCCA PLTPI=NO
DFHPA1927  IYCTZCCA SEC=YES
DFHPA1927  IYCTZCCA XCMD=NO
DFHPA1927  IYCTZCCA XFCT=1CVFFCT
DFHPA1927  IYCTZCCA XJCT=1CVFJCT
DFHPA1927  IYCTZCCA XPCT=1CVFPCT
DFHPA1927  IYCTZCCA XPPT=UNKNOWN
DFHPA1927  IYCTZCCA XPSB=NO
DFHPA1927  IYCTZCCA XTST=1CVFTST
DFHPA1927  IYCTZCCA XTRAN=1CVFTRN
DFHPA1927  IYCTZCCA CICSSVC=212
DFHPA1927  IYCTZCCA SRBSVC=211
DFHPA1927  IYCTZCCA .END
DFHPA1103  IYCTZCCA END OF FILE ON SYSIN.
+DFHTR0103 TRACE TABLE SIZE IS 64K
+DFHSM0122I IYCTZCCA Limit of DSA storage below 16MB is 5,120K.
+DFHSM0123I IYCTZCCA Limit of DSA storage above 16MB is 20M.
+DFHSM0113I IYCTZCCA Storage protection is not active.
+DFHSM0126I IYCTZCCA Transaction isolation is not active.
+DFHDM0101I IYCTZCCA CICS is initializing.
+DFHLG0101I IYCTZCCA Log manager domain initialization has started.
+DFHSI1500 IYCTZCCA CICSTS32.CICS. Startup is in progress.
+DFHXS1100I IYCTZCCA Security initialization has started.
+DFHDU0304I IYCTZCCA Transaction Dump Data set DFHDMPA opened.
+DFHSI1501I IYCTZCCA Loading CICS nucleus.
+DFHXS1105 IYCTZCCA Resource profiles for class A1CVFPCT have been built.
+DFHDU03041 IYCTZCCA Transaction Dump Data set DFHDMPA opened.
+DFHXS1105 IYCTZCCA Resource profiles for class F1CVFFCT have been built.
+DFHXS1105 IYCTZCCA Resource profiles for class J1CVFJCT have been built.
+DFHXS1106 IYCTZCCA
 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 IYCTZCCA Transaction Dump Data set DFHDMPA closed.
+DFHKE1800 IYCTZCCA ABNORMAL TERMINATION OF CICSTS32.CICS IS COMPLETE.
IEF450I SSYTZCCA CICS - ABEND=S000 U1800 REASON=00000000
                           TIME=13.55.26
$HASP395 SSYTZCCA ENDED
```

Figure 35. Security initialization failure

```
DFHPA1927  IYCTZCCE SEC=YES
DFHPA1927  IYCTZCCE XUSER=YES
DFHPA1927  IYCTZCCE DFLTUSER=ORMAN
DFHPA1927  IYCTZCCE XCMD=NO
DFHPA1927  IYCTZCCE XFCT=1CVFFCT
DFHPA1927  IYCTZCCE XJCT=1CVFJCT
DFHPA1927  IYCTZCCE XPCT=1CVFPCT
DFHPA1927  IYCTZCCE XFCT=1CVFPPT
DFHPA1927  IYCTZCCE XPSB=NO
DFHPA1927  IYCTZCCE XTST=1CVFTST
DFHPA1927  IYCTZCCE XTRAN=1CVFTRN
DFHPA1927  IYCTZCCE CICSSVC=212
DFHPA1927  IYCTZCCE SRBSVC=211
DFHPA1927  IYCTZCCE .END
DFHPA1103  IYCTZCCE END OF FILE ON SYSIN.
+DFHTR0103 TRACE TABLE SIZE IS 64K
+DFHSM0122I IYCTZCCE Limit of DSA storage below 16MB is 5,120K.
+DFHSM0123I IYCTZCCE Limit of DSA storage above 16MB is 20M.
+DFHSM0113I IYCTZCCE Storage protection is not active.
+DFHSM0126I IYCTZCCE Transaction isolation is not active.
+DFHDM0101I IYCTZCCE CICS is initializing.
+DFHLG0101I IYCTZCCE Log manager domain initialization has started.
+DFHSI1500 IYCTZCCE CICSTS32.CICS Startup is in progress                  .
+DFHXS1100I IYCTZCCE Security initialization has started.
+DFHDU0304I IYCTZCCE Transaction Dump Data set DFHDMPA opened.
+DFHSI1501I IYCTZCCE Loading CICS nucleus.
+DFHXS1105 IYCTZCCE Resource profiles for class A1CVFPCT have been built.
+DFHXS1105 IYCTZCCE Resource profiles for class F1CVFFCT have been built.
+DFHXS1105 IYCTZCCE Resource profiles for class J1CVFJCT have been built.
+DFHXS1105 IYCTZCCE Resource profiles for class M1CVFPPT have been built.
+DFHXS1105 IYCTZCCE Resource profiles for class S1CVFTST have been built.
+DFHXS1105 IYCTZCCE Resource profiles for class T1CVFTRN have been built.
+DFHXS1105 IYCTZCCE Resource profiles for class SURROGAT have been built.
ICH4081 USER(ORMAN   ) GROUP(      ) NAME(???
  LOGON/JOB INITATION - USER AT TERMINAL      NOT RACF-DEFINED
IRR012I VERIFICATION FAILED. USER PROFILE NOT FOUND.
+DFHXS1104 IYCTZCCE
 Default security could not be established for userid ORMAN.  The
 security domain cannot continue, so CICS is terminated.  SAF codes are
 (X'00000004',X'00000000'). ESM codes are (X'00000004',X'00000000').
+DFHDU0303I IYCTZCCE Transaction Dump Data set DFHDMPA closed.
+DFHKE1800 IYCTZCCE ABNORMAL TERMINATION OF CICSTS32.CICS IS COMPLETE.
IEF450I SSYTZCCE CICS - ABEND=S000 U1800 REASON=00000000
                           TIME=13.55.26
$HASP395 SSYTZCCE ENDED
```

Figure 36. Example of CICS job log if DFLTUSER fails to sign on

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 26/09/95 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 26/09/95 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 26/09/95 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 26/09/95 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
```

CICS region user ID access problem

CICS security initialization can fail if the CICS region user ID does not have access to the necessary Category 1 transactions. A message similar to the following is shown:

```
DFHXS1103I CICSAPPL Default security for userid CICSUSER has been established.
DFHDU0304I CICSAPPL Transaction dump data set DFHDMPA opened.
DFHXS1111 CICSAPPL
 01/18/99 16:05:15 CICSAPPL ???? Security violation by user TESTRGN for resource CATA
in class TCICSTRN.SAF codes are
(X'00000004',X'00000000').ESM codes are (X'00000004',X'00000000').
DFHXS1113 CICSAPPL
The region userid cannot access system transaction CATA.  CICS will terminate.
SAF codes are (X'00000004',X'00000000').ESM codes are (X'00000004',X'00000000').
```

When this occurs the SAF and return codes (X'00000004') indicate that no profile in the TCICSTRN (or GCICSTRN) class is protecting the resource CATA. To resolve this a suitable profile for CATA must be defined in the TCICSTRN (or GCICSTRN) class, and CICS region user ID TESTRGN must be given at least READ access. For guidance you can use example CLIST DFH$CAT1 which is in library CICSTS32.CICS.SDFHSAMP (see "Category 1 transactions" on page 140).

After adding a suitable profile, you must issue the command:

```
SETROPTS RACLIST (TCICSTRN) REFRESH
```

Even if you have also added the profile to the GCICSTRN class, you still only issue this command for TCICSTRN. If you do NOT issue this SETROPTS command, and you restart CICS, initialization will fail again with the same error, because RACF will not be using the updated definitions.

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 sign-on 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 client" on page 186.)
  - The CICS PEM server sign-on transaction is running as a **synclevel 0** transaction. (See "Setting up the PEM client" on page 186.)
- The user data may be in error; check that:
  - Valid lengths are being sent. (See "PEM client input and output data" on page 188 and "Format of user data" on page 187.)
  - Userids and passwords are sent in uppercase EBCDIC. (See "Setting up the PEM client" on page 186.)
  - GDS variables (required in basic conversations) are being used: (See "Format of user data" on page 187.)

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 "PEM client input and output data" on page 188.

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 8. CICSPlex SM security

This part describes how to implement security for CICSPlex SM.

Chapter 23. Implementing CICSPlex SM security

This topic explains how to implement RACF security for CICSPlex SM. The first section provides general information to help you determine who needs access to the various CICSPlex SM functions. The remaining sections provide detailed information on defining CICSPlex SM class names, using resource names, activating security, and refreshing RACF profiles.

Note: For information on using a SAF-compliant external security manager (ESM) other than RACF, refer to Chapter 24, "Invoking a user-supplied external security manager," on page 349.

1. Decide who needs access to CICSPlex SM.
2. Review the general security requirements for CICSPlex SM.
3. Create RACF profiles for the CICSPlex SM data sets.
4. Define the CICSPlex SM started tasks to RACF.
5. If CICS transaction security is active in a CMAS, define the CICSPlex SM transactions to RACF.
6. If CICS transaction security is active in a MAS, define the CICSPlex SM transactions to RACF.
7. Create RACF profiles for the CICSPlex SM views.
8. Create RACF profiles for the CICSPlex SM Web User Interface resources. See the *CICSPlex System Manager Web User Interface Guide* for more information.
9. If desired, activate simulated security checking using the CICSSYS, CPLEXDEF, or MAS views
10. Activate security in the CMASs and MASs using the CICSPlex SM and CICS security-related system initialization parameters

Determining who requires access to CICSPlex SM resources

To determine who requires access to the CICSPlex SM resources, answer the questions and complete the matrix. You can then use the results to create the PERMIT statements that are required in RACF to control access to the resources.

You can control access to CICSPlex SM resources in two ways:

- By restricting access to the objects managed via CICSPlex SM views. This does not affect access to the views themselves but it prevents them from displaying any data.
- By restricting access to Web User Interface view sets, menus and the View Editor. This does not affect access to the objects being managed but prevents access to the view sets, menus and View Editor themselves.

1. Answer the following questions to determine who requires access to the CICSPlex SM resources:

   What groups of users will use CICSPlex SM?

   Your enterprise probably already has several user groups defined to RACF. The groups that typically require access to CICSPlex SM include systems programming, operations, the help desk, applications programming, and performance monitoring. These groups are used as column headings in the security matrix. You can supply their corresponding RACF group IDs. (If necessary, you can ignore, replace, or add groups to the matrix as appropriate for your enterprise.)

Which CICSPlex SM views will each group need to use?

CICSPlex SM manages CICS resources via views. Views are grouped by functionality: configuration, topology, workload management, real-time analysis, operations, monitoring, business application services, and CICSplex management. Not all view groups are appropriate for all users. Certain groups of users will need to use only a subset of views. For example, the systems programming group might need to work with all views, while the help desk group might only need to use one or two. The view groups are listed vertically on the left side of the matrix, along with the high-level qualifier of their CICSPlex SM resource names.

What type of access does each RACF group need?

After deciding who needs to use what, stop universal access to all of the objects managed by all of the views. You can then selectively permit read, update, or alter access to specific view groups. To complete the matrix, specify READ, UPDATE, or ALTER access for each RACF group that needs access to a group of views:

| • Specify READ access to allow a user to inquire on a resource.

| • Specify UPDATE access to allow a user to change a value, using the
| SET or UPDATE command, or perform an action. The user can also
| create or remove a definition, such as a BAS resource object.

| • Specify ALTER access to allow a user to discard an installed
| resource from CICS and allow a user to install a BAS resource
| object.

Which CICSPlex SM Web User Interface views, menus will each group need access to?

Web User Interface views and menus are usually user-defined but like Web User Interface views are most likely grouped by functionality. Not all view sets and menus are appropriate for all users. Certain groups of users require access to a subset of views. For example, the systems programming group might require access to all views and to the View Editor, while the help desk group might not need to use the View Editor or those views that manage the definition of CICSPlex SM resources.

2. Fill out the security matrix when you have answered the questions:

Table 38. Security matrix

RACF group → **CICSPlex SM view group ↓**	**System Programming ID( )**	**Operations ID( )**	**Help Desk ID( )**	**Application Programming ID( )**	**Performance ID( )**
Configuration CONFIG					
Topology TOPOLOGY					
Workload Management WORKLOAD					
Real-Time Analysis ANALYSIS					
Operations OPERATE					
Monitor MONITOR					
Business Application Services BAS					

Table 38. Security matrix (continued)

RACF group → CICSPlex SM view group ↓	System Programming ID( )	Operations ID( )	Help Desk ID( )	Application Programming ID( )	Performance ID( )
PlexManager BBM.PLEXMGR					

Table 39 is a sample of a completed security matrix for a production CICSplex:

Table 39. Sample security matrix

RACF group → CICSPlex SM view group ↓	System Programming ID(SYSPGRP)	Operations ID(OPSGRP)	Help Desk ID(HELPGRP)	Application Programming ID(APPLGRP)	Performance ID(PERFGRP)
Configuration CONFIG	UPDATE				
Topology TOPOLOGY	UPDATE	UPDATE	READ		
Workload Management WORKLOAD	UPDATE			READ	
Real-Time Analysis ANALYSIS	UPDATE	UPDATE	READ		READ
Operations OPERATE	ALTER	UPDATE	READ	READ	READ
Monitor MONITOR	UPDATE	READ			READ
Business Application Services BAS	ALTER	ALTER		UPDATE	
PlexManager BBM.PLEXMGR	UPDATE				

3. Ensure that the CPSMOBJ class is active and that generic profiles can be defined:

   ```
   SETROPTS CLASSACT(CPSMOBJ)
   SETROPTS GENERIC(CPSMOBJ)
   SETROPTS GENCMD(CPSMOBJ)
   ```

4. Create RACF profile to protect all of the views and action commands for all CICSPlex SM functions:

   ```
   RDEF CPSMOBJ ** UACC(NONE) OWNER(admin_group) NOTIFY(admin_user)
   ```

   CPSMOBJ is the CICSPlex SM member class. The double asterisks indicate that all of the CICSPlex SM views are included in this RDEF statement.

5. Using the information in the sample matrix, you can permit access to the specific view groups. For example, the system programming group requires update access to all of the view groups and ALTER access to the BAS views. You can define this with just three PERMIT statements:

   ```
   PERMIT ** CLASS(CPSMOBJ)  ID(SYSPGRP) ACCESS(UPDATE)
   PERMIT BBM.PLEXMGR.** CLASS(FACILITY) ID(SYSPGRP) ACCESS(UPDATE)
   PERMIT BAS.** CLASS(CPSMOBJ) ID(SYSPGRP) ACCESS(ALTER)
   ```

   The double asterisks indicate that all of the CICSPlex SM views are affected by this PERMIT statement.

   The following PERMIT statements grant the appropriate access to all of the topology views for the operations and help desk groups:

```
PERMIT TOPOLOGY.** CLASS(CPSMOBJ) ID(OPSGRP)  ACCESS(UPDATE)
             PERMIT TOPOLOGY.** CLASS(CPSMOBJ) ID(HELPGRP) ACCESS(READ)
```

For the workload management views:

```
PERMIT WORKLOAD.** CLASS(CPSMOBJ) ID(APPLGRP) ACCESS(READ)
```

For the real-time analysis views:

```
PERMIT ANALYSIS.** CLASS(CPSMOBJ) ID(OPSGRP)  ACCESS(UPDATE)
PERMIT ANALYSIS.** CLASS(CPSMOBJ) ID(HELPGRP) ACCESS(READ)
PERMIT ANALYSIS.** CLASS(CPSMOBJ) ID(PERFGRP) ACCESS(READ)
```

For the operations views:

```
PERMIT OPERATE.** CLASS(CPSMOBJ) ID(OPSGRP)  ACCESS(UPDATE)
PERMIT OPERATE.** CLASS(CPSMOBJ) ID(HELPGRP) ACCESS(READ)
PERMIT OPERATE.** CLASS(CPSMOBJ) ID(APPLGRP) ACCESS(READ)
PERMIT OPERATE.** CLASS(CPSMOBJ) ID(PERFGRP) ACCESS(READ)
```

For the monitor views:

```
PERMIT MONITOR.** CLASS(CPSMOBJ) ID(APPLGRP) ACCESS(READ)
PERMIT MONITOR.** CLASS(CPSMOBJ) ID(PERFGRP) ACCESS(READ)
```

For the business application services views:

```
PERMIT BAS.** CLASS(CPSMOBJ) ID(OPSGRP) ACCESS(ALTER)
PERMIT BAS.** CLASS(CPSMOBJ) ID(APPLGRP) ACCESS(UPDATE)
```

For simplicity, these PERMIT statements grant access to broad groups of views by using the double asterisks in the resource names. However, if required, you can use more specific resource names in your PERMIT statements. See “Specifying CICSPlex SM resource names in profiles” on page 324 for details.

Using your own completed security matrix and the information in the remainder of this section, you can create as many profiles as required for your enterprise. Chapter 25, “Example tasks: security,” on page 353 provides detailed profile examples.

General requirements for CICSPlex SM security

You should review your RACF configurations to ensure that the following minimum requirements are met:

- The IDs for all users expected to use CICSPlex SM should be defined to RACF in each MVS system in which there is a CMAS. For each individual user, the ID must be the same for each MVS system.
- User access authority to CICSPlex SM definitions and CICS commands and resources should be defined to RACF in a consistent manner in all MVS systems used by CICSPlex SM.

In addition, you should be aware that, in the CMAS address space, a security environment is created for the user specified in the DFLTUSER system initialization parameter associated with the MAS.

Creating profiles for the CICSPlex SM data sets

You should restrict access to CICSPlex SM data sets using RACF data set protection. Use the following guidelines:

- Prohibit universal access, by specifying UACC(NONE).

- Ensure that minimum access to the data sets is authorized for the RACF USERID assigned to the:
  - Each CMAS job (or started task).
  - Each MAS.
  - All individuals allowed to use CICSPlex SM via the CICSPlex SM WUI and API (both system administrators and end users).

Table 40 lists the CICSPlex SM data sets and the minimum access that should be granted to each type of user ID.

Table 40. Access by user ID for CICSPlex SM data sets

Data set name	CMAS	MAS	System Admin.	Individual User
SYS1.CICSTS32.CPSM.SEYULPA	NONE	READ	UPDATE	NONE
SYS1.CICSTS32.CPSM.SEYULINK	READ	NONE	UPDATE	NONE
CICSTS32.CPSM.SEYUAUTH	READ	READ	UPDATE	READ
CICSTS32.CPSM.SEYULOAD	READ	READ	UPDATE	NONE
CICSTS32.CPSM.SEYUPARM	READ	READ	UPDATE	NONE
CICSTS32.CPSM.SEYUCMOD	NONE	NONE	UPDATE	NONE
CICSTS32.CPSM.SEYUCOB	NONE	NONE	UPDATE	READ
CICSTS32.CPSM.SEYUC370	NONE	NONE	UPDATE	READ
CICSTS32.CPSM.SEYUDEF	READ	READ	UPDATE	READ
CICSTS32.CPSM.SEYUCLIB	NONE	NONE	UPDATE	READ
CICSTS32.CPSM.SEYUMLIB	NONE	NONE	UPDATE	READ
CICSTS32.CPSM.SEYUPLIB	NONE	NONE	UPDATE	READ
CICSTS32.CPSM.SEYUTLIB	NONE	NONE	UPDATE	READ
CICSTS32.CICS.SDFHINST	NONE	NONE	UPDATE	NONE
CICSTS32.CPSM.SEYUMAC	NONE	NONE	UPDATE	READ
CICSTS32.CPSM.SEYUOS2	NONE	NONE	UPDATE	NONE
CICSTS32.CPSM.SEYUPL1	NONE	NONE	UPDATE	READ
CICSTS32.CPSM.SEYUPROC	NONE	NONE	UPDATE	READ
CICSTS32.CPSM.SEYUSAMP	NONE	NONE	UPDATE	READ
CICSTS32.CPSM.EYUSDEF	NONE	NONE	UPDATE	UPDATE
CICSTS32.CPSM.EYUDREP	UPDATE	NONE	UPDATE	NONE
CICSTS32.CPSM.EYUIPRM	NONE	NONE	UPDATE	NONE

For more details about RACF data set protection, see the *z/OS Security Server RACF Security Administrator's Guide*.

Defining the CICSPlex SM started tasks

When you run a CMAS as a started task, you must associate the appropriate procedure names with a suitably authorized USERID. This is normally achieved using the STARTED general resource class, or the RACF ICHRIN03 tables. The names of the associated USERIDs need not match the names of the procedures. Each USERID must have the appropriate level of access to all of the data sets referenced in the cataloged procedures.

For additional information about the STARTED class, see the *z/OS Security Server RACF Security Administrator's Guide*. For more information about ICHRIN03, see the *z/OS Security Server RACF System Programmer's Guide*.

Note: If the USERID and group name that you assign are not defined to RACF, the started tasks will execute with only the limited authority of an undefined user.

In this case, the address space will be able to access protected resources only if the universal access authority (UACC) for the resource is sufficient to allow the requested operation.

Defining the CICSPlex SM transactions in a CMAS

If transaction–attach security is active in a CMAS (that is, SEC=YES and XTRAN=YES∨*classname* are specified in the system initialization parameters), you must define to RACF the CICSPlex SM transactions that run in a CMAS.

1. You must define the following CICSPlex SM transactions to RACF:

BMLT	LPLT	PPLT	WMSC
LCPP	LPRT	PRLT	WMWC
LCMU	LPSC	PRPR	WMWT
LECI	LPSM	PSLT	WSCL
LECR	LRLT	TICT	WSLW
LECS	LSGT	TIRT	XDBM
LEEI	LSRT	TIST	XDNC
LEER	LWTM	TSMH	XDND
LEMI	MCCM	TSPD	XDNE
LEMS	MCTK	TSSC	XDNR
LENS	MMEI	TSSJ	XDNS
LMIR	MMIS	WMCC	XDSR
LNCI	MMST	WMGR	XLEV
LNCS	PEAD	WMLA	XLNX
LNMI	PELT	WMQB	XLST
LNMS	PMLT	WMQM	XMLT
LPDG	PNLT	WMQS	XQST
LPLK			

   A list of these transactions is also contained in the CSD group EYU$CDEF.

   a. If the CMAS is started by a PLT program, ensure that the region user ID , and any user ID that might be specified on the PLTPIUSR system initialization parameter, has authority to attach these transactions.
   b. Depending on the security attributes specified for any CMTCMDEF or CMTPMDEF, ensure that any user IDs that might flow from connected CMASs have the authority to attach these transactions. Note that the CICS default user ID might flow if the connected CMAS or MASs are defined as equivalent systems.

2. Define the following debugging transactions to RACF if transaction security is active, regardless of the CICS release running as the CMAS:
   - CODB
   - COD0
   - COD1
   - COD2
   - COLU

   These transactions are associated with a terminal and are supplied for debugging purposes under the guidance of IBM support personnel. Authority to initiate these transactions must be restricted to only those users who might become involved in working with IBM to resolve problems.

3. Give users access to the CESD shutdown-assist transaction. Users who can attach CICSPlex SM transactions or define debugging transactions need access to CESD in case of CMAS failure.

4. Allow only users who might need to shut down a CMAS to access the COSD transaction. The COSD transaction allows a terminal user to shut down a CMAS.

Defining the CICSPlex SM transactions in a MAS

For MASs capable of running with an external security manager, it may be necessary to define the CICSPlex SM transactions which run in the MAS to the ESM.

If transaction–attach security is active in a MAS (that is, SEC=YES and XTRAN=YES|*classname* are specified in the system initialization parameters), you must define to RACF the following transactions in the appropriate class:

COHT
COIE
COIR
COI0
| CONA
COND
CONH
CONL
CONM
CORT
COWC

The region userid, and any userid that may be specified on the PLTPIUSR system initialization parameter, should be given READ access to these transactions.

For CICS/MVS, CICS/ESA, and CICS TS for OS/390, users who may initiate the MAS agent code using transaction COLM (for a local MAS) should also be given access to these transactions. If your CICS system performs surrogate user checks (that is, the XUSER system initialization parameter has a value of YES), then the CICS region userid should be a surrogate of the user of the COLM transaction.

For details of the PLTIPUSR and XUSER system initialization parameters, see the *CICS System Definition Guide*.

Users who may enter dynamic transactions in a CICSPlex SM workload management requesting region must have READ access to the COWC transaction.

For CICS/MVS, CICS/ESA, and CICS TS for OS/390, users who may invoke the CICSPlex SM debugging transactions should be given READ access to the following transactions:

CODB
COD0
COD1
COD2
COLU

The security attributes of the CONNECTION/SESSION pair defined for the link to the CMAS define which users are authorized to run these transactions. See Chapter 11, “Overview of intercommunication security,” on page 153 for information on intercommunication security.

The COSH transaction allows a terminal user to stop MAS agent code execution. Access to this transaction should be restricted to those users who may need to stop the MAS in this way.

Defining the CICSPlex SM transactions for a WUI

For Web User Interface (WUI) regions that are capable of running with an external security manager (ESM) such as RACF, you must define the CICSPlex SM transactions that run in the region to the ESM.

This is necessary when transaction attach security is active in the WUI region, where `SEC=YES` and `XTRAN=YES` system initialization parameters are specified.

1. Create the same definitions as you would for a MAS region. See “Defining the CICSPlex SM transactions in a MAS” on page 323.
2. Define READ access to the COVG and COVC transactions for the following user IDs:
   - The region user ID
   - All user IDs that are specified on the **`PLTPIUSR`** system initialization parameter for the region
   - All WUI system administrators.
3. Define READ access to the COVE, COVP, and COVU transactions for the WUI default user ID.
4. Define READ access to the COVA transaction for all WUI end users.

A list of these transactions is also contained in the CSD group EYU$WDEF .

Specifying CICSPlex SM resource names in profiles

This section provides the resource names for CICSPlex SM views to be used in RACF profiles. Refer to Chapter 25, “Example tasks: security,” on page 353 for profile examples.

You can create RACF profiles for CICSPlex SM views for a specific CICS system, a group of CICS systems, or all systems comprising a CICSplex.

CICSPlex SM views are divided into groups that reflect the functions they perform. Within each functional group, the views are divided by their type. Functional groups can be even further qualified with the addition of a context and, for some groups, a scope. You can control access to a specific set of views (and their associated action commands) by identifying the set in a profile, using one of the following resource name formats:

| *`function.type.context`*

| *`function.type.context.scope`*

where:

`function`
is the name of the CICSPlex SM function to be affected:

Function	Meaning
ANALYSIS	Real-time analysis
BAS	Business application services
CONFIG	CMAS configuration
MONITOR	Resource monitoring
OPERATE	Operations
TOPOLOGY	CICSplex configuration
WORKLOAD	Workload management

type is the specific or generic name of an area that qualifies the CICSPlex SM function to be affected. The specific names are:

Type	Meaning
AIMODEL	CICS AIMODEL
BRFACIL	Link3270 Bridge Facility
CONNECT	CICS connections
DB2DBCTL	DB2/DBCTL resources and subsystems
DEF	CICSPlex SM definitions
DOCTEMP	Document templates
ENQMODEL	CICS global enqueue models
ENTJAVA	CorbaServers and deployed DJARs
EXIT	CICS exits
FEPI	CICS FEPI resources
FILE	CICS files
IPCONN	IPIC connections
JOURNAL	Journal models
PARTNER	CICS partners
PROCTYPE	CICS BTS Process types
PROFILE	CICS profiles
PROGRAM	CICS programs
REGION	CICS region data
RQMODEL	CICS request models
TASK	CICS active tasks
TCPIPS	TCP/IP services

TDQUEUE
: CICS transient data queues

TERMINAL
: CICS terminals

TRAN
: CICS transactions

TSQUEUE
: CICS temporary storage queues

UOW
: CICS units of work

The type must be valid for the specified function. Table 41 on page 327 lists the valid `function.type` combinations.

`context`
is the specific or generic name of the CMAS or CICSplex to be affected by the designated `function` and `type`. If the `function` is CONFIG or TOPOLOGY, the `context` must be a CMAS. For all other functions, the `context` must be a CICSplex.

| **`scope`** is the specific or generic name of a CICS system within the CICSplex
| identified as the context or the CICSplex itself. Do not specify *scope* when:
| • The context is a CMAS
| • *or* the type is DEF

| for CICSPlex SM definitions.

| **Note:**

| 1. In this section only, the term `scope` means CICS systems. It does
| ***not*** mean the scope (CICS system groups) you have defined as
| part of the CICSPlex SM environment, nor does it refer to a BAS
| logical scope.
| 2. To include all of the systems comprising a CICS system group
| when their names do not match a generic system name, you
| must establish a profile for each system.

| Using asterisks in resource names

To reduce the number of profiles you need to define, you can use * (one asterisk) and ** (two consecutive asterisks) to represent one or more entries. Use of one or two asterisks is optional.

Note: Before using asterisks in profile definitions, ensure that generics have been activated for the relevant class:

```
SETROPTS GENERIC(CPSMOBJ,CPSMXMP)
```

The following examples demonstrate how asterisks can be used:

`OPERATE.*.EYUPLX01.EYUPLX01`
Indicates that all views and action commands associated with any type valid within the OPERATE function are to be recognized when the context and scope are EYUPLX01.

`OPERATE.PROGRAM.**`
Indicates that all views and action commands associated with the PROGRAM type within the OPERATE function are to be recognized, regardless of the current context and scope.

OPERATE.**
: Indicates that all views and action commands associated with any type valid within the OPERATE function are to be recognized, regardless of the current context and scope.

******
: Indicates that *all* views and action commands associated with *any* type valid within *any* function are to be recognized, regardless of the current context and scope.

Valid resource name combinations

| Table 41 lists the valid function and type combinations and the resource names associated with each combination. You can use these function and type combinations when creating profiles to control access to resource tables using the CICSPlex SM API or WUI. If you are using the WUI, note that the resource name is given at the foot of the views in the corresponding view set.

Table 41. Function and type combinations for resources accessible via CICSPlex SM

Function.Type	Resource Table Name and Usage
ANALYSIS.DEF	**ACTION** Create, display, and maintain action definitions
	APACTV Display analysis definitions associated with an analysis point specification
	APSPEC Create, display, and maintain analysis point specifications
	CMDMPAPS Identify the role of a primary CMAS
	CMDMSAPS Identify the role of a secondary CMAS
	EVALDEF Create, display, and maintain evaluation definitions
	EVENT Display changes in the status of a CICSplex
	EVENTDTL Display evaluation definitions associated with an analysis definition that caused an event
	LNKSRSCG Describe the link between a CICS system group and an analysis specification
	LNKSRSCS Describe the link between a CICS system and an analysis specification
	RTAACTV Display analysis and status definitions in CICS systems
	RTADEF Create, display, and maintain analysis definitions
	RTAGROUP Create, display, and maintain analysis groups
	RTAINAPS Display analysis groups in analysis point specifications
	RTAINGRP Display analysis and status definitions in analysis groups
	RTAINSPC Display analysis groups in analysis specifications
	RTASPEC Create, display, and maintain analysis specifications
	STATDEF Create, display, and maintain status definitions
	STAINGRP Identify the membership relation of a status definition in an RTAGROUP

Table 41. Function and type combinations for resources accessible via CICSPlex SM (continued)

Function.Type	Resource Table Name and Usage
BAS.CONNECT	**CONNDEF** Install ISC/MRO connection definitions.
	IPCONDEF Install IPIC connection definitions.
	SESSDEF Install session definitions.
BAS.DB2DBCTL	**DB2CDEF** Install DB2 connection definitions
	DB2EDEF Install DB2 entry definitions
	DB2TDEF Install DB2 transaction definitions
BAS.DEF	**CONNDEF** Create, display, and maintain MRO and ISC over SNA connection definitions.
	DB2CDEF Create, display, and maintain DB2 connection definitions.
	DB2EDEF Create, display, and maintain DB2 entry definitions.
	DB2TDEF Create, display, and maintain DB2 transaction definitions.
	DOCDEF Create, display, and maintain document template definitions.
	EJCODEF Create, display, and maintain CorbaServer definitions.
	EJDJDEF Create, display, and maintain deployed JAR file definitions.
	ENQMDEF Create, display, and maintain enqueue models definitions.
	FENODDEF Create, display, and maintain FEPI node definitions.
	FEPOODEF Create, display, and maintain FEPI pool definitions.
	FEPRODEF Create, display, and maintain FEPI property set definitions.
	FETRGDEF Create, display, and maintain FEPI target definitions.
	FILEDEF Create, display, and maintain file definitions.
	FSEGDEF Create, display, and maintain OS/2 key file segment definitions.
	IPCONDEF Create, display, and maintain IPIC connection definitions.
	JRNMDEF Create, display, and maintain journal model definitions.
	LIBDEF Create, display, and maintain LIBRARY definitions.
	LSRDEF Create, display, and maintain LSR pool definitions.
	MAPDEF Create, display, and maintain mapset definitions.
	PARTDEF Create, display, and maintain partner definitions.
	PIPEDEF Create, display, and maintain pipeline definitions.
	PROCDEF Create, display, and maintain process type definitions.

Table 41. Function and type combinations for resources accessible via CICSPlex SM (continued)

Function.Type	Resource Table Name and Usage
BAS.DEF (continued)	**PROFDEF** Create, display, and maintain profile definitions.
	PROGDEF Create, display, and maintain program definitions.
	PRTNDEF Create, display, and maintain partition set definitions.
	RASGNDEF Create, display, and maintain resource assignments
	RASINDSC Display resource assignments in descriptions
	RASPROC Display resource assignment process
	RDSCPROC Display resource description process
	RESDESC Create, display, maintain, and install resource descriptions
	RESGROUP Create, display, maintain, and install resource groups
	RESINDSC Display resource groups in descriptions
	RESINGRP Display resource definitions in groups
	RQMDEF Create, display, and maintain request model definitions.
	SESSDEF Create, display, and maintain session definitions
	SYSRES Display CICS system resources
	TCPDEF Create, display, and maintain TCP/IP service definitions.
	TDQDEF Create, display, and maintain transient data queue definitions
	TERMDEF Create, display, and maintain terminal definitions
	TRANDEF Create, display, and maintain transaction definitions
	TRNCLDEF Create, display, and maintain transaction class definitions
	TYPTMDEF Create, display, and maintain typeterm definitions
	URIMPDEF Create, display, and maintain URI map definitions
	WEBSVDEF Create, display, and maintain Web services definitions

Table 41. *Function and type combinations for resources accessible via CICSPlex SM (continued)*

Function.Type	Resource Table Name and Usage
BAS.DEF (continued)	**CONINGRP** Describe the membership of an MRO or ISC over SNA connection definition in a resource group
	DOCINGRP Describe the membership of a document template definition in a resource group
	D2CINGRP Describe the membership of a DB2 connection definition in a resource group
	D2EINGRP Describe the membership of a DB2 entry definition in a resource group
	D2TINGRP Describe the membership of a DB2 transaction definition in a resource group
	EJCINGRP Describe the membership of a CorbaServer definition in a resource group
	EJDINGRP Describe the membership of a deployed JAR file definition in a resource group
	ENQINGRP Describe the membership of an ENQ/DEQ model definition in a resource group
	FILINGRP Describe the membership of a file definition in a resource group
	FNOINGRP Describe the membership of a FEPI node definition in a resource group
	FPOINGRP Describe the membership of a FEPI pool definition in a resource group
	FPRINGRP Describe the membership of a FEPI property set definition in a resource group
	FSGINGRP Describe the membership of a file key segment definition in a resource group
	FTRINGRP Describe the membership of a FEPI target definition in a resource group
	IPCINGRP Describe the membership of an IPIC connection definition in a resource group
	JRMINGRP Describe the membership of a journal model definition in a resource group
	LIBINGRP Describe the membership of a LIBRARY definition in a resource group
	LSRINGRP Describe the membership of an LSR pool definition in a resource group
	MAPINGRP Describe the membership of a map set definition in a resource group
	PARINGRP Describe the membership of a partner definition in a resource group
	PGMINGRP Describe the membership of a program definition in a resource group
	PIPINGRP Describe the membership of a pipeline definition in a resource group
	PRCINGRP Describe the membership of a process type definition in a resource group
	PRNINGRP Describe the membership of a partition set definition in a resource group
	PROINGRP Describe the membership of a profile definition in a resource group
	RQMINGRP Describe the membership of a request model definition in a resource group

Table 41. Function and type combinations for resources accessible via CICSPlex SM (continued)

Function.Type	Resource Table Name and Usage
BAS.DEF (continued)	**SESINGRP** Describe the membership of a session definition in a resource group **TCLINGRP** Describe the membership of a transaction class definition in a resource group **TCPINGRP** Describe the membership of a TCPIP Service definition in a resource group **TDQINGRP** Describe the membership of a transient data queue definition in a resource group **TRMINGRP** Describe the membership of a terminal definition in a resource group **TRNINGRP** Describe the membership of a transaction definition in a resource group **TSMINGRP** Describe the membership of a temporary storage model definition in a resource group **TYPINGRP** Describe the membership of a typeterm definition in a resource group **URIINGRP** Describe the membership of a URI map definition in a resource group **WEBINGRP** Describe the membership of a web service definition in a resource group
BAS.DOCTEMP	**DOCTEMP** Install document template definitions.
BAS.ENQMODEL	**ENQMDEF** Install enqueue model definitions.
BAS.ENTJAVA	**EJCODEF** Install CorbaServer definitions. **EJDJDEF** Install deployed JAR file definitions.
BAS.FILE	**FILEDEF** Install file definitions.
BAS.JOURNAL	**JRNMDEF** Install journal model definitions.
BAS.PARTNER	**PARTDEF** Install partner definitions.
BAS.PROCTYPE	**PROCDEF** Install BTS Process type definitions.
BAS.PROFILE	**PROFDEF** Install profile definitions.
BAS.PROGRAM	**LIBDEF** Install LIBRARY definitions. **MAPDEF** Install map set definitions. **PROGDEF** Install program definitions. **PRTNDEF** Install partition set definitions.
BAS.REGION	**LSRDEF** Install LSR pool definitions. **TRNCLDEF** Install transaction class definitions.

Table 41. Function and type combinations for resources accessible via CICSPlex SM (continued)

Function.Type	Resource Table Name and Usage
BAS.TCPIPS	**TCPDEF** Install TCPIP service definitions.
	PIPEDEF Install Pipeline definitions
	URIMPDEF Install URI map definitions.
	WEBSVDEF Install Web services definitions
BAS.TDQUEUE	**TDQDEF** Install transient data queue definitions.
BAS.TERMINAL	**TERMDEF** Install terminal definitions.
	TYPTMDEF Install typeterm definitions.
BAS.TRAN	**TRANDEF** Install transaction definitions.
CONFIG.DEF	**CICSPLEX** Display and manage CMAS in CICSplex
	CMAS Display and manage active CMASs
	CMASLIST Describe a CMAS and its connection to characteristics
	CMASPLEX Display CICSplexes for a CMAS
	CMTCMDEF Create, display, and maintain CMAS links
	CMTCMLNK Display active CMAS links
	CMTPMLNK Display an active CMAS-to-MAS link
	CPLEXDEF Create, display, and maintain CICSplex definitions
	CPLXCMAS Display CMAS to CICSplex
MONITOR.CONNECT	**MCONNECT** ISC and MRO connections
	MMODENAME LU 6.2 modenames
MONITOR.DB2DBCTL	**MDB2THRD** DB2 threads
MONITOR.DEF	**LNKSMSCG** Describe the link between a CICS system group and a monitor specification
	LNKSMSCS Describe the link between a CICS system and a monitor specification
	MONDEF Create, display, and maintain monitor definitions
	MONGROUP Create, display, and maintain monitor groups
	MONINGRP Create, display, and maintain monitor definitions in monitor groups
	MONINSPC Create, display, and maintain monitor groups in monitor specifications
	MONSPEC Create, display, and maintain monitor specifications
	POLMON Describe a monitor definition in a specific CICS system
MONITOR.FEPI	**MFEPICON** FEPI connections

Table 41. Function and type combinations for resources accessible via CICSPlex SM (continued)

Function.Type	Resource Table Name and Usage
MONITOR.FILE	**MCMDT** Data tables
	MLOCFILE Local files
	MREMFILE Remote files
MONITOR.JOURNAL	**MJOURNL** Journals
	MJRNLNAM Journal names
MONITOR.PROGRAM	**MPROGRAM** Programs
MONITOR.REGION	**MCICSDSA** Dynamic storage areas
	MCICSRGN CICS systems
	MLSRPBUF LSRPOOL buffer pool
	MLSRPOOL LSRPOOL
	MTRANCLS Transaction classes
MONITOR.TDQUEUE	**MINDTDQ** Indirect transient data queues
	MNTRATDQ Intrapartition transient data queues
	MREMTDQ Remote transient data queues
	MTDQGBL Global intrapartition transient data queues
	MXTRATDQ Extrapartition transient data queues
MONITOR.TERMINAL	**MTERMNL** Terminals
MONITOR.TRAN	**MLOCTRAN** Local transactions
	MREMTRAN Remote transactions
MONITOR.TSQUEUE	**MTSQGBL** Global temporary storage queues
OPERATE.AIMODEL	**AIMODEL** AutoInstall models
	CRESAIMD Describe an instance of an autoinstalled terminal model within a CICS system
OPERATE.BRFACIL	**BRFACIL** LINK3270 bridge facility

Table 41. Function and type combinations for resources accessible via CICSPlex SM (continued)

Function.Type	Resource Table Name and Usage
OPERATE.CONNECT	**CONNECT** MRO and ISC over SNA connections **CRESCONN** Describe an instance of an MRO or ISC over SNA connection within a CICS system **CRESIPCN** Describe an instance of an IPIC connection within a CICS system **CRESMODE** Describe an instance of an LU6.2 modename within a CICS system | **IPCONN** IPIC connections **MODENAME** LU 6.2 modenames
OPERATE.DB2DBCTL	**CRESDB2C** Describe an instance of a DB2 connection within a CICS system **CRESDB2E** Describe an instance of a DB2 entry within a CICS system **CRESDB2T** Describe an instance of a DB2 transaction within a CICS system **DB2CONN** DB2 connection **DB2ENTRY** DB2 entry **DB2TRN** DB2 transaction **DBCTLSS** DBCTL subsystem **DB2SS** DB2 subsystem **DB2THRD** DB2 threads **DB2TRAN** DB2 transactions
OPERATE.DOCTEMP	**CRESDOCT** Describe an instance of a document template within a CICS system **DOCTEMP** Document templates
OPERATE.ENQMODEL	**CRESENQM** Describe an instance of an ENQ/DEQ model within a CICS system **ENQMODEL** Enqueue models
OPERATE.ENTJAVA	**CLCACHE** Shared class caches **EJDJBEAN** Enterprise beans in a CICS-deployed JAR file **EJCOBEAN** Enterprise beans in CorbaServers **EJCOSE** CorbaServers **EJDJAR** Deployed JAR files **JVM** Java virtual machines **JVMPOOL** JVM pools **JVMPROF** JVM profiles

Table 41. Function and type combinations for resources accessible via CICSPlex SM (continued)

Function.Type	Resource Table Name and Usage
OPERATE.EXIT	**CRESGLUE** Describe an instance of a global user exit within a CICS system
	CRESTRUE Describe an instance of a task-related user exit within a CICS system
	EXITGLUE Global user exits
	EXITTRUE Task-related user exits
	EXTGLORD Ordered global user exits
OPERATE.FEPI	**CRESFECO** Describe an instance of a FEPI connection within a CICS system
	CRESFEND Describe an instance of a FEPI node within a CICS system
	CRESFEPO Describe an instance of a FEPI pool within a CICS system
	CRESFETR Describe an instance of a FEPI target within a CICS system
	FEPICONN FEPI connections
	FEPINODE FEPI nodes
	FEPIPOOL FEPI pools
	FEPIPROP FEPI property sets
	FEPITRGT FEPI targets
OPERATE.FILE	**CFDTPOOL** Coupling facility data tables pools
	CMDT Data tables
	CRESDSNM Describe an instance of a data set within a CICS system
	CRESFILE Describe an instance of a file within a CICS system
	DSNAME Data sets
	LOCFILE Local files
	REMFILE Remote files
OPERATE.JOURNAL	**CRESJRNL** Describe an instance of a journal within a CICS system
	CRESJRNM Describe an instance of a journal name within a CICS system
	JRNLMODL Journal models
	JRNLNAME System logs and general logs
	STREAMNM MVS log streams
OPERATE.PARTNER	**CRESPART** Describe an instance of a partner table within a CICS system
	PARTNER CICS partners

Table 41. Function and type combinations for resources accessible via CICSPlex SM (continued)

Function.Type	Resource Table Name and Usage
OPERATE.PROCTYPE	**CRESPRTY** Describe an instance of a process type within a CICS system
	PROCTYP Process types
OPERATE.PROFILE	**CRESPROF** Describe an instance of a profile within a CICS system
	PROFILE CICS profiles
OPERATE.PROGRAM	**CRESPRGM** Describe an instance of a program within a CICS system
	PROGRAM Programs
	LIBDSN LIBRARY data set names
	LIBRARY LIBRARY data sets
	RPLLIST DFHRPL data sets

Table 41. Function and type combinations for resources accessible via CICSPlex SM (continued)

Function.Type	Resource Table Name and Usage
OPERATE.REGION	**CICSDSA** Dynamic storage areas
	CICSPAGP CICS page pools
	CICSRGN CICS systems
	CICSSTOR All CICS dynamic storage areas
	CRESSDMP Describe an instance of a system dump code within a CICS system
	CRESTDMP Describe an instance of a transaction dump code within a CICS system
	DOMSPOOL CICS storage domain subpools
	DSPGBL Global CICS dispatcher
	DSPMODE CICS dispatcher TCB mode
	DSPPOOL CICS Dispatcher TCB pool
	ENQUEUE CICS enqueues
	LOADACT CICS Loader activity by dynamic storage area
	LOADER CICS loader activity
	LSRPBUF Buffer usage for LSR pools
	LSRPOOL LSR pools
	| **MASHIST** MAS history
	MONITOR CICS monitoring and statistics
	MQCONN WebSphere MQ connection
	MVSESTG MVS storage element
	MVSTCB MVS TCB
	MVSTCBGL Global MVS TCB
	MVSWLM MVS workload manager
	RECOVERY CICS recovery manager
	REQID Timed requests
	SYSDUMP System dump codes
	TRANCLAS Transaction classes
	TRANDUMP Transaction dump codes
OPERATE.RQMODEL	**CRESRQMD** Describe an instance of a request within a CICS system
	RQMODEL Request models

Table 41. Function and type combinations for resources accessible via CICSPlex SM (continued)

Function.Type	Resource Table Name and Usage
| OPERATE.TASK	**EXCI**
|	**HTASK** Completed task
	TASK Active tasks
	TASKESTG Task storage element
	TASKFILE Task file usage
	TASKRMI RMI usage by individual task
	TASKTSQ TSQ usage by individual task
	TSKSPOLS All task subpools
	TSKSPOOL Task storage subpools
	WORKREQ EJB work requests
OPERATE.TCPIPS	**CRESTCPS** Describe an instance of a TCPIP service within a CICS system
	HOST URI Host
|	**IPFACIL** IPIC connection sessions
	PIPELINE Pipelines
	TCPIPGBL TCP/IP global statistics
	TCPIPS TCP/IP services
	URIMAP URI maps
	URIMPGBL URI map global statistics
	WEBSERV Web services
OPERATE.TDQUEUE	**CRESTDQ** Describe an instance of a transient data queue within a CICS system
	EXTRATDQ Extrapartition transient data queues
	INDTDQ Indirect transient data queues
	INTRATDQ Intrapartition transient data queues
	REMTDQ Remote transient data queues
	TDQGBL Intrapartition transient data queue usage
OPERATE.TERMINAL	**CRESTERM** Describe an instance of a terminal within a CICS system
	TERMNL Terminals

Table 41. Function and type combinations for resources accessible via CICSPlex SM (continued)

Function.Type	Resource Table Name and Usage
OPERATE.TRAN	**CRESTRAN** Describe an instance of a transaction within a CICS system
	LOCTRAN Local transactions
	REMTRAN Remote transactions
	TASKASSC Task association data
OPERATE.TSQUEUE	**CRESTSMD** Describe an instance of a temporary storage queue within a CICS system
	TSQUEUE Temporary storage queues
	TSMODEL Temporary storage models
	TSPOOL Temporary storage pools
	TSQGBL Global temporary storage queues
	TSQSHR Shared temporary storage queues
	TSQNAME Long temporary storage queues
OPERATE.UOW	**UOWDSNF** Display shunted units of work
	UOWENQ Display enqueues for executing units of work
	UOWLINK Display links for unit of work
	UOW Display executing units of work
TOPOLOGY.DEF	**CSYSGRP** Create, display, and maintain CICS system groups
	CSYSDEF Create, display, and maintain CICS systems
	CSGLCGCG Describe the link of a CICS system group to an outer system group
	CSGLCGCS Describe the link of a CICS system to a system group
	MAS Display CICS systems in a CICSplex
	MASSTAT Display CICS systems in a CICSplex by CMAS
	PERIODEF Display period definitions
	SYSLINK Display information about the links that exist between CICS systems.

Table 41. Function and type combinations for resources accessible via CICSPlex SM (continued)

Function.Type	Resource Table Name and Usage
WORKLOAD.DEF	**DTRINGRP** Display transactions in transaction groups
	LNKSWSCG Describe the link between a CICS system group and a workload specification
	LNKSWSCS Describe the link between a CICS system and a workload specification
	TRANGRP Create, display, and maintain transaction groups
	WLMATAFF Display and discard active affinities
	WLMATGRP Display and discard active transaction groups
	WLMATRAN Display and discard transaction directory
	WLMAWAOR Display active AORs in a workload
	WLMAWDEF Display and discard active workload definitions
	WLMAWORK Display active workloads
	WLMAWTOR Display active AORs in a workload
	WLMDEF Create, display, and maintain workload definitions
	WLMGROUP Create, display, and maintain workload groups
	WLMINGRP Display workload definitions in groups
	WLMINSPC Display workload groups in workload specifications
	WLMSPEC Create, display, and maintain workload specifications

Activating simulated CICS security

When you create RACF profiles using the CICSPlex SM resource classes to permit access to the operations and monitoring views, CICSPlex SM determines which views a user can access. However, CICSPlex SM cannot determine if that user is authorized to access the CICS resources represented within the view.

You can enhance the security provided by your CICSPlex SM profiles by activating *simulated CICS security checking*. Simulated security uses your existing RACF profiles to control access to CICS resources and/or CICS commands. It is available only for the operations and monitor views. When using this combination of profiles, your CICSPlex SM profiles determine which sets of views can be accessed and your CICS resource profiles determine which resources within the view can be accessed. For example, you can create a CICSPlex SM profile that allows a user to issue the file view commands and any associated action commands, and then have CICS simulated security determine which files the user is authorized to access.

To activate or deactivate simulated security checking, use the CSYSDEF view (for a single CICS system) or CPLEXDEF view (for multiple systems). You can indicate whether you want CICS resource checking, CICS command checking, or both, to occur. CICS resource checking controls which resources are displayed in a view.

CICS command checking controls what commands can be used within the view. The CSYSDEF and CPLEXDEF views are described in *CICSPlex System Manager Administration*.

To activate or deactivate simulated security checking temporarily for an active CICS system, use the MAS view (as described in *CICSPlex System Manager Administration*).

Note:

1. Refer to “Activating security parameters” on page 342 for important information on how the CICSPlex SM and CICS security parameters can affect simulated security.
2. Simulated security involves significantly more processing overhead than using only CICSPlex SM profiles and will have a negative impact on performance.

Exempting users and resources from security checking

There might be certain individuals who do not require security checking. There might also be certain CICS resources that are sufficiently protected by CICSPlex SM profiles and, therefore, do not need to be involved in security checking.

You can exempt these individuals and resources from simulated CICS security checking using the CICSPlex SM CPSMXMP resource class. Exemption bypasses only the simulated CICS security checks, not the basic CICSPlex SM resource checks.

For example, if a user does not have RACF authority to issue the CICS command `CEMT INQ FILE`, you can enable that user to achieve the same result by creating a profile in the exemption class that allows the user to issue the equivalent CICSPlex SM command **`LOCFILE`**.

To create exemption profiles:

1. Decide which resource you want to exempt and specify this on the **`PERMIT`** command. Use the resource name format described in “Specifying CICSPlex SM resource names in profiles” on page 324.
2. Specify the class name CPSMXMP. This RACF class controls exemption from simulated security checking.
3. Specify the type of access that you require.
   - If you want to bypass the security checking completely, specify `ACCESS(READ)`.
   - If you do not want to bypass security checking, specify `ACCESS(NONE)`.
   - If you want CICS to perform exemption checking up to the level of access that is indicated in the PROFILE, you can use `ACCESS(UPDATE)` to provide access to INQUIRE, SET and PERFORM level commands or `ACCESS(ALTER)` which would provide access to DISCARD command.
4. Specify the user or the group that you want the exemptions to apply to.

The following example shows how you could define an exemption profile that allows the individuals comprising the group EYUGRP2 to bypass security checking for all views and action commands associated with the TERMINAL type within the MONITOR function, when the context is EYUPLX01 and the scope is EYUMAS1A:

```
PERMIT  MONITOR.TERMINAL.EYUPLX01.EYUMAS1A  /* Resource name    */+
        CLASS(CPSMXMP)                      /* Class name       */+
        ACCESS(UPDATE)                      /* Access           */+
        ID(EYUGRP2)                         /* User or group    */+
                                            /*  granted access  */
```

Activating security parameters

To activate security for CICSPlex SM, you must:

- Specify the CICSPlex SM parameter SEC in the EYUPARM data set or member defined in the JCL used to start the CMAS and MAS, as described in the *CICS Transaction Server for z/OS Installation Guide*.
- Specify the CICS parameter SEC= in the CICS system initialization parameters used to start the MAS, as described in the*CICS Transaction Server for z/OS Installation Guide*.

Together these parameters determine what security checking is performed. Table 42 explains the possible parameter combinations.

Table 42. Parameters controlling security checking

CMAS (CICSPlex SM parameter)	MAS (CICS system initialization parameter)	Explanation
SEC(YES)	SEC=YES	Both view selection checking and simulated security checking can occur, depending on the settings in the CICSSYS and CPLEXDEF views. This means that after CICSPlex SM determines whether a user can display a particular view, simulated security determines what information can be provided in the view.
SEC(YES)	SEC=NO	View selection occurs; simulated security checking **does not** occur even if it is requested in the CICSSYS or CPLEXDEF views. This means that after CICSPlex SM determines whether a user can display a designated view, no simulated security checking is performed to determine what information is to be provided in that view.
SEC(NO)	SEC=YES	CICSPlex SM does not allow the MAS to connect to the CMAS. This prevents a MAS that has requested security from connecting to a CMAS that cannot provide security.

Note: CICSPlex SM honors any of the system initialization parameters XCMD, XDB2, XDCT, XFCT, XHFS, XJCT, XPCT, XPPT, and XRES; that is, CICSPlex SM includes or excludes the designated commands and resources from security checking. For each MAS, you can specify YES, NO, or CLASS NAME for these system initialization parameters. However, for the CMAS, you *must* specify NO for each of these system initialization parameters.

Refreshing RACF profiles for CICSPlex SM

CICSPlex SM uses a cached copy of RACF information in order to reduce unnecessary I/O to the RACF database. When you change a RACF definition, you will, in some situations, need to force the CMAS to refresh the cached information.

Refreshing general resource profiles in the cache

A CMAS requests RACF to create a cached copy of its general resource profiles during initialization:

- During CMAS initialization, global copies of the RACF profiles in the CPSMOBJ (including GCPSMOBJ) and CPSMXMP are created.
- During MAS initialization, the MAS determines which CICS security classes are in use, using the information specified in the XCMD, XDB2, XDCT, XFCT, XJCT, XPCT and XPPT system initialization parameters. This information is passed to the CMAS where it is used to create global copies of the relevant RACF profiles.

Perform the following steps to refresh the general resource profiles in the cache.

1. To identify the profiles for which global copies exist, issue the RACF command **SETROPTS LIST**. The "GLOBAL=YES RACLIST ONLY" section of the output shows which general resource classes have been globally copied.
2. Whenever a change is made to one of these classes (for example, with the **RALTER**, **RDEFINE**, **RDELETE** or **PERMIT** commands), you must refresh the cache. Use the following RACF command:

   ```
   SETR RACLIST(classname) GENERIC(classname) REFRESH
   ```

   In a multi-CMAS environment, this command will refresh the cache only on the MVS images that share the same RACF database.
3. If other CMASes run on other MVS images that do not share the same RACF database, repeat the **SETR** command on the other images.

Refreshing user profiles in the cache

A CMAS stores security information for a userid in the cache when it performs a security check. By default, the information is retained in the cache until the user has remained inactive in the CMAS for the time specified by the **SECTIMEOUT** CICSPlex SM parameter, and the CMAS has performed a userid timeout check.

Whenever a change is made to a userid (for example, with the **CONNECT**, **REMOVE**, **ALTUSER**, or **DELUSER** commands), the change becomes visible when the CMAS discards security information for the user form the cache.

- To force a CMAS to discard security information from the cache before timeout processing has occurred, use one the following actions.
  - For a single CMAS, use the **RESET USERID** action on the CMAS object.
  - For more than one CMAS, use the **RESET USERID** action on the CMASLIST object.

  Both actions are available from the WUI and the API. Perform the action when a user has previously used the CMAS and you do not want to wait for normal timeout processing to occur.
- To force a CMAS to immediately check for userids that are eligible for timeout, use one of the following actions.
  - For a single CMAS, use the **PURGE** action on the CMAS object.
  - For more than one CMAS, use the **PURGE** action on the CMASLIST object.

  Both actions are available from the WUI and the API.

CICSPlex SM security checking sequence

A user can issue a single CICSPlex SM command that causes data to be gathered about or an action to be performed against one or more CICS systems comprising a CICSplex. These CICS systems can reside in different MVS images.

When a user issues a request, the request is directed to the CMAS that manages the target CICS system or systems. Figure 37 on page 346 and Figure 38 on page 347 are flowcharts showing the procedure followed by CICSPlex SM to evaluate the security requirements of a request from a user. Here is a description of that procedure:

1. CICSPlex SM determines whether CICSPlex SM rules allow the request to be processed.
   - If not, CICSPlex SM terminates the request and issues an error message.
2. CICSPlex SM determines whether simulated CICS security checking is to be performed.
   - If not, processing continues at 9.
3. CICSPlex SM determines whether the user is exempt from simulated CICS security checking.
   - If so, processing continues at 9.
4. CICSPlex SM determines whether simulated CICS command checking is to be performed.
   - If not, processing continues at 6.
5. CICSPlex SM determines whether the user is allowed to process the command.
   - If not, CICSPlex SM terminates the request and issues an error message.
6. CICSPlex SM determines whether the request is an action (not a request for information).
   - If not, processing continues at 9.
7. CICSPlex SM determines whether simulated CICS resource checking is to be performed.
   - If not, processing continues at 9.
8. CICSPlex SM determines whether the user is allowed access to information about the resource.
   - If not, CICSPlex SM terminates the request and issues an error message.
9. CICSPlex SM performs the action or gets the information.
10. CICSPlex SM determines whether the request is an action (not a request for information).
    - If so, CICSPlex SM returns the results of the action.
11. CICSPlex SM determines whether simulated CICS security checking is to be performed.
    - If not, CICSPlex SM returns the requested information in the appropriate view.
12. CICSPlex SM determines whether the user is exempt from simulated CICS security checking.
    - If so, CICSPlex SM returns the requested information in the appropriate view.
13. CICSPlex SM determines whether simulated CICS resource checking is to be performed.

- If not, CICSPlex SM returns the requested information in the appropriate view.

14. CICSPlex SM determines whether the user is allowed access to information about the resource.
    - If not, CICSPlex SM excludes the requested information from the appropriate view.
15. CICSPlex SM determines whether information for another resource is requested.
    - If so, processing continues at 14.
    - Otherwise, CICSPlex SM returns the requested information in the appropriate view.

No further security checking is required.

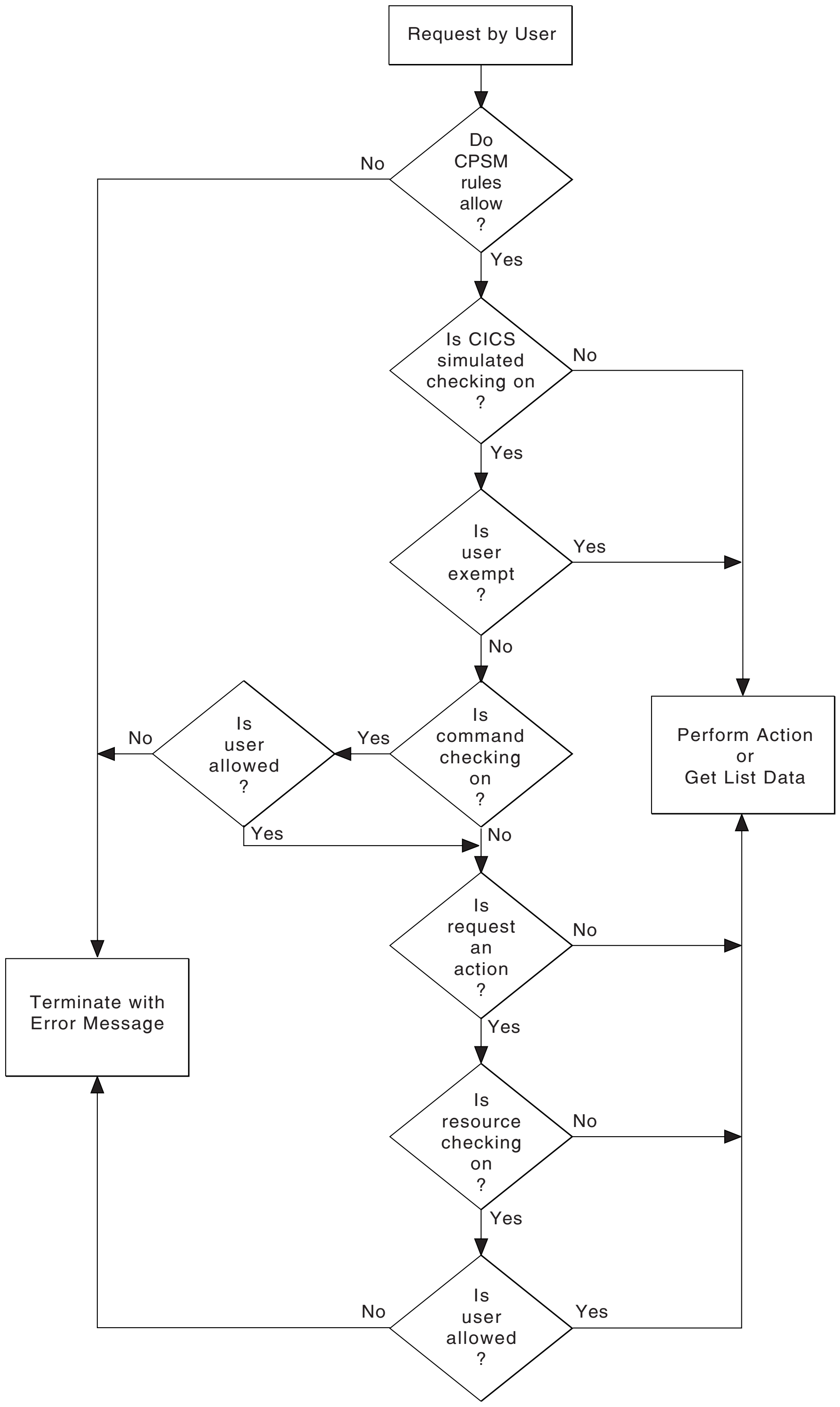


Figure 37. Flowchart of CICSPlex SM security checking sequence - part 1

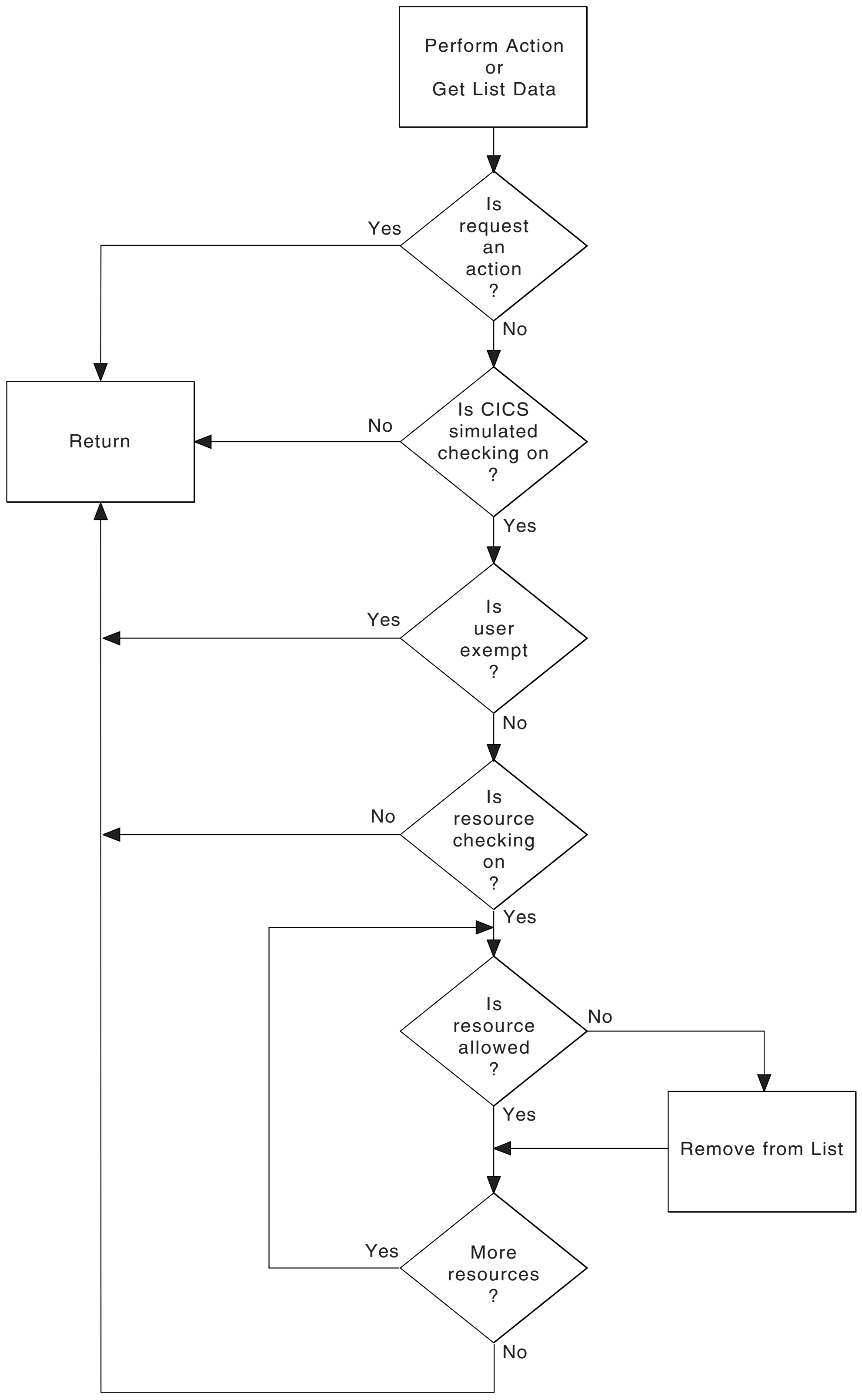


Figure 38. Flowchart of CICSPlex SM security checking sequence - part 2

Chapter 24. Invoking a user-supplied external security manager

CICSPlex SM provides an interface to an external security manager (ESM), which can be user-supplied or can be Resource Access Control Facility (RACF).

Be aware that upon return from any user-supplied program, CICSPlex SM must always receive control in primary-space translation mode, with the original contents of all access registers restored, and with all general-purpose registers restored (except for those which provide return codes or linkage information). For information about translation modes, refer to the *IBM ESA/390 Principles of Operation* manual.

Note: This information is intended primarily for non-RACF users. For information about security processing using RACF, refer to Chapter 23, “Implementing CICSPlex SM security,” on page 317.

An overview of the CICSPlex SM-ESM interface

CICSPlex SM security uses, via the RACROUTE macro, the MVS system authorization facility (SAF) interface to route authorization requests to the ESM. Normally, if RACF is present, the MVS router passes control to it. However, you can modify the action of the MVS router by invoking the router exit. The router exit can be used, for example, to pass control to a user-supplied or vendor-supplied ESM. (If you want to use your own security manager, you must supply an MVS router exit routine.)

The control points at which CICSPlex SM issues a RACROUTE macro to route authorization requests are described in “CICSPlex SM security control points” on page 351.

Using the MVS router

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 resource control. The resource-managing components and subsystems (such as CICSPlex SM) 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 MVS router exit

The MVS router provides an optional installation exit that is invoked whether or not RACF is installed and active on the system. If your installation does not use RACF, you can use the router exit to pass control to your own ESM. If you do use RACF, you could use the exit for preprocessing before RACF is invoked.

The MVS router exit routine is invoked whenever CICSPlex SM (or another component of your system) issues a RACROUTE macro. The router passes a parameter list (generated by the RACROUTE macro) to the exit routine. In addition, the exit receives the address of a 150-byte work area.

On entry to the exit routine, register 1 contains the address of the area described in Table 43.

Table 43. Area addressed by register 1, on entry to exit routine

Offset	Length	Description
0	4	**Parameter list address:** points to the MVS router parameter list. (See “The MVS router parameter list.”)
4	4	**Work area address:** points to a 150-byte work area that the exit can use.

The exit must be named ICHRTX00 and must be located in the link pack area (LPA).

The MVS router parameter list

The MVS router parameter list is generated when the RACROUTE macro is issued, and describes the security processing request by providing the request type. If the router exit routine exists, the router passes the parameter list to this exit. (If it does not exist, and if RACF is active, the router passes the parameter list to the RACF router.)

You can map the MVS router parameter list using the ICHSAFP macro. Its format is shown in the *MVS/ESA Diagnosis: Data Areas* manual.

Router exit return codes

Your exit routine must return a return code in register 15. The hexadecimal values of the return code are shown in Table 44.

Table 44. MVS router exit return codes

Code	Meaning
0	The exit has completed successfully. Control proceeds to the RACF front-end routine for further security processing and an invocation of RACF.
C8	The exit has completed successfully. The MVS router translates this return code to a router return code of '0' and returns control to the issuer of the RACROUTE macro (CICSPlex SM), bypassing RACF processing. (See the next section.)
CC	The exit has completed successfully. The MVS router translates this return code to a router return code of '4' and returns control to CICSPlex SM, bypassing RACF processing. (See the next section.)
D0	The exit has completed successfully. The MVS router translates this return code to a router return code of '8' and returns control to CICSPlex SM, bypassing RACF processing. (See the next section.)
Other	If the exit routine sets any return code other than those described above, the MVS router returns control directly to CICSPlex SM and passes the untranslated code as the router return code. Further RACF processing is bypassed.

Passing control to a user-supplied ESM

Normally, a caller (such as CICSPlex SM) invokes the MVS router and passes it request type, requester, and subsystem parameters via the RACROUTE exit parameter list. Using these parameters, the MVS router calls the router exit which, on completing its processing, passes a return code to the router. If the return code is '0', as defined above, the router invokes RACF. RACF reports the result of that invocation to the router by entering return and reason codes in register 15 and register 0 respectively. The router converts the RACF return and reason codes to router return and reason codes and passes them to the caller. The router provides additional information to the caller by placing the unconverted RACF return and reason codes in the first and second words of the router input parameter list.

If your installation does not use RACF, you can make the MVS router exit pass control to an alternative ESM. However, if you do so you must still provide CICSPlex SM with the RACF return and reason codes that it expects to receive. You set the router exit return code, as defined in Table 44 on page 350, so that RACF is not invoked; and you simulate the results of a RACF invocation by coding the exit so that it places the RACF return and reason codes in the first and second fullwords of the router input parameter list. RACF return and reason codes are documented in the *MVS/ESA Authorized Assembler Programming Reference* manual.

CICSPlex SM security control points

All RACROUTE macros are issued from a CMAS. Macros required to support simulated CICS security checking are issued from the CMAS to which the target MAS is connected.

The following list summarizes the RACROUTE macros used by CICSPlex SM to invoke the ESM, and the control points at which they are issued.

RACROUTE
The “front end” to the macros described below, it invokes the MVS router. If RACF is not present on the system, RACROUTE can route to an alternative ESM, via the MVS router exit.

RACROUTE REQUEST=VERIFY
Issued at user signon (with the parameter ENVIR=CREATE), and at user sign-off (with parameter ENVIR=DELETE) to a CMAS. For ISPF end-user interface requests, signon calls are made during window creation in the CMAS that supports the named context. Sign-off calls are made when the window is closed. This macro creates or destroys an access control environment element (ACEE). It is issued at the following CICSPlex SM CMAS control points:
- ISPF end-user interface user connection to a CMAS
- API CONNECT thread creation
- Single system image command routing
- ISPF end-user interface user disconnect from a CMAS
- API DISCONNECT thread termination

RACROUTE REQUEST=FASTAUTH
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, and is issued at the following CICSPlex SM CMAS control points:
- Simulated CICS security checking
- View selection / API security

RACROUTE REQUEST=AUTH
This is a higher path length form of resource checking and is issued during CAS / PLEXMGR security checking. It may also be called to perform logging and auditing after a REQUEST=FASTAUTH.

RACROUTE REQUEST=LIST
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 CICSPlex SM CMAS control points:
- When CICSPlex SM security is being initialized for a MAS
- When the CMAS or CMASD sceurity action command (SEC) is issued.

For a detailed description of these macros, see the *z/OS Security Server RACF Macros and Interfaces* manual.

Chapter 25. Example tasks: security

This topic provides examples of typical security setup tasks that you can use as a model for your own.

Here are some general points that apply to all of the RACF examples:

- Each RACF command shown in these task examples must be issued once against every RACF database in your CICSPlex SM configuration. So, if there are two unconnected RACF databases, one on MVS1 and one on MVS2, each RACF command must be issued twice (once on each system).
- In all of the RACF command examples, strings in lowercase must be replaced by values suitable for your own enterprise. For example, you must replace the string `admin_user` with the USERID of the administrator responsible for security of the relevant CICSPlex SM resources.
- All of the RACF task examples use the enhanced generic naming facility (**) of RACF. If you don't use this at your enterprise, see the RACF documentation for information about creating equivalent profiles.
- Operations and administration RACF groups have been used in these examples: we recommend that you create such groups.

We're going to start by creating some RACF profiles to protect all CICSPlex SM functions and resources. When we've done this, we'll permit access selectively to particular users of particular resources.

Example: Protecting all CICSPlex SM resources

To create the RACF profile to protect all CICSPlex SM resources, do the following:

1. Ensure that the CPSMOBJ class is active and that generic profiles can be defined:
   ```
   SETROPTS CLASSACT(CPSMOBJ) GENERIC(CPSMOBJ)
   ```
2. Create a RACF profile to protect all views and action commands for all CICSPlex SM functions:
   ```
   RDEF CPSMOBJ ** UACC(NONE) OWNER(admin_group) NOTIFY(admin_user)
   ```
   This command defines a profile (**) that RACF treats as matching all `CPSMOBJ` resource entity names, and which therefore protects all CICSPlex SM resources; it also specifies that `admin_user` is to be notified of any violations.
3. The next step is very similar to Step 2: we define one RACF profile for each CICSplex in the configuration. Each profile will protect all CICSPlex SM functions and resources for that CICSplex. The purpose of doing this is to give you more flexibility in granting access to CICSplex-specific resources. In this example, we have two CICSplexes, and so create two RACF profiles:
   ```
   RDEF CPSMOBJ *.*.PLXPROD1.* UACC(NONE) OWNER(admin_group) +
        NOTIFY(admin_user)
   RDEF CPSMOBJ *.*.PLXPROD2.* UACC(NONE) OWNER(admin_group) +
        NOTIFY(admin_user)
   ```
   Note that you can't replace Step 2 with multiple CICSplex-specific profiles: such profiles won't necessarily protect CICSplexes that you create later, nor can they protect CICSPlex SM functions whose context is the CMAS rather than the CICSplex. For example, the CONFIG views would be left unprotected if you didn't also perform Step 2.
4. In Step 3 we protected all CICSPlex SM functions and resources at the CICSplex level. In this step, we're going to define profiles to control access to

the CICSPlex SM CONFIG and TOPOLOGY definition functions, so that we can selectively permit any “special” users, such as administrators, the access they need. (Anyone who has update access to these two functions can alter the CICSplex configuration, and so access must be limited.)

```
RDEF CPSMOBJ CONFIG.DEF.** UACC(NONE) OWNER(admin_group)
RDEF CPSMOBJ TOPOLOGY.DEF.** UACC(NONE) OWNER(admin_group)
```

Now that we've controlled access to CICSPlex SM functions and resources, we can begin to grant access to particular users or groups of users.

Example: Giving CICSPlex SM operators appropriate authorizations

CICSPlex SM operators need access, at least, to all of the operations views. In this example, we'll show you how to give CICSPlex SM operators update access to all operations views and read access to the monitoring views. This will allow operators to look at monitor data, but not to create or change monitor definitions.

1. Give CICSPlex SM operators update access to the OPERATE views:
   ```
   RDEF CPSMOBJ OPERATE.** OWNER(admin_group) UACC(NONE)
   PE OPERATE.** CLASS(CPSMOBJ) ID(ops_group) A(UPDATE)
   ```
2. Give CICSPlex SM operators read access to the MONITOR views:
   ```
   RDEF CPSMOBJ MONITOR.** UACC(NONE) OWNER(admin_group)
   PE MONITOR.** CLASS(CPSMOBJ) ID(ops_group) A(READ)
   ```

In both steps, you can see that we begin by creating a RACF profile to protect the resource, and then grant access to users in group `ops_group`.

Example: Giving a user read access to all transactions on MVS system A

In this example, we show you how to give user PAYUSR1 read access to all transactions running on CICS systems on MVS system A. In the example, we have three CICS systems (say, CICSAA01, CICSAA02, and CICSAA03) which all belong to CICSplex PLXPROD1.

1. Define the appropriate RACF profile:
   ```
   RDEF CPSMOBJ OPERATE.TRAN.PLXPROD1.CICSAA0* UACC(NONE) +
         OWNER(admin_group)
   ```
2. Give user PAYUSR1 read access to all transactions on MVS system A:
   ```
   PE OPERATE.TRAN.PLXPROD1.CICSAA0* CLASS(CPSMOBJ) I(PAYUSR1) A(READ)
   ```

Part 9. Appendixes

Appendix A. 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.

Specify a language request is specified in the following RACF command:

```
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 45.

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 78.

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

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

In CICS, you can use only the languages listed in Table 45. Languages other than ENU, CHNS, and JPN are available only if you provide translated message tables for them, using the message editing utility program, and then specify the CICS language suffix in the NATLANG system initialization parameter. See the *CICS Operations and Utilities Guide* for information on creating translated message tables.

Table 45. 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

Table 45. CICS language suffixes (continued)

Suffix	IBM Code	Language name
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 B. Resource and command check cross reference

This topic provides a complete command and resource check cross reference.

Table 46. Resource and command check cross reference

EXEC CICS COMMAND	**Resource Check**			**Check class=XCMD**	
	Class	**Access**	**Resource**	**Access**	**Resource**
ABEND					
ACQUIRE				UPDATE	TERMINAL
ACQUIRE for BTS (see note 9)	XFCT	UPDATE	BTS repository file		
ADDRESS					
ALLOCATE					
ASKTIME					
BIF DEEDIT					
BUILD ATTACH					
CANCEL (see note 1)	XPCT	READ	transid		
CANCEL for BTS	XFCT	UPDATE	BTS repository file		
CHANGE PASSWORD					
CHANGE TASK					
COLLECT STATISTICS				READ	STATISTICS
COLLECT STATISTICS FILE	XFCT	READ	file	READ	STATISTICS
COLLECT STATISTICS JOURNALNAME	XJCT	READ	journal	READ	STATISTICS
COLLECT STATISTICS JOURNALNUM	XJCT	READ	journal	READ	STATISTICS
COLLECT STATISTICS PROGRAM	XPPT	READ	program	READ	STATISTICS
COLLECT STATISTICS TDQUEUE	XDCT	READ	tdqueue	READ	STATISTICS
COLLECT STATISTICS TRANSACTION	XPCT	READ	transid	READ	STATISTICS
CONNECT PROCESS					
CONVERSE					
CREATE CONNECTION (see note 2)				ALTER	CONNECTION
CREATE CORBASERVER				ALTER	CORBASERVER
CREATE DB2CONN (see note 3)		ALTER			DB2CONN
CREATE DB2ENTRY (see note 3)	XDB2	ALTER	db2entry	ALTER	DB2ENTRY

Table 46. Resource and command check cross reference (continued)

EXEC CICS COMMAND	Resource Check			Check class=XCMD	
	Class	Access	Resource	Access	Resource
CREATE DB2TRAN (see note 3)	XDB2	ALTER	db2tran	ALTER	DB2TRAN
CREATE DJAR				ALTER	DJAR
CREATE DOCTEMPLATE	XRES	ALTER	DOCTEMPLATE.doctemplate (see note 12)	ALTER	DOCTEMPLATE
CREATE ENQMODEL				ALTER	ENQMODEL
CREATE FILE	XFCT	ALTER	file	ALTER	FILE
CREATE IPCONN				ALTER	IPCONN
CREATE JOURNALMODEL				ALTER	JOURNALMODEL
CREATE LSRPOOL				ALTER	LSRPOOL
CREATE LIBRARY				ALTER	LIBRARY
CREATE MAPSET	XPPT	ALTER	mapset	ALTER	MAPSET
CREATE PARTITIONSET	XPPT	ALTER	partitionset	ALTER	PARTITIONSET
CREATE PIPELINE				ALTER	PIPELINE
CREATE PARTNER				ALTER	PARTNER
CREATE PROCESSTYPE (see note 10)				ALTER	PROCESSTYPE
CREATE PROFILE				ALTER	PROFILE
CREATE PROGRAM	XPPT	ALTER	program	ALTER	PROGRAM
CREATE REQUESTMODEL				ALTER	REQUESTMODEL
CREATE SESSIONS (see note 3)				ALTER	SESSIONS
CREATE TCPIPSERVICE				ALTER	TCPIPSERVICE
CREATE TDQUEUE (see note 3)	XDCT	ALTER	tdqueue	ALTER	TDQUEUE
CREATE TERMINAL (see note 3)				ALTER	TERMINAL
CREATE TRANCLASS				ALTER	TCLASS
CREATE TRANSACTION	XPCT	ALTER	transid	ALTER	TRANSACTION
CREATE TSMODEL				ALTER	TSMODEL
CREATE TYPETERM				ALTER	TYPETERM
CREATE URIMAP				ALTER	URIMAP
CREATE WEBSERVICE				ALTER	WEBSERVICE
DEFINE ACTIVITY (see notes 7 and 9	XFCT	UPDATE	BTS repository file		

Table 46. Resource and command check cross reference (continued)

EXEC CICS COMMAND	**Resource Check**			**Check class=XCMD**	
	Class	**Access**	**Resource**	**Access**	**Resource**
DEFINE PROCESS (see notes 7 and 9)	XFCT	UPDATE	BTS repository file		
DELAY					
DELETE	XFCT	UPDATE	file		
DELETE ACTIVITY (see note 9)	XFCT	UPDATE	BTS repository file		
DELETEQ TD	XDCT	UPDATE	tdqueue		
DELETEQ TS	XTST	UPDATE	tsqueue		
DEQ					
DISABLE PROGRAM	XPPT	UPDATE	program	UPDATE	EXITPROGRAM
DISCARD AUTINSTMODEL				ALTER	AUTINSTMODEL
DISCARD CONNECTION					
DISCARD CORBASERVER				ALTER	CORBASERVER
DISCARD DB2CONN				ALTER	DB2CONN
DISCARD DB2ENTRY	XDB2	ALTER	db2entry	ALTER	DB2ENTRY
DISCARD DB2TRAN	XDB2	ALTER	db2tran	ALTER	DB2TRAN
DISCARD DJAR				ALTER	DJAR
DISCARD DOCTEMPLATE	XRES	ALTER	DOCTEMPLATE.doctemplate (see note 12)	ALTER	DOCTEMPLATE
DISCARD ENQMODEL				ALTER	ENQMODEL
DISCARD FILE	XFCT	ALTER	file	ALTER	FILE
DISCARD IPCONN				ALTER	IPCONN
DISCARD JOURNALMODEL				ALTER	JOURNALMODEL
DISCARD JOURNALNAME	XJCT	ALTER	journal	ALTER	JOURNALNAME
DISCARD LIBRARY				ALTER	LIBRARY
DISCARD PARTNER				ALTER	PARTNER
DISCARD PIPELINE				ALTER	PIPELINE
DISCARD PROCESSTYPE (see note 10)				ALTER	PROCESSTYPE
DISCARD PROFILE				ALTER	PROFILE
DISCARD PROGRAM	XPPT	ALTER	program	ALTER	PROGRAM
DISCARD REQUESTMODEL				ALTER	REQUESTMODEL
DISCARD TCPIPSERVICE				ALTER	TCPIPSERVICE
DISCARD TDQUEUE	XDCT	ALTER	tdqueue	ALTER	TDQUEUE

Table 46. Resource and command check cross reference (continued)

EXEC CICS COMMAND	**Resource Check**			**Check class=XCMD**	
	Class	**Access**	**Resource**	**Access**	**Resource**
DISCARD TERMINAL				ALTER	TERMINAL
DISCARD TRANCLASS				ALTER	TCLASS
DISCARD TRANSACTION	XPCT	ALTER	transid	ALTER	TRANSACTION
DISCARD TSMODEL				ALTER	TSMODEL
DISCARD URIMAP				ALTER	URIMAP
DISCARD WEBSERVICE				ALTER	WEBSERVICE
DOCUMENT CREATE	XRES	READ	DOCTEMPLATE.doctemplate (see note 12)		
DOCUMENT INSERT	XRES	READ	DOCTEMPLATE.doctemplate (see note 12)		
DUMP TRANSACTION					
ENABLE PROGRAM	XPPT	UPDATE	program	UPDATE	EXITPROGRAM
ENDBR (see note 5)					
ENQ					
ENTER TRACENUM					
EXTRACT					
EXTRACT EXIT	XPPT	READ	program	UPDATE	EXITPROGRAM
EXTRACT STATISTICS				READ	STATISTICS
FEPI					FEPI
FORMATTIME					
FREE					
FREEMAIN					
GDS					
GETMAIN					
HANDLE ABEND PROGRAM	XPPT	READ	program		
HANDLE AID					
HANDLE CONDITION					
IGNORE CONDITION					
| INQUIRE ASSOCIATION				READ	ASSOCIATION
INQUIRE ACTIVITYID (see note 9)	XFCT	READ	BTS repository file		
INQUIRE AUTINSTMODEL				READ	AUTINSTMODEL
INQUIRE AUTOINSTALL				READ	AUTOINSTALL
INQUIRE BEAN				READ	BEAN
INQUIRE BRFACILITY				READ	BRFACILITY

Table 46. Resource and command check cross reference (continued)

EXEC CICS COMMAND	Resource Check			Check class=XCMD	
	Class	Access	Resource	Access	Resource
INQUIRE CFDTPOOL				READ	CFDTPOOL
INQUIRE CLASSCACHE				READ	CLASSCACHE
INQUIRE CONNECTION				READ	CONNECTION
INQUIRE CORBASERVER				READ	CORBASERVER
INQUIRE CONTAINER (see note 9)	XFCT	READ	BTS repository file		
INQUIRE DB2CONN				READ	DB2CONN
INQUIRE DB2ENTRY	XDB2	READ	db2entry	READ	DB2ENTRY
INQUIRE DB2TRAN	XDB2	READ	db2tran	READ	DB2TRAN
INQUIRE DELETSHIPPED				READ	DELETSHIPPED
INQUIRE DJAR				READ	DJAR
INQUIRE DOCTEMPLATE	XRES	READ	DOCTEMPLATE.doctemplate (see note 12)	READ	DOCTEMPLATE
INQUIRE DSNAME				READ	DSNAME
INQUIRE DUMPDS				READ	DUMPDS
INQUIRE ENQMODEL				READ	ENQMODEL
INQUIRE EXCI				READ	EXCI
INQUIRE EVENT (see note 9)	XFCT	READ	BTS repository file		
INQUIRE EXITPROGRAM	XPPT	READ	program	READ	EXITPROGRAM
INQUIRE FILE	XFCT	READ	file	READ	FILE
INQUIRE HOST				READ	HOST
INQUIRE IPCONN				READ	IPCONN
INQUIRE IRC				READ	IRC
INQUIRE JOURNALMODEL				READ	JOURNALMODEL
INQUIRE JOURNALNAME	XJCT	READ	journal	READ	JOURNAL
INQUIRE JVM				READ	JVM
INQUIRE JVMPOOL				READ	JVMPOOL
INQUIRE JVMPROFILE				READ	JVMPROFILE
INQUIRE LIBRARY				READ	LIBRARY
INQUIRE MODENAME				READ	MODENAME
INQUIRE MONITOR				READ	MONITOR
INQUIRE MVSTCB				READ	MVSTCB
INQUIRE NETNAME				READ	TERMINAL

Table 46. Resource and command check cross reference (continued)

EXEC CICS COMMAND	**Resource Check**			**Check class=XCMD**	
	Class	**Access**	**Resource**	**Access**	**Resource**
INQUIRE PARTNER				READ	PARTNER
INQUIRE PIPELINE				READ	PIPELINE
INQUIRE PROCESS (see note 9)	XFCT	READ	BTS repository file		
INQUIRE PROCESSTYPE (see note 10)	XPTT			READ	PROCESSTYPE
INQUIRE PROFILE				READ	PROFILE
INQUIRE PROGRAM	XPPT	READ	program	READ	PROGRAM
INQUIRE REQID (see note 8)	XPCT	READ	transid	READ	REQID
INQUIRE REQUESTMODEL				READ	REQUESTMODEL
INQUIRE RRMS				READ	RRMS
INQUIRE STATISTICS				READ	STATISTICS
INQUIRE STORAGE				READ	STORAGE
INQUIRE STREAMNAME				READ	STREAMNAME
INQUIRE SUBPOOL				READ	SUBPOOL
INQUIRE SYSDUMPCODE				READ	SYSDUMPCODE
INQUIRE SYSTEM				READ	SYSTEM
INQUIRE TASK				READ	TASK
INQUIRE TCLASS				READ	TCLASS
INQUIRE TCPIP				READ	TCPIP
INQUIRE TCPIPSERVICE				READ	TCPIPSERVICE
INQUIRE TDQUEUE	XDCT	READ	tdqueue	READ	TDQUEUE
INQUIRE TERMINAL				READ	TERMINAL
INQUIRE TIMER (see note 9)	XFCT	READ	BTS repository file		
INQUIRE TRACEDEST				READ	TRACEDEST
INQUIRE TRACEFLAG				READ	TRACEFLAG
INQUIRE TRACETYPE				READ	TRACETYPE
INQUIRE TRANCLASS				READ	TCLASS
INQUIRE TRANDUMPCODE				READ	TRANDUMPCODE
INQUIRE TRANSACTION	XPCT	READ	program	READ	TRANSACTION
INQUIRE TSMODEL				READ	TSMODEL
INQUIRE TSPOOL				READ	TSPOOL

Table 46. Resource and command check cross reference (continued)

EXEC CICS COMMAND	Resource Check			Check class=XCMD	
	Class	**Access**	**Resource**	**Access**	**Resource**
INQUIRE TSQUEUE	XTST	READ	tsqueue	READ	TSQUEUE
INQUIRE TSQNAME	XTST	READ	tsqname	READ	TSQUEUE
INQUIRE UOW				READ	UOW
INQUIRE UOWDSNFAIL				READ	UOWDSNFAIL
INQUIRE UOWENQ				READ	UOWENQ
INQUIRE UOWLINK				READ	UOWLINK
INQUIRE URIMAP				READ	URIMAP
INQUIRE VTAM				READ	VTAM
INQUIRE WEB				READ	WEB
INQUIRE WEBSERVICE				READ	WEBSERVICE
INQUIRE WORKREQUEST				READ	WORKREQUEST
ISSUE					
LINK	XPPT	READ	program		
LINK ACQPROCESS (see note 9)	XFCT	UPDATE	BTS repository file		
LINK ACTIVITY / ACQACTIVITY (see note 9)	XFCT	UPDATE	BTS repository file		
LOAD	XPPT	READ	program		
MONITOR					
PERFORM CLASSCACHE				UPDATE	CLASSCACHE
PERFORM CORBASERVER (see note 12)				UPDATE	CORBASERVER
PERFORM DELETSHIPPED				UPDATE	DELETSHIPPED
PERFORM DJAR				UPDATE	DJAR
PERFORM DUMP				UPDATE	DUMP
PERFORM PIPELINE				UPDATE	PIPELINE
PERFORM RESETTIME				UPDATE	RESETTIME
PERFORM SECURITY				UPDATE	SECURITY
PERFORM SHUTDOWN				UPDATE	SHUTDOWN
PERFORM STATISTICS				UPDATE	STATISTICS
POINT					
POP HANDLE					
POST					

Table 46. Resource and command check cross reference (continued)

EXEC CICS COMMAND	**Resource Check**			**Check class=XCMD**	
	Class	**Access**	**Resource**	**Access**	**Resource**
PURGE MESSAGE					
PUSH HANDLE					
QUERY SECURITY (see note 6)					
READ	XFCT	READ	file		
READ PREV (see note 5)					
READ NEXT (see note 5)					
READQ TD	XDCT	UPDATE	tdqueue		
READQ TS	XTST	READ	tsqueue		
RECEIVE					
RELEASE	XPPT	READ	program		
RESET ACTIVITY (see note 9)	XFCT	UPDATE	BTS repository file		
RESET ACQPROCESS (see note 9)	XFCT	UPDATE	BTS repository file		
RESETBR (see note 5)					
RESYNC ENTRYNAME				UPDATE	EXITPROGRAM
RETRIEVE					
RETURN / RETURN ENDACTIVITY (see note 9) (see note 11	XFCT	UPDATE	BTS repository file		
REWRITE	XFCT	UPDATE	file		
ROUTE					
RUN (see note 9) (see note 11)	XFCT	UPDATE	BTS repository file		
RUN / ASYNCH SYNC (see note 9)	XFCT	UPDATE	DFHLRQ file		
SEND					
SET AUTOINSTALL				UPDATE	AUTOINSTALL
SET BRFACILITY				UPDATE	BRFACILITY
SET CLASSCACHE				UPDATE	CLASSCACHE
SET CONNECTION				UPDATE	CONNECTION
SET CORBASERVER				UPDATE	CORBASERVER
SET DB2CONN				UPDATE	DB2CONN
SET DB2ENTRY	XDB2	UPDATE	db2entry	UPDATE	DB2ENTRY
SET DB2TRAN	XDB2	UPDATE	db2tran	UPDATE	DB2TRAN
SET DELETSHIPPED				UPDATE	DELETSHIPPED
SET DOCTEMPLATE	XRES	UPDATE	DOCTEMPLATE.doctemplate (see note 12)	UPDATE	DOCTEMPLATE

Table 46. Resource and command check cross reference (continued)

EXEC CICS COMMAND	Resource Check			Check class=XCMD	
	Class	**Access**	**Resource**	**Access**	**Resource**
SET DSNAME				UPDATE	DSNAME
SET DUMPDS				UPDATE	DUMPDS
SET ENQMODEL				UPDATE	ENQMODEL
SET FILE	XFCT	UPDATE	file	UPDATE	FILE
SET HOST				UPDATE	HOST
SET IPCONN				UPDATE	IPCONN
SET IRC				UPDATE	IRC
SET JOURNALNAME	XJCT	UPDATE	journal	UPDATE	JOURNAL
SET JVMPOOL				UPDATE	JVMPOOL
SET LIBRARY				UPDATE	LIBRARY
SET MODENAME				UPDATE	MODENAME
SET MONITOR				UPDATE	MONITOR
SET NETNAME				UPDATE	TERMINAL
SET PIPELINE				UPDATE	PIPELINE
SET PROCESSTYPE (see note 10)	XPTT			UPDATE	PROCESSTYPE
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 TCPIP				UPDATE	TCPIP
SET TCPIPSERVICE				UPDATE	TCPIPSERVICE
SET TDQUEUE (see note 3)	XDCT	UPDATE	tdqueue	UPDATE	TDQUEUE
SET TERMINAL				UPDATE	TERMINAL
SET TRACEDEST				UPDATE	TRACEDEST
SET TRACEFLAG				UPDATE	TRACEFLAG
SET TRACETYPE				UPDATE	TRACETYPE
SET TRANCLASS				UPDATE	TCLASS
SET TRANDUMPCODE				UPDATE	TRANDUMPCODE
SET TRANSACTION	XPCT	UPDATE	transid	UPDATE	TRANSACTION
SET TSQNAME	XTST	UPDATE	tsqueue	UPDATE	TSQUEUE
SET TSQUEUE	XTST	UPDATE	tsqname	UPDATE	TSQUEUE
SET UOW				UPDATE	UOW
SET UOWLINK				UPDATE	UOWLINK
SET URIMAP				UPDATE	URIMAP
SET VTAM				UPDATE	VTAM

Table 46. Resource and command check cross reference (continued)

EXEC CICS COMMAND	Resource Check			Check class=XCMD	
	Class	Access	Resource	Access	Resource
SET WEB				UPDATE	WEB
SET WEBSERVICE				UPDATE	WEBSERVICE
SET WORKREQUEST				UPDATE	WORKREQUEST
SIGNOFF					
SIGNON					
SPOOLCLOSE					
SPOOLOPEN					
SPOOLREAD					
SPOOLWRITE					
START (see note 7)	XPCT	READ	transid		
STARTBR	XFCT	READ	file		
STARTBROWSE ACTIVITY (see note 9)	XFCT	READ	BTS repository file		
STARTBROWSE CONTAINER (BTS) (see note 9)	XFCT	READ	BTS repository file		
STARTBROWSE EVENT (see note 9)	XFCT	READ	BTS repository file		
STARTBROWSE PROCESS (see note 9	XFCT	READ	BTS repository file		
SUSPEND (see note 9)	XFCT	UPDATE	BTS repository file		
SYNCPOINT					
TCPIP					
UNLOCK					
VERIFY PASSWORD					
WAIT					
WAIT JOURNALNAME	XJCT	READ	journal		
WAIT JOURNALNUM	XJCT	READ	journal		
WAITCICS					
WEB					
WRITE	XFCT	UPDATE	file		
WRITE JOURNALNAME	XJCT	UPDATE	journal		
WRITE JOURNALNUM	XJCT	UPDATE	DFHJnn		
WRITE OPERATOR					
WRITEQ TD	XDCT	UPDATE	tdqueue		
WRITEQ TS	XTST	UPDATE	tsqueueltsqname		
XCTL	XPPT	READ	program		

Note:

1. 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).
2. The **CREATE CONNECTION** command is subject to command security checking when you define a connection, for example; CREATE CONNECTION(con1) *Attribute*(...). However, when you use the **CREATE CONNECTION COMPLETE** or **CREATE CONNECTION DISCARD** command, no command security checking is performed unless you have been authorized to use COMPLETE and DISCARD. COMPLETE and DISCARD can only be used by those authorized to perform **CREATE CONNECTION(con1)** and **CREATE SESSIONS(ses1)** commands. Otherwise, ILLOGIC is returned.
3. An install surrogate user check can also occur.
4. No security check is performed, because the **STARTBR** command must be issued before this command and a security check is issued on the **STARTBR** command.
5. The **QUERY SECURITY** command is not controlled by resource or command checks, but it can cause them to be issued.
6. A start surrogate user check can also occur.
7. The resource check for the transid is only done if the reqid is associated with a transaction.
8. CICS business transaction services (BTS) application programming commands.
9. CICS business transaction services commands that are subject to command security. All other CICS business transaction services commands are not subject to command-level security.
10. Any BTS commands that use timing operands will access the BTS LRQ file
11. In addition to UPDATE access to the CORBASERVER resource, ALTER access to the associated DJAR resource is required for the **PERFORM CORBASERVER SCAN** command.
12. When calling the external security manager, CICS uses the name of the DOCTEMPLATE resource definition for the CICS document template, prefixed by its resource type, DOCTEMPLATE. For example, for a document template whose resource definition is named “WELCOME", the profile name passed to the security manager is DOCTEMPLATE.WELCOME. The EXEC CICS DOCUMENT CREATE and INSERT commands reference document templates using the 48-character name of the template (as specified in the TEMPLATENAME attribute of the DOCTEMPLATE resource definition). However, security checking for these commands uses the name of the DOCTEMPLATE resource definition that corresponds to the TEMPLATENAME attribute.

Appendix C. The sign-on table migration utility

The sign-on table migration utility, DFHSNMIG, is provided to help you migrate security information defined in an SNT to the CICS segment of a RACF user's profile. For each user entry in the SNT it creates a CLIST of RACF commands, generating either an ADDUSER or an ALTUSER command as appropriate for each SNT user entry. Because the DFHSNT macro is not supported in CICS TS, you must assemble your SNT using an earlier version of CICS.

DFHSNMIG can be found as an APF-authorized program in CICSTS32.CICS.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 78 for an example of specifying DFLTGRP on the ADDUSER command.

Figure 39 shows an example sign-on 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 39. Sample sign-on table entry

Figure 40 on page 373 shows an example of output from DFHSNMIG, which has changed the SNT shown in Figure 39 into entries for the RACF database.

```
/*-------------------------------------------------------*/
/*                                                       */
/* 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 40. Example of output from DFHSNMIG

Bibliography

The CICS Transaction Server for z/OS library

The published information for CICS Transaction Server for z/OS is delivered in the following forms:

The CICS Transaction Server for z/OS Information Center

The CICS Transaction Server for z/OS Information Center is the primary source of user information for CICS Transaction Server. The Information Center contains:

- Information for CICS Transaction Server in HTML format.
- Licensed and unlicensed CICS Transaction Server books provided as Adobe Portable Document Format (PDF) files. You can use these files to print hardcopy of the books. For more information, see "PDF-only books."
- Information for related products in HTML format and PDF files.

One copy of the CICS Information Center, on a CD-ROM, is provided automatically with the product. Further copies can be ordered, at no additional charge, by specifying the Information Center feature number, 7014.

Licensed documentation is available only to licensees of the product. A version of the Information Center that contains only unlicensed information is available through the publications ordering system, order number SK3T-6945.

Entitlement hardcopy books

The following essential publications, in hardcopy form, are provided automatically with the product. For more information, see "The entitlement set."

The entitlement set

The entitlement set comprises the following hardcopy books, which are provided automatically when you order CICS Transaction Server for z/OS, Version 3 Release 2:

Memo to Licensees, GI10-2559
CICS Transaction Server for z/OS Program Directory, GI13-0515
CICS Transaction Server for z/OS Release Guide, GC34-6811
CICS Transaction Server for z/OS Installation Guide, GC34-6812
CICS Transaction Server for z/OS Licensed Program Specification, GC34-6608

You can order further copies of the following books in the entitlement set, using the order number quoted above:

CICS Transaction Server for z/OS Release Guide
CICS Transaction Server for z/OS Installation Guide
CICS Transaction Server for z/OS Licensed Program Specification

PDF-only books

The following books are available in the CICS Information Center as Adobe Portable Document Format (PDF) files:

CICS books for CICS Transaction Server for z/OS

General

CICS Transaction Server for z/OS Program Directory, GI13-0515
CICS Transaction Server for z/OS Release Guide, GC34-6811
CICS Transaction Server for z/OS Migration from CICS TS Version 3.1, GC34-6858
CICS Transaction Server for z/OS Migration from CICS TS Version 1.3, GC34-6855
CICS Transaction Server for z/OS Migration from CICS TS Version 2.2, GC34-6856
CICS Transaction Server for z/OS Installation Guide, GC34-6812

Administration

CICS System Definition Guide, SC34-6813

CICS Customization Guide, SC34-6814
CICS Resource Definition Guide, SC34-6815
CICS Operations and Utilities Guide, SC34-6816
CICS Supplied Transactions, SC34-6817

Programming

CICS Application Programming Guide, SC34-6818
CICS Application Programming Reference, SC34-6819
CICS System Programming Reference, SC34-6820
CICS Front End Programming Interface User's Guide, SC34-6821
CICS C++ OO Class Libraries, SC34-6822
CICS Distributed Transaction Programming Guide, SC34-6823
CICS Business Transaction Services, SC34-6824
Java Applications in CICS, SC34-6825
JCICS Class Reference, SC34-6001

Diagnosis

CICS Problem Determination Guide, SC34-6826
CICS Messages and Codes, GC34-6827
CICS Diagnosis Reference, GC34-6862
CICS Data Areas, GC34-6863-00
CICS Trace Entries, SC34-6828
CICS Supplementary Data Areas, GC34-6864-00

Communication

CICS Intercommunication Guide, SC34-6829
CICS External Interfaces Guide, SC34-6830
CICS Internet Guide, SC34-6831

Special topics

CICS Recovery and Restart Guide, SC34-6832
CICS Performance Guide, SC34-6833
CICS IMS Database Control Guide, SC34-6834
CICS RACF Security Guide, SC34-6835
CICS Shared Data Tables Guide, SC34-6836
CICS DB2 Guide, SC34-6837
CICS Debugging Tools Interfaces Reference, GC34-6865

CICSPlex SM books for CICS Transaction Server for z/OS

General

CICSPlex SM Concepts and Planning, SC34-6839
CICSPlex SM User Interface Guide, SC34-6840
CICSPlex SM Web User Interface Guide, SC34-6841

Administration and Management

CICSPlex SM Administration, SC34-6842
CICSPlex SM Operations Views Reference, SC34-6843
CICSPlex SM Monitor Views Reference, SC34-6844
CICSPlex SM Managing Workloads, SC34-6845
CICSPlex SM Managing Resource Usage, SC34-6846
CICSPlex SM Managing Business Applications, SC34-6847

Programming

CICSPlex SM Application Programming Guide, SC34-6848
CICSPlex SM Application Programming Reference, SC34-6849

Diagnosis

CICSPlex SM Resource Tables Reference, SC34-6850
CICSPlex SM Messages and Codes, GC34-6851
CICSPlex SM Problem Determination, GC34-6852

CICS family books

Communication

CICS Family: Interproduct Communication, SC34-6853
CICS Family: Communicating from CICS on zSeries, SC34-6854

Licensed publications

The following licensed publications are not included in the unlicensed version of the Information Center:

CICS Diagnosis Reference, GC34-6862
CICS Data Areas, GC34-6863-00
CICS Supplementary Data Areas, GC34-6864-00
CICS Debugging Tools Interfaces Reference, GC34-6865

Other CICS books

The following publications contain further information about CICS, but are not provided as part of CICS Transaction Server for z/OS, Version 3 Release 2.

Designing and Programming CICS Applications	SR23-9692
CICS Application Migration Aid Guide	SC33-0768
CICS Family: API Structure	SC33-1007
CICS Family: Client/Server Programming	SC33-1435
CICS Transaction Gateway for z/OS Administration	SC34-5528
CICS Family: General Information	GC33-0155
CICS 4.1 Sample Applications Guide	SC33-1173
CICS/ESA 3.3 XRF Guide	SC33-0661

Books from related libraries

z/OS Security Server

z/OS Security Server RACF Auditor's Guide	SA22-7684
z/OS Security Server RACF Command Language Reference	SA22-7687
z/OS Security Server RACROUTE Macro Reference	SA22-7692
z/OS Security Server RACF Data Areas	GA22-7680
z/OS Security Server RACF Diagnosis Guide	GA22-7689
z/OS Security Server RACF General User's Guide	SA22-7685
z/OS Security Server RACF Macros and Interfaces	SA22-7682
z/OS Security Server RACF Messages and Codes	SA22-7686
z/OS Security Server RACF Security Administrator's Guide	SA22-7683
z/OS Security Server Network Authentication Service Administration	SC24-5926
z/OS Security Server RACF System Programmer's Guide	SA22-7681
z/OS Security Server RACF Security Administrator's Guide	SA22-7683

Systems Network Architecture (SNA)

SNA Reference: Peer Protocols	SC31-6808

z/OS

z/OS MVS Planning: Operations	SA22-7601
z/OS MVS System Codes	SA22-7626
z/OS MVS Initialization and Tuning Guide	SA22-7591

z/OS System Secure Sockets Layer

z/OS System Secure Sockets Layer Programming	SC24-5901

Determining if a publication is current

IBM regularly updates its publications with new and changed information. When first published, both hardcopy and BookManager® softcopy versions of a publication are usually in step. However, due to the time required to print and distribute hardcopy books, the BookManager version is more likely to have had last-minute changes made to it before publication.

Subsequent updates will probably be available in softcopy before they are available in hardcopy. This means that at any time from the availability of a release, softcopy versions should be regarded as the most up-to-date.

For CICS Transaction Server books, these softcopy updates appear regularly on the *Transaction Processing and Data Collection Kit* CD-ROM, SK2T-0730-xx. Each reissue of the collection kit is indicated by an updated order number suffix (the -xx part). For example, collection kit SK2T-0730-06 is more up-to-date than SK2T-0730-05. The collection kit is also clearly dated on the cover.

Updates to the softcopy are clearly marked by revision codes (usually a # character) to the left of the changes.

Accessibility

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

You can perform most tasks required to set up, run, and maintain your CICS system in one of these ways:
- using a 3270 emulator logged on to CICS
- using a 3270 emulator logged on to TSO
- using a 3270 emulator as an MVS system console

IBM Personal Communications provides 3270 emulation with accessibility features for people with disabilities. You can use this product to provide the accessibility features you need in your CICS system.

Index

Notices

Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. A current list of IBM trademarks is available on the Web at Copyright and trademark information at www.ibm.com/legal/copytrade.shtml.

Adobe, the Adobe logo, PostScript, and the PostScript logo are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, and/or other countries.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

Microsoft, Windows, and Windows NT are trademarks of Microsoft Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Other product and service names might be trademarks of IBM or other companies.

Readers' Comments — We'd Like to Hear from You

CICS Transaction Server for z/OS
RACF Security Guide
Version 3 Release 2

Publication No. SC34-6835-03

We appreciate your comments about this publication. Please comment on specific errors or omissions, accuracy, organization, subject matter, or completeness of this book. The comments you send should pertain to only the information in this manual or product and the way in which the information is presented.

For technical questions and information about products and prices, please contact your IBM branch office, your IBM business partner, or your 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. IBM or any other organizations will only use the personal information that you supply to contact you about the issues that you state on this form.

Comments:

Thank you for your support.

Submit your comments using one of these channels:
- Send your comments to the address on the reverse side of this form.
- Send a fax to the following number: +44–1962–816151
- Send your comments via email to: idrcf@hursley.ibm.com

If you would like a response from IBM, please fill in the following information:

Name	Address
Company or Organization	
Phone No.	Email address

Readers’ Comments — We'd Like to Hear from You
SC34-6835-03

Cut or Fold
Along Line

Fold and Tape **Please do not staple** Fold and Tape

PLACE
POSTAGE
STAMP
HERE

IBM United Kingdom Limited
User Technologies Department (MP095)
Hursley Park
Winchester
Hampshire
SO21 2JN
United Kingdom

Fold and Tape **Please do not staple** Fold and Tape

SC34-6835-03

Cut or Fold
Along Line

Product Number: 5655-M15

SC34-6835-03

Spine information:

IBM CICS Transaction Server for z/OS RACF Security Guide

Version 3
Release 2