CICS Transaction Server for z/OS
5.6

RACF Security Guide

Note

Before using this information and the product it supports, read the information in Product Legal Notices.

This edition applies to the IBM® CICS® Transaction Server for z/OS®, Version 5 Release 6 (product number 5655-Y305655-BTA) and to all subsequent releases and modifications until otherwise indicated in new editions.

Contents

About this PDF xi

Chapter 1. Introduction to CICS security with RACF 1
- Security facilities in CICS 1
  - What CICS security protects 1
  - CICS users 2
  - The CICS region user ID 3
  - The CICS default user ID 4
  - Terminal user security 5
  - Non-terminal security 5
  - The QUERY SECURITY command 5
  - Intercommunication security 6
  - PassTickets for sign-on security 6
  - API and SPI restrictions DFHAPIR parmlib member 7
- RACF facilities 8
  - RACF administration 8
  - Delegation of RACF administrative responsibility 9
  - RACF profiles 10
  - RACF user profiles 11
  - RACF group profiles 16
  - RACF data set profiles 17
  - RACF general resource profiles 18
  - RACF classes for CICS resources 19
  - RACF classes for protecting system resources 20
  - Resource classes for DB2ENTRY resources 23
  - Summary of RACF commands 23
  - Security classification of data and users 26
  - Defining your own resource classes 26
  - Controlling access to fields in RACF profiles 27

Chapter 2. Implementing RACF protection in a single CICS region 29
- CICS system resource security 29
  - CICS installation requirements for RACF 29
  - Specifying the CICS region userid 30
  - Authorizing CICS procedures to run under RACF 31
  - Defining user profiles for CICS region user IDs 32
  - Defining the default CICS user ID to RACF 34
  - Authorizing access to MVS log streams 35
  - Authorizing access to CICS data sets 36
  - Authorizing access to the temporary storage pools 39
  - Authorizing access to temporary storage servers 40
  - Authorizing access to named counter pools and servers 41
  - Authorizing access to SMSVSAM servers 42
  - Authorizing access to the CICS region 43
  - Controlling the opening of a CICS region's z/OS Communications Server ACB 44
  - Controlling userid propagation 44
  - Surrogate job submission in a CICS environment 45
  - Authorizing the CICS region userid as a surrogate user 45
  - JES spool protection in a CICS environment 46
  - Authorizing use of **HPO** in **PARM** parameter on an EXEC PGM=DFHSIP statement or in SYSIN 46

Security-related system initialization parameters... 46
Verifying CICS users... 53
Identifying CICS terminal users... 53
Setting up PassTickets for secure sign-on... 53
Using PassTickets in FEPI applications... 54
The sign-on process... 55
The sign-off process... 56
Controlling access to CICS from specific ports of entry... 57
Defining port of entry profiles... 58
Auditing sign-on and sign-off activity... 61
Preset terminal security... 61
Using an MVS system console as a CICS terminal... 64
Obtaining CICS-related data for a user... 65
Support for mixed-case passwords... 67
National language and non-terminal transactions... 68
Transaction security... 69
CICS parameters controlling transaction-attach security... 69
Defining transaction profiles to RACF... 71
Authorization failures and error messages... 72
Protecting non-terminal transactions... 72
Resource security... 73
Security using the XRES resource security parameter... 73
Resource security checking by CICS and RACF... 76
Security for transient data... 80
Security for files... 82
Security for journals and log streams... 83
Security for started transactions... 84
Security for transactions started with EXEC CICS RUN TRANSID... 87
Security for XPCT-checked transactions... 87
Security for application programs... 88
Security for temporary storage... 89
Security for z/OS UNIX files... 90
Security for program specification blocks... 93
Security checking of transactions running under CEDF, CEDG, CEDX, or CEDY... 94
Defining generic profiles for resources... 94
Defining generic profiles for resources... 95
Resource and command check cross-reference... 96
Surrogate user security... 113
Situations where surrogate user checking applies... 113
RACF definitions for surrogate user checking... 117
Examples of RACF definitions for surrogate user checking... 117
CICS command security... 118
Introduction to command security... 118
CICS resources subject to command security checking... 120
Parameters for specifying command security... 129
Security checking of transactions running under CEDF... 130
CEMT considerations... 131
Authorization failures in application programs... 131
Security checking using the **QUERY SECURITY** command... 131
How **QUERY SECURITY** works... 131
The RESTYPE option... 133
The RESCLASS option... 138
Specifying user-defined resources to RACF... 139
Querying a user's surrogate authority... 141
Logging for QUERY SECURITY... 141
Examples: Using the **QUERY SECURITY** command for resource security checking... 141
Security for CICS transactions... 142
Security for submitting a JCL job to the internal reader... 148

When and how CICS determines the user ID of the CICS region....150

Chapter 3. Identity propagation and distributed security.... 153
Support and requirements for identity propagation....154
Sample network topologies for using identity propagation.... 155
Configuring identity propagation....157
Configuring RACF for identity propagation....157

Chapter 4. Invoking an external security manager.... 159
Overview of the MVS router....159
How ESM exit programs access CICS-related information.... 159
The RACF user exit parameter list.... 159
The installation data parameter list....160

Chapter 5. Security for CICSPlex SM.... 163
Implementing CICSPlex SM security.... 163
Determining who requires access to CICSPlex SM resources.... 163
General requirements for CICSPlex SM security.... 167
Creating profiles for the CICSPlex SM data sets.... 167
Determining the agent user ID for CICSPlex SM components....168
Defining the CICSPlex SM started tasks....170
Defining the CICSPlex SM transactions in a CMAS....170
Defining the transactions in a managed CICS region....171
Setting up CICSPlex SM Web User Interface security.... 172
Specifying CICSPlex SM resource names in profiles....176
Security for platforms and applications.... 202
Activating simulated CICS security.... 205
Considerations for CICS surrogate security checks....206
Activating security for CICSPlex SM.... 207
Refreshing RACF profiles for CICSPlex SM....207
CICSPlex SM security checking sequence.... 208
Invoking a user-supplied external security manager.... 212
An overview of the CICSPlex SM ESM interface....212
Overview of the MVS router.... 212
CICSPlex SM security control points.... 212
Example tasks: security....213
Example: Protecting all CICSPlex SM resources....213
Example: Giving CICSPlex SM operators appropriate authorizations.... 214
Example: Giving a user read access to all transactions on MVS system A....215

Chapter 6. Security for platforms and applications.... 217

Chapter 7. Security for intercommunication....221
Overview of intercommunication security.... 221
Introduction to intercommunication security.... 221
Planning for intercommunication security.... 221
Intercommunication bind-time security.... 221
Intercommunication link security....222
User security for intercommunication....222
Transaction, resource, command, and surrogate user security for intercommunication....223
Summary of intercommunication security levels....223
Implementing LU6.2 security....224
Bind-time security with LU6.2.... 224
Link security with LU6.2....228
User security with LU6.2....229
SNA profiles and attach-time security....233
Transaction, resource, and command security with LU6.2....234

Transaction routing security with LU6.2....235
Function shipping security with LU6.2....236
Distributed program link security with LU6.2....237
Security checking done in AOR with LU6.2....238
Implementing LU6.1 security....239
Link security with LU6.1....240
Specifying link security for LU6.1 connections....240
Specifying ATTACHSEC with LU6.1....240
Transaction, resource, and command security with LU6.1....241
Function shipping security with LU6.1....241
Security checking done in AOR with LU6.1....242
APPC password expiration management....243
What you require to use APPC PEM....244
External security interface....244
Roles of PEM client and CICS PEM server....245
Overview of APPC PEM processing....247
Setting up the PEM client....252
PEM client input and output data....253
Implementing IPIC security....261
IPIC bind-time security....261
IPIC link security....263
IPIC user security....264
IPIC transaction, resource, and command security....266
CICS routing transaction, CRTE....266
Security checking done in AOR with IPIC....267
Implementing MRO security....268
Security implications of choice of MRO access method....268
Bind-time security with MRO....268
Logon security checking with MRO....269
Link security with MRO....271
Specifying link security for MRO connections....272
User security with MRO....272
Transaction, resource, and command security with MRO....274
Transaction routing security with MRO....275
Function shipping security with MRO....276
Distributed program link security with MRO....277
Security checking done in AOR with MRO....277
Security for data tables....278
Security for CICS shared data tables....278
Security for coupling facility data tables....281

Chapter 8. Security for TCP/IP clients....285
About security for TCP/IP clients....285
Message protection....285
Identification and authentication....288
Support for security protocols....292
Configuring CICS to use SSL....297
Setting up profiles in RACF....298
Requesting a certificate from a certificate authority....299
Building a key ring manually....300
Building a key ring with certificates using DFH$RING....301
Creating new RACF certificates....302
Associating a RACF user ID with a certificate....303
Using an existing certificate that is not owned by the CICS region user ID....304
Configuring a RACF site certificate for use with CICS TS....304
Making a certificate untrusted....305
System initialization parameters for SSL....305

TCPIPSERVICE attributes for SSL....306
Creating an SSL cipher suite specification file....307
Customizing encryption negotiations....308
Making your CICS TS system conformant to NIST SP800-131A....309
Configuring LDAP for CICS use....311
Using certificate revocation lists (CRLs)....312

Chapter 9. Security for data sources....315
Security for Db2....315
Controlling access to Db2-related resources in CICS....316
Providing authorization IDs to Db2 for the CICS region and for CICS transactions....323
Authorizing users to access resources in Db2....330
Db2 multilevel security and row-level security....333
Security for DBCTL....333
PSB authorization checking by CICS....334
Security for data sets: encryption....334

Chapter 10. Security for external interfaces....337
Security for EXCI....337
Using MRO logon and bind-time security....337
Link security....338
User security....338
Surrogate user checking....339
Security for ONC RPC....339
Security in ONC RPC....339
Security in CICS and its effect on CICS ONC RPC operations....340
Writing the resource checker....342

Chapter 11. Security for Java applications....345
Configuring security for OSGi applications....345
Configuring security for a Liberty JVM server....345
The Liberty angel process....348
Authenticating users in a Liberty JVM server....352
Authorizing users to run applications in a Liberty JVM server....354
Authorizing applications by using OAuth 2.0....355
Authorization using SAF role mapping....358
Configuring security for a Liberty JVM server with the Enterprise Java security API....359
Configuring security for a Liberty JVM server by using an LDAP registry....364
Configuring security for remote JCICSX API development....367
Configuring SSL (TLS) for a Liberty JVM server using a Java keystore....372
Configuring SSL (TLS) for a Liberty JVM server using RACF....373
Configuring SSL (TLS) for remote JCICSX API development....374
Setting up SSL (TLS) client certificate authentication in a Liberty JVM server....377
Using the syncToOSThread function....378
Enabling a Java security manager....379

Chapter 12. Security for Node.js applications....381

Chapter 13. Security for CICS web support....383
CICS as an HTTP server: authentication and identification....383
CICS as an HTTP client: authentication and identification....384
Password expiry management for HTTP basic authentication....385
CICS system and resource security for CICS web support....387
Security for inbound ports....387
Security for CICS system components....388
Resource and transaction security for application-generated responses....388
Resource-level security for static responses using document templates....391

SSL with CICS web support ... 392
Introduction to Application Transparent Transport Layer Security (AT-TLS) ... 392
Security for Atom feeds ... 403

Chapter 14. Support for securing web services ... 405
Enabling CICS for WS-Security processing ... 405
Planning to secure SOAP web services ... 406
Options for securing SOAP messages ... 407
Authentication using a Security Token Service ... 408
The Trust client interface ... 409
Signing of SOAP messages ... 410
Signature algorithms ... 410
Example of a signed SOAP message ... 410
CICS support for encrypted SOAP messages ... 411
Encryption algorithms ... 412
Example of an encrypted SOAP message ... 412
Configuring RACF for Web Services Security ... 413
Configuring provider mode web services for identity propagation ... 415
Configuring the pipeline for Web Services Security ... 417
Writing a custom security handler ... 420
Invoking the Trust client from a message handler ... 420
Security for z/OS Connect ... 421
Configuring permissions for Services and APIs ... 422

Chapter 15. Security in BTS ... 425
Resource security in BTS ... 425
Process and activity user IDs ... 425
Attach-time security for processes and activities ... 426
Command security in BTS ... 426

Chapter 16. Security for the CICS-MQ adapter ... 427
Implementing security for CICS-MQ adapter transactions ... 427
CICS-MQ adapter user IDs ... 428
Command security for MQCONN and MQMONITOR resources ... 428
Surrogate user security for MQMONITOR resources ... 429
IBM MQ connection security for the CICS-MQ adapter ... 429

Chapter 17. Security for the CICS-MQ bridge ... 431

Chapter 18. Kerberos support ... 435
Configuring RACF for Kerberos ... 435
Configuring CICS web services for Kerberos ... 436
Developing Kerberos applications ... 437
Using a Kerberos security token in a 3270 emulator sign-on ... 437

Chapter 19. Support for JWT using RACF ... 439

Chapter 20. Support for Multi-Factor Authentication using RACF ... 441

Chapter 21. Configuring CICS for SAML ... 443
Validating your configuration ... 443
Configuring a provider pipeline ... 444
Configuring a requester pipeline ... 446
Configuring the CICS STS ... 447
The STS configuration file ... 448
Patterns for developing SAML-aware programs ... 449

Chapter 22. Developing SAML applications..... 451
Developing a SAML-aware initial program..... 451
Pattern for developing a SAML-aware initial program..... 451
Developing a program that uses a validated SAML token in an outbound request..... 452
Developing a program that creates or augments SAML tokens..... 452

Notices..... 455

Index..... 461

About this PDF

This PDF describes how to plan and implement security across your CICS systems. It 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. This PDF will also be of interest for CICS system programmers who need to communicate their requirements to the security administrator for their installation.

This PDF replaces *RACF Security Guide*, SC34-7423-00.

For details of the terms and notation used, see Conventions and terminology used in CICS documentation in IBM Documentation.

Date of this PDF

This PDF was created on 2022-12-08 (Year-Month-Date).

Chapter 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.

Security facilities in CICS

As an online transaction-processing system (often supporting many thousands of users), CICS requires the protection of a security system to ensure that the resources to which it manages access are protected and are secure from unauthorized access. CICS provides a number of facilities that protect your resources against unauthorized access.

To provide the required 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.

For specific security information about Business Transaction Services, see Security in BTS.

What CICS security protects

CICS manages 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 that 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

CICS itself does **not** provide facilities to protect its own assets from external access. You must 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 must 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
- 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.
- 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 About security for TCP/IP clients.
- 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 transactions.
- 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.
- 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.
- 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.
- 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 Overview of BTS.
- 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.

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

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.

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 35.

CICS system data sets
For more information, see “Authorizing access to CICS data sets” on page 36.

CICS user data sets
For more information, see “Authorizing access to user data sets” on page 39.

Temporary storage data sharing servers
For more information, see “Authorizing access to temporary storage servers” on page 40.

The SMSVSAM server
For more information, see “Authorizing access to SMSVSAM servers” on page 42.

Named counter servers
For more information, see “Authorizing access to named counter pools and servers” on page 41.

The z/OS Communications Server ACB
For more information, see “Controlling the opening of a CICS region's z/OS Communications Server ACB” on page 44.

JES spool data sets
For more information see “JES spool protection in a CICS environment” on page 46.

The CICS interregion program
For more information, see “Logon security checking with MRO” on page 269.

Coupling facility data tables
For more information, see “Security for coupling facility data tables” on page 281.

RACF key rings
For more information, see “Building a key ring manually” on page 300.

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 44.

As a surrogate for other user IDs used during CICS execution
For more information, see Surrogate user security.

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 46.

When executing CICS-supplied non-terminal transactions
For more information, see Security for CICS-supplied transactions.

In CICS intersystem communication using LU6.2
For more information, see "Implementing LU6.2 security" on page 224.

In CICS intersystem communication using LU6.1
For more information, see "Implementing LU6.1 security" on page 239.

In CICS intersystem communication using MRO
For more information, see "Implementing MRO security" on page 268.

In Liberty JVM server using the angel process
For more information, see The Liberty server angel process.

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 30.

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 "Verifying CICS users" on page 53.
- 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 115.
- 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 240 and "Link security with LU6.2" on page 228. The default user ID can also be used as the link user ID for IPIC. For more information, see Security checking done in AOR with IPIC.
- 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 233 and "User security with MRO" on page 272. The default user ID can also be assigned to transactions attached by IPIC sessions. For more information see IPIC user security.
- 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 CICS routing transaction, CRTE.
- 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 278.
- 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 288.
- It is assigned to the USERID attribute of an MQMONITOR if the USERID has not been explicitly specified (for example, the dynamically installed MQMONITOR DFHMQINI).
- The initial user ID used by default by z/OS Connect, Liberty, and WEBSERVICEs.
- The user ID used by default for unauthenticated z/OS Connect, Liberty, and WEBSERVICE requests.

For more information about how you can configure security for a Liberty JVM server, see Configuring security for a Liberty JVM server. For more information about how you can configure permissions for z/OS Connect, see Configuring permissions for z/OS Connect Services and APIs.

Terminal user security

To secure resources from unauthorized access, CICS requires 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 sign-on 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. You can use the **GMTRAN** system initialization parameter to control what happens if the user fails to complete the sign-on. For example, all subsequent transactions use the CICS default user ID, or the terminal session is disconnected.

See “Terminal profiles” on page 58 for information about the terminal security facilities provided by RACF. See “Verifying CICS users” on page 53 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.

Non-terminal security

You can specify security for transactions that are not associated with terminals.

Transactions that are not associated with terminals include:

- 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 72.

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.

For more information:

- See QUERY SECURITY.
- See "RACF general resource profiles" on page 18 for information about the resource classes that RACF supports for resource security checking within transactions.
- For more information about resource security checking, see "Resource security" on page 73.

Intercommunication security

You can connect a number of CICS regions together by using intercommunication; for example, intersystem communication over SNA (ISC over SNA) which uses an SNA access method, such as ACF/VTAM, to provide the required communication protocols. The basic security principles apply to interconnected systems, but the resource definition is more complex and there are additional security requirements.

APPC (LU6.2) session security

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. 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 "Implementing LU6.2 security" on page 224 for more information.

Multiregion operation (MRO)

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 "Implementing MRO security" on page 268 for information about MRO security.

IP interconnectivity (IPIC) security

The security mechanisms for IPIC connections are similar to those provided for APPC (LU6.2) connections, although 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.

For information about IPIC connections, see Communication between systems.

PassTickets for sign-on security

A PassTicket is a secure representation of a password that your program can use to sign on to a particular application on a particular system, such as another CICS region. A specific PassTicket may be used for authentication once only, and it must be used within 10 minutes of being generated. The PassTicket can be used anywhere a password can be used.

Why use PassTickets?

Using a PassTicket in place of a password means that applications do not have to store passwords (or ask users to re-enter them) in order to sign on to the destination system, and passwords are not transmitted across the network.

How does it work?

The client on the originating system needs to generate a PassTicket for the destination system by using the RACF PassTicket generator algorithm.

If the originating system is CICS, to create a PassTicket for the signed-on user, your application issues the **EXEC CICS REQUEST PASSTICKET** or **FEPI REQUEST PASSTICKET** command to request RACF, or a functionally-equivalent external security manager that supports PassTickets, to generate a PassTicket.

The destination system authenticates the user ID and PassTicket with an external security manager. Existing commands and procedures can be used for authentication.

When CICS authenticates the user ID and PassTicket, it calls an external security manager such as RACF to check whether the PassTicket supplied is for the specified user ID for that region. An optional check might be performed to verify that the specified user ID is connected to the specified group ID.

Find out more

Using the secured signon function in z/OS Security Server RACF Security Administrator's Guide gives you more information about PassTickets.

Incorporating the PassTicket generator algorithm into your program in z/OS Security Server RACF Macros and Interfaces gives you more details on the algorithm used for the RACF PassTicket generator.

"Setting up PassTickets for secure sign-on" on page 53 gives you instructions on how to implement PassTickets in your CICS environment.

API and SPI restrictions DFHAPIR parmlib member

The DFHAPIR parmlib member contains rules that identify restricted CICS API and SPI commands. The CICS translator uses this parmlib member to check program source against the specified command rules during translation. By default, the command rules apply to every user on the LPAR where DFHAPIR is located. However, you can exempt some users or LPARs by using profiles in RACF.

When you need to define RACF profiles for the DFHAPIR parmlib member

A RACF profile DFHAPIR.lpar in the FACILITY class defines whether users are subject to the command rules file. If users have no access to the profile, or if a profile is not defined, they are subject to the command rules file. If a user has READ access to the profile, the user is not subject to the command rules file.

The following examples will show the two common scenarios:

- The command rules apply to most users but some users are exempt.
- The command rules apply to some LPARs, but not all.

Examples: Defining RACF profiles for the DFHAPIR parmlib member

The RACF FACILITY class is used to protect the DFHAPIR parmlib member. The following examples show how you can define RACF profiles and access lists for the two common scenarios.

Example 1: Defining a general profile that applies to all LPARs

This example creates a profile that applies to all LPARs. UACC(NONE) means that the command rules apply, by default, to all users.

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

To allow some users to bypass the command rules, give them READ access to the profile by using PERMIT commands to set up an access list.

```
PERMIT DFHAPIR.** CLASS(FACILITY) ID(user) ACCESS(READ)
```

Example 2: Defining profiles that apply to particular LPARs

To enforce command rules only on particular LPARs, set up DFHAPIR.*lpar* profiles where *lpar* is the four-character LPAR name. These should be set up for those LPARs where users are subject to the command rules file:

```
RDEFINE FACILITY DFHAPIR.** UACC(READ)
RDEFINE FACILITY DFHAPIR.lpar UACC(NONE)
```

As for the more general profile, UACC(NONE) means that the command rules apply to all users on the LPAR. Use an access list to grant exempt users READ access to the profile if required.

```
PERMIT DFHAPIR.lpar CLASS(FACILITY) ID(user) ACCESS(READ)
```

RACF facilities

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

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 more information on logging RACF audit messages to SMF, see “Logging RACF audit messages to SMF” on page 80.

  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)
  ```

  This 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 18.
- **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 23.

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" on page 11.
- The CICS segment, which holds data for each CICS user. See "The CICS segment" on page 11.
- The LANGUAGE segment, which specifies the user's national language preference. See "The LANGUAGE segment" on page 15.

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 65.

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 Message routing.

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.

For more information about message routing, see Message routing.

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 **RSTSIGNOFF** 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.

Authorizing access to temporary storage servers 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. 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 **RSTSIGNTIME** 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 65.

If you specify some of the CICS segment options, but omit others, the defaults described in this document 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 29 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 27.

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))
```

Changing the RACF profile of a remote user

CICS is notified of certain changes in the RACF profile of a signed-on remote user, or a signed-on user who is not directly using a physical terminal or console, through a type 71 RACF Event Notification (ENF).

RACF sends type 71 ENFs when RACF commands such as **ALTUSER** with the REVOKE option, **CONNECT**, **REMOVE**, **DELGROUP** and **DELUSER** affect the group authorization of a user. In addition, with RACF APAR OA58677 and SAF APAR OA58678, RACF sends a type 71 ENF when a user ID is revoked automatically as a result of too many failed password attempts.

CICS monitors for such RACF type 71 ENFs. Notification of a change to the user ID overrides any setting that is specified in the **USRDELAY** system initialization parameter. Therefore, review your **USRDELAY** settings.

For example, CICS is notified when you use the REVOKE option on the **ALTUSER** command, with no date specified, to revoke a user ID with immediate effect. However, CICS is not notified when a user ID expires, or for a user ID that is signed on to a local region (for example, a TOR that uses the CESN transaction to sign on).

When a RACF profile change occurs and CICS receives a new attach request for a user ID, CICS performs an implicit sign-on for the user ID and the new RACF profile information is used. Existing tasks for that user continue with the RACF profile that was valid when the task was attached.

If you specified a low value for the **USRDELAY** system initialization parameter to ensure that CICS detects changes to RACF profiles promptly, you might want to increase this value, because CICS is now notified immediately if RACF profile changes occur. The primary impact of a high **USRDELAY** value is that the amount of storage used for RACF control blocks is increased.

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 one of the following situations occurs:

- The transaction performs a syncpoint.
- The attach request ends.
- Sign off occurs because RACF notifies CICS of changes to a user profile, and an attached request associated with that signed-on user ID completes, for all operands of ATTACHSEC except LOCAL.
- Sign off occurs because RACF notifies CICS of changes to a user profile, a new attach request is made, and the value in the **USRDELAY** system initialization parameter has not expired. This sign off is followed by a sign on.

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 Surrogate user security).

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 CICS-issued messages.

You can specify two languages, but CICS assigns each user only one language. CICS assigns the primary language if the language is specified and corresponds to the single-character code specified on the **NATLANG** system initialization parameter. Otherwise, CICS assigns the secondary language by the same standard. 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 66.

To specify a preferred national languages in the LANGUAGE segment of the RACF user profile, ensure that the language is defined in the CICS system and then use the **LANGUAGE** parameter on the ADDUSER or ALTUSER command. The following example shows a language request in the ALTUSER command:

```
ALTUSER userid LANGUAGE(PRIMARY(language_code) SECONDARY(language_code))
```

The information you can specify is as follows:

LANGUAGE
: Specifies 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(*language_code*)
Specifies 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 National language codes.

SECONDARY(*language_code*)
Specifies the user's secondary 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 National language codes.

Notes:

1. CICS messages are supported only in UK English, Simplified Chinese, and Japanese. If any other language other than those three is specified, English is used by default.
2. CICS ignores the RACF default national language defined by the SETROPTS command:

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

For more information about national language, see "National language and non-terminal transactions" on page 68.

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. By adding a user to a group, the user has access to all the resources to which the group has access. Users can be connected to one or more groups.

A group profile can contain information about the group, for example, who owns it, which subgroups it has, and a list of connected users. For details of how to define and use group profiles, see the z/OS Security Server RACF Security Administrator's Guide. A group profile is different from a resource group profile, which defines a group of resources and is explained in "RACF general resource profiles" on page 18.

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 user ID in a group, not the access list of the resource profile. When someone leaves a department, removing the user ID from the department user group revokes all privileges. No other administration of profiles is required. Thus, you keep RACF administration to a minimum and avoid 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 18.

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. In this way, you do not have to change the access list of the affected profiles (through the use of the PERMIT command). If you do not change a CICS general resource profile, you do not have to refresh its in-storage copy. However, users might have 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.

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)
```

CICS is notified of certain changes in the RACF profile of a signed-on remote user, or a signed-on user who is not directly using a physical terminal or console, through a type 71 RACF Event Notification (ENF).

RACF data set profiles

Using RACF facilities, you can protect data sets on direct access storage devices (DASD) and tape 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 16 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 CICSTS56.CICS, you can define a generic profile named 'CICSTS56.CICS.**', and any user in the access list of this profile can access, at the authorized level, data sets with the high-level qualifier CICSTS56.CICS. For example:

```
ADDSD 'CICSTS56.CICS.**' UACC(NONE) NOTIFY(admin_userid)
```

Use the SETROPTS GENERIC command before defining generic profiles, as described in “Summary of RACF commands” on page 23.

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

RACF supplies a number of resource classes that CICS uses for its resources.

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*.

They are described in "RACF classes for CICS resources" on page 19. 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 20.

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" on page 18. 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 z/OS Security Server RACF Macros and Interfaces.

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. 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.

Notes:

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 member class and the resource grouping class respectively.
2. There are no default resource class names for DB2ENTRY resources. You must define your own resource classes for these resources. See Resource classes for DB2ENTRY resources for more information.
3. Profile names in the RCICSRES class must contain a prefix that specifies the CICS resource type to which they apply: ATOMSERVICE.*name* for ATOMSERVICE definitions, BUNDLE.*name* for bundles, DOCTEMPLATE.*name* for CICS document templates, EPADAPTER.*name* for EP adapters, EPADAPTERSET.*epadapterset_resource_name* for EP adapter sets, EVENTBINDING.*name* for event bindings, or XMLTRANSFORM.*name* for XML transforms.

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
MCICSPPT	NCICSPPT	CICS programs
SCICSTST	UCICSTST	CICS temporary storage queues
CCICSCMD	VCICSCMD	EXEC CICS SYSTEM commands and EXEC CICS FEPI system commands
RCICSRES	WCICSRES	Document templates, bundles, EP adapters, EP adapter sets, event bindings, ATOMSERVICE definitions, and XML transforms

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 “Implementing CICSPlex SM security” on page 163.

CPSMXMP
Controls exemption from simulated CICS security checking in CICSPlex SM. For more information, see “Exempting users and resources from security checking” on page 206

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 Resolving conflicts among grouping profiles in z/OS Security Server RACF Security Administrator's Guide.

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 several general resource classes.

APPCLU
Verifies the identity of APPC partner logical units (LU type 6.2) during z/OS Communications Server session establishment. For more information, see “Defining profiles in the APPCLU general resource class” on page 225.

APPL
Controls terminal users' access to z/OS Communications Server applications, including CICS. For more information, see “Authorizing access to the CICS region” on page 43.

CONSOLE
Controls user access to consoles. For more information, see “Console profiles” on page 60.

DIGTCERT
Contains digital certificates, and related information. For more information, see “Creating new RACF certificates” on page 302.

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 21.

FIELD
Controls access to fields in RACF profiles. For more information, see “Controlling access to fields in RACF profiles” on page 27.

IDIDMAP
The IDIDMAP resource profile contains the distributed identity filter. RACF uses the term *distributed identity filter* to describe a mapping association between a RACF user ID and one or more distributed identities. For more information, see Configuring RACF for identity propagation.

JESSPOOL
Protects JES spool data sets. For more information, see "JES spool protection in a CICS environment" on page 46.

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 35.

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 64.
- 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 30.

PROPCNTL
Prevents the CICS region user ID 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 44.

PTKTDATA
Contains the encryption keys used for generating and validating PassTickets. For more information, see "PassTickets for sign-on security" on page 6.

STARTED
Contains profiles that provide the user IDs for MVS started jobs. For more information, see "Using STARTED profiles for started jobs" on page 31.

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 42.

SURROGAT
Specifies which user IDs can act as surrogates for other user IDs. For more information, see Surrogate user security.

TERMINAL
Controls the ability of users to sign on at individual terminals. The corresponding resource group class is GTERMINL. For more information, see Terminal profiles.

VTAMAPPL
Controls the ability of users to open an SNA ACB. For more information, see "Controlling the opening of a CICS region's z/OS Communications Server ACB" on page 44.

Note: VTAM® is now z/OS Communications Server (for SNA or IP)

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 38.

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 "Implementing MRO security" on page 268.

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 278.

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

For more information, see "Security for coupling facility data tables" on page 281.

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 41.

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 39.

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 35.

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 Controlling access to DB2-related resources in CICS.

CICSPlex SM resources
The FACILITY class controls access to many CICSPlex SM resources. For more information, see "Implementing CICSPlex SM security" on page 163.

API and SPI restrictions DFHAPIR parmlib member
The FACILITY class controls the ability of users to access the DFHAPIR parmlib member, which contains rules that identify restricted CICS API and SPI commands. For more information, see "API and SPI restrictions DFHAPIR parmlib member" on page 7.

HPO system initialization parameter overrides
The FACILITY class controls the ability to specify **HPO** in the **PARM** parameter on an EXEC PGM=DFHSIP statement or in the SYSIN data set.

Resource classes for DB2ENTRY resources

CICS supports resource security checking for CICS-defined DB2ENTRY resources, for which there are no IBM-supplied RACF resource classes.

For DB2ENTRY resources, 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 define DB2ENTRY profiles in any of the CICS default resource classes. 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 18.

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?.

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 46.)

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 23.

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.

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 Customizing security processing.

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 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 139.

When setting up installation defined classes, we recommend that you copy the IBM-supplied defaults from the CDT, an example of which is in z/OS Security Server RACF Macros and Interfaces. You will then need to change the name, group or member name, POSIT number, and ID. See the description of the ICHERCDE macro in z/OS Security Server RACF Macros and Interfaces 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 CICSTS56.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 active 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 23.

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.

Chapter 2. Implementing RACF protection in a single CICS region

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

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.CICSTS56.CICS.SDFHLINK, which should be made an APF-authorized library in your MVS linklist.

For more information, see Installing CICS-required modules in the MVS linklist.

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, 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 Installing CICS-required modules in the MVS linklist.

Protecting CICS load libraries

Although, in general, CICS runs in unauthorized state, the CICS initialization program, DFHSIP, must 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 CICSTS56.CICS.SDFHAUTH. This library must be defined to the operating system as APF-authorized.

To prevent unauthorized or accidental modification of CICSTS56.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 startup 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 startup 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.CICSTS56.CICS.SDFHLINK and to SYS1.CICSTS56.CICS.SDFHLPA; and the other libraries (including CICSTS56.CICS.SDFHLOAD) that make up the STEPLIB and DFHRPL library concatenations.

See “Authorizing access to CICS data sets” on page 36 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.) 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, Link security with LU6.1, or Link security with MRO, 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 user ID 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, NOPHRASE and NOIDCARD 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, you should 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 this example 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 this example the CICS region userid is defined as CICSDAA2, the same as the APPLID.

Defining user profiles for CICS region user IDs

Before bringing up a CICS region, ensure that the required user IDs are defined - the CICS region user ID and the CICS default user ID.

If you are suitably authorized, you can define a RACF user profile for a CICS region by using the **ADDUSER** command. For example, to define CICSR as a user ID for a CICS region, enter the following RACF command from TSO:

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

Do not assign the OPERATIONS attribute to CICS region user IDs. Doing so would allow the CICS region to access RACF-protected data sets for which no specific authorization has been performed. CICS region user IDs do not need the **OPERATIONS** attribute if the appropriate **CONNECT** or **PERMIT** commands have been issued. These commands authorize the CICS region user ID 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 45 for more information. The region userid must also have surrogate authority to use the default user; see Surrogate user security.

Authorities required for CICS region user IDs

The CICS control program runs under the CICS region user ID.

Therefore, this user ID needs access to all the resources that CICS itself must use. There are two types of these resources:

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

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 35.
- 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 36.
- 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 39.
- 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 40.
- 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 42.
- z/OS Communications Server (for SNA or IP) applications

  Consider carefully for each program whether you will allow it to become a z/OS Communications Server application. If you do this, CICS needs authority to open its z/OS Communications Server ACB. See "Controlling the opening of a CICS region's z/OS Communications Server ACB" on page 44.
- Jobs submitted to the internal reader

  If any of your applications submit JCL to the JES internal reader, you can allow them to be submitted without the USERID parameter. See "Controlling userid propagation" on page 44.

  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 45.
- 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 46.

Authorizing internal resources

There are several internal functions in which CICS behaves like an application program, but is performing housekeeping functions that are not directly for any 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 Security for CICS-supplied transactions.

- 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 113.

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

    See "Post-initialization processing" on page 114.

  – The userid used for transient data trigger transactions

    See "Transient data trigger-level transactions" on page 115.

- 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 73.

Defining the default CICS user ID to RACF

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

You can use the same default user ID on all CICS regions. You can specify this default user ID on the **DFLTUSER** system initialization parameter, or leave **DFLTUSER** set to the default of CICSUSER.

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

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

Step 1. Define the CICS default user to RACF

Use the **ADDUSER** command with the CICS operand to define a CICS default user to RACF.

Usually, the default CICS user ID should be defined as a protected user ID. This is particularly the case if the CICS region is a started task. Protected user IDs cannot be used to enter the system by any means that requires a password, and users cannot cause a protected user ID to be revoked. For more information, see "Using protected user IDs" on page 31.

Example:

The following command defines CICS default user CICSUSER as a protected user ID to RACF:

```
ADDUSER CICSUSER  DFLTGRP(group_id) NAME(user_name)
        OWNER(userid or group)
        NOOIDCARD
        NOPASSWORD
```

Step 2. Authorize the CICS region user ID to be a surrogate user of the default user ID

If you have specified the system initialization parameter **XUSER=YES** (the default), authorize the CICS region user ID to be a surrogate user of the default user ID. For example, the following command authorizes a CICS region user ID to be a surrogate user of CICSUSER:

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

Sign-on processing of the CICS default user

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

When CICS successfully signs on a valid RACF user ID 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 65 for details of the sign-on process for obtaining CICS terminal operator data.

How CICS assigns the security attributes of the default user

CICS assigns the security attributes of the default user ID 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.

CICS also assigns the security attributes of the default user ID to any trigger-level transactions that are initiated for transient data queues without a USERID parameter.

Ensure the default user ID 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 43.
- 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 53, 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 user id 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 user id, see “Specifying the CICS region userid” on page 30.

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 on page 36.

Table 3. Summary of generic data set profiles

Required access level	**Type of CICS data sets protected**
READ	Load libraries

Table 3. Summary of generic data set profiles (continued)	
Required access level	**Type of CICS data sets protected**
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 37 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 on page 37. 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  'CICSTS56.CICS.SDFHLOAD'  NOTIFY(cics_sys_admin_id) UACC(NONE)
PERMIT 'CICSTS56.CICS.SDFHLOAD' ID(cics_id1,...,cics_group1,..,cics_groupn)
                         ACCESS(READ)
ADDSD  'CICSTS56.CICS.SDFHAUTH'  NOTIFY(cics_sys_admin_id) UACC(NONE)
PERMIT 'CICSTS56.CICS.SDFHAUTH' ID(cics_id1,...,cics_group1,..,cics_groupn)
                         ACCESS(READ)
ADDSD  'CICSTS56.CICS.applid.**'  NOTIFY(cics_sys_admin_id) UACC(NONE)
PERMIT 'CICSTS56.CICS.applid.**'  ID(applid_userid) ACCESS(UPDATE)
ADDSD  'CICSTS56.CICS.applid.DFHXR*'  NOTIFY(cics_sys_admin_id) UACC(NONE)
PERMIT 'CICSTS56.CICS.applid.DFHXR*'  ID(applid_userid) ACCESS(CONTROL)
ADDSD  'CICSTS56.CICS.applid.DFHINTRA' NOTIFY(cics_sys_admin_id) UACC(NONE)
PERMIT 'CICSTS56.CICS.applid.DFHINTRA'  ID(applid_userid) ACCESS(CONTROL)
ADDSD  'CICSTS56.CICS.applid.DFHTEMP' NOTIFY(cics_sys_admin_id) UACC(NONE)
PERMIT 'CICSTS56.CICS.applid.DFHTEMP'  ID(applid_userid) ACCESS(CONTROL)
ADDSD  'CICSTS56.CICS.DFHCSD'          NOTIFY(cics_sys_admin_id) UACC(NONE)
PERMIT 'CICSTS56.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 'CICSTS56.CICS.**'. However, you must strictly control who has read access to CICSTS56.CICS.SDFHAUTH, because it contains APF-authorized programs, and the profile protecting this data set **must** be defined with UACC(NONE). In Figure 1 on page 37 all of the partitioned data sets are defined with UACC(NONE) and have an explicit access list.

Although CICS modules exist in libraries SYS1.CICSTS56.CICS.SDFHLPA and SYS1.CICSTS56.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 z/OS Communications Server 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

on page 37, all the names have a high-level qualifier of CICSTS56.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 needs 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 'CICSTS56.CICS.appl1.dataset1'  ID(user or group)  ACCESS(READ)
PERMIT 'CICSTS56.CICS.appl1.dataset2'  ID(user or group)  ACCESS(READ)
PERMIT 'CICSTS56.CICS.appl2.dataset3'  ID(user or group)  ACCESS(UPDATE)
PERMIT 'CICSTS56.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. For guidance on using BWO, see Back-up-while-open (BWO).

ACCESS(ALTER) for VSAM data sets when specifying SMS Data Class attribute Dynamic Volume Count

Dynamic Volume Count

You can use SMS Data Class attribute Dynamic Volume Count to extend your VSAM data to multiple volumes. Depending on your SMS release level, ACCESS(ALTER) may be required by the CICS region userid to update the ICF catalog volume list during EOV extend processing. To determine the correct access level for your DFSMS release, see Required RACF Authorization Tables in z/OS DFSMS Access Method Services Commands.

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” on page 40 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” on page 41)
- CICS regions to the named counter servers (see “Access to named counter servers” on page 41).

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 user ID 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 user ID 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 42 for information about the responses to the CFDT server.

Access to named counter servers

You can control access to the named counter servers by CICS regions.

Each time that a CICS region connects to a named counter server, a security check is made against the CICS region userid to verify that the region is authorized to use the services of that named counter server.

Give each CICS region userid that connects to a named counter server UPDATE access to the CICS RACF profile called DFHNC.*poolname* in the FACILITY general resource class. This authorizes the CICS region to use the services of the named counter server for the 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. For CICS regions using VSAM record-level sharing (RLS), access to SMSVSAM servers is controlled by RACF security checks made against the CICS region userid to verify that the region is authorized to register with an SMSVSAM server.

In a test environment you might want to use the default action and allow any CICS region using VSAM RLS to connect to an SMSVSAM server. If you want 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.

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 z/OS Communications Server generic resources name if GRNAME 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 z/OS Communications Server generic resources name. All sign-on verifications in a CICSplex, where all the terminal-owning regions have the same z/OS Communications Server 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 z/OS Communications Server ACB

You can control which users among those who are running non-APF-authorized programs can OPEN the z/OS Communications Server SNA ACB associated with a CICS address space (CICS region).

This ensures that only authorized CICS regions can present themselves as z/OS Communications Server applications providing services with this APPLID, thus preventing unauthorized users impersonating real CICS regions. The CICS region user ID 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 user ID 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.

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
```

Note: VTAM is now the z/OS Communications Server (for SNA or IP)

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 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.

About this task

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 Surrogate user security.

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 the **SPOOLOPEN INPUT** command is used, 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 might also be done.

When the **SPOOLWRITE** command is used to write to the internal reader, CICS performs a surrogate user check to verify if the user is authorized to submit a job with the user ID specified on the job card. For more information, see Security for submitting a JCL job to the internal reader.

Authorizing use of HPO in PARM parameter on an EXEC PGM=DFHSIP statement or in SYSIN

When you specify **HPO** in the **PARM** parameter on an EXEC PGM=DFHSIP statement in the JCL of a CICS region or in the SYSIN data set of the CICS startup job stream, you should use a RACF profile to control the use of **HPO** and give the CICS region user ID necessary access to this profile.

Creating a RACF profile for HPO

You must create a RACF profile DFHSIT.HPO in the FACILITY class to define the necessary security authorization required for the use of **HPO** in the **PARM** parameter on an EXEC PGM=DFHSIP statement or in SYSIN:

Example:

```
RDEFINE FACILITY DFHSIT.HPO UACC(NONE)
```

UACC(NONE) means that the command rules apply, by default, to all users.

Giving CICS region user ID READ access to profile DFHSIT.HPO

You must give the associated region user ID READ access to the profile by using **PERMIT** command:

Example:

```
PERMIT DFHSIT.HPO CLASS(FACILITY) ID(CICS_region_userid) ACCESS(READ)
```

Security-related system initialization parameters

Several system initialization parameters are available for specifying system security requirements.

SEC
The SEC system initialization parameter specifies the level of resource security management you want for your CICS region. There are two options:

YES
Initializes the CICS external security interface; control of CICS security is determined by the other security-related system initialization parameters:

CMDSEC	**XAPPC**	**XPPT**
DFLTUSER	**XCMD**	**XPSB**
ESMEXITS	**XDB2**	**XRES**

PSBCHK	**XDCT**	**XRFSOFF**
PLTPISEC	**XFCT**	**XRFSTME**
RESSEC	**XHFS**	**XTRAN**
SECPRFX	**XJCT**	**XTST**
SNSCOPE	**XPCT**	**XUSER**

NO
No security checking is performed, and no user sign-on is available, for users who want to access CICS resources and other resources in this region.

Note: Even if you have specified SEC=NO, with MRO bind-time security, the CICS region user ID 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 268 for more information.

SECPRFX
This parameter is effective only if you also specify SEC=YES. 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 user ID 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 SECPRFX parameter has the following options:

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 user ID when passing authorization requests to RACF.

To change these values, use an ICHRTX00 SAF pre-processing exit. For more information, see "When and how CICS determines the user ID of the CICS region" on page 150.

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 main 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 example 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
Use CMDSEC to specify whether or not you want CICS to use 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 CMDSEC system initialization parameter.

DFLTUSER
Specify a value for DFLTUSER to identify to CICS the name you have defined to RACF as the default user ID. If you omit this parameter, the name defaults to CICSUSER. See "Defining the default CICS user ID to RACF" on page 34.

ESMEXITS
Use ESMEXITS to specify whether you want CICS to pass installation data for use by the RACF installation exits.

PLTPISEC
Use 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
Use 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
Use 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 user IDs that are signing on by explicit sign-on request; for example, the EXEC CICS SIGNON command or the CESN transaction. Use SNSCOPE to specify whether or not a user ID 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.

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 on page 49. Each class is shown with its 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
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 queues
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 resources subject to XRES security checks. For a list of resources subject to XRES security checks, see "Security using the XRES resource security parameter" on page 73.
XTRAN={YES|name|NO}	Attached transactions
XTST={YES|name|NO}	Temporary storage queues
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 19 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 the 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.

Table 5. Specifying external security with default resource classes (continued)	
System initialization parameter	**Effect**
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 on page 50.

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 and XDB2) 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 System initialization parameter descriptions and summary.

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 “Verifying CICS users” on page 53 and “Transaction security” on page 69 for information about specifying resource and command security for transactions.

XAPPC, XHFS, and XUSER

The syntax of the **XAPPC**, **XHFS**, and **XUSER** system initialization parameters is slightly different from the syntax of the other X*name* parameters. You can specify only 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 see CICS initialization failures related to security.

XAPPC enables RACF LU6.2 bind-time (also known as APPC) security. For more information see “Bind-time security with LU6.2” on page 224.

For more information see “Defining profiles in the APPCLU general resource class” on page 225.

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 90.

XUSER activates surrogate user security, and AUTHTYPE checking for Db2. For more information, see Surrogate user security. 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 23.

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 18.)

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 23.

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 18.)

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 CICSTS56.CICS.SDFHSAMP. See also “Specifying user-defined resources to RACF” on page 139.

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 23.

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 18.)

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 65 for more information about CICS terminal-user data, and how CICS obtains it.

Setting up PassTickets for secure sign-on

Using a PassTicket in place of a password means that applications do not have to store passwords (or ask users to re-enter them) in order to sign on to the destination system, and passwords are not transmitted across the network.

The *originating system* is the system where a PassTicket is generated, and the *destination system* is the system that the signed-on user ID attempts to access with the PassTicket and where the PassTicket is authenticated.

Before you begin

To use PassTickets, the systems involved must meet the following requirements:

- The PassTicket generation and validation algorithm means that the system that generates the PassTicket and the system that authenticates it must both use a level of the external security manager that supports PassTickets.
- End users must use the same user ID in the destination system as the one that they use in the originating system.
- Because PassTickets are time-stamped, the system clocks for the destination system and the originating system must be synchronized to within the valid time range. A PassTicket is considered to be within the valid time range when the time of generation, with respect to the clock on the generating computer, is within plus or minus 10 minutes of the time of evaluation, with respect to the clock on the evaluating computer. For more information about system time differences and synchronization, see Using the secured signon function in z/OS Security Server RACF Security Administrator's Guide.

Procedure

The following procedure assumes that RACF is the external security manager used in the implementation. If you use another external security manager, refer to the documentation for your product.

1. Define Secure Sign-on keys to enable the external security manager to process PassTickets.

   To process PassTickets, the external security manager uses Secure Sign-on keys that are shared by the originating system and the destination system. You must define a Secure Sign-on key for each destination system. For information about how to do this with RACF by defining profiles in the

PTKTDATA resource class, see Using the secured signon function in z/OS Security Server RACF Security Administrator's Guide.

2. Define RACF profiles to allow an originating system to generate a PassTicket.

   It is strongly recommended that you limit PassTicket generation to only those regions that require it. The regions should be set with the system initialization parameter **XPTKT=YES**. This is the default.

   This is a profile for users on a specific originating system:

   ```
   RDEFINE PTKTDATA IRRPTAUTH.applid.* UACC(NONE)
   PERMIT IRRPTAUTH.applid.* CLASS(PTKTDATA) ID(user) ACCESS(UPDATE)
   ```

   applid is the generic applid of the originating region.*user* is the user or group of users allowed to generate PassTickets on this region.

3. Define RACF profiles to allow a destination region to accept a PassTicket.

   ```
   RDEF PTKTDATA applid SSIGNON(key-description) UACC(NONE)
   ```

   applid is the generic applid of the destination region.

4. If RACLIST is used on PTKTDATA, refresh the definitions.

   Issue `SETR RACLIST(PTKTDATA) REFRESH`.

Supporting parallel PassTicket requests for the same user ID

PassTickets are one-time use tickets. The generation mechanism has a granularity of 1 second; therefore, if the same command to generate a PassTicket for a user ID is issued twice within the same second, it will generate the same PassTicket. This would limit requests that use PassTickets to one per second for each applid. However, it is possible to support parallel PassTicket requests for the same user ID.

To allow more than one request per second per user for an applid, you can define separate keys for separate groups for a user.

```
RDEF PTKTDATA applid.group1.userid SSIGNON(KEYMASKED(key1)) UACC(NONE)
RDEF PTKTDATA applid.group2.userid SSIGNON(KEYMASKED(key2)) UACC(NONE)
```

You must write custom code that generates PassTickets with the appropriate key.

For the destination region to accept the PassTicket, the region will need to use the group associated with the key when authenticating the password, for example, on the **SIGNON** or **VERIFY PASSWORD** command. If no group is specified, the default group is used.

Using PassTickets in FEPI applications

A FEPI application uses the PassTicket and user ID to perform a sign-on in the back-end system just as if it were sending a password and user ID.

Before you begin

To process PassTickets, the external security manager uses keys, known as Secure Signon keys, that are shared by the front-end and back-end systems. You must define a Secure Signon key for each target system with which FEPI communicates. For information about how to do this with RACF by defining profiles in the PTKTDATA resource class, see Using the secured signon function in z/OS Security Server RACF Security Administrator's Guide. Users of other ESMs should refer to the documentation for their product.

Procedure

1. The end user is verified by signing on to the front-end CICS system in the usual way.
2. When the end-user runs a transaction that uses FEPI, your application issues a **FEPI REQUEST PASSTICKET** command to obtain a PassTicket.

If EDF is being used, the PassTicket is not displayed. The user ID for which the PassTicket is generated is that of the currently signed-on user. Your FEPI application can use an **EXEC CICS ASSIGN** command to check the user ID of the currently signed-on user.

3. Your FEPI application uses the PassTicket and user ID to perform a sign-on in the back-end system. For example:

```
EXEC CICS FEPI SEND FORMATTED
                   CONVID(convid) FROM(CESN userid PassTicket)
                   FROMLENGTH(length_of_data)
```

   It is the application's responsibility to provide the sign-on processing, because CICS cannot know either the type of back-end system (CICS or IMS) or the back-end program being used for sign-on processing.

4. The back-end system uses an unchanged interface to perform the sign-on. A CICS system that receives a user ID and a PassTicket can use its existing procedures to sign on the user ID. RACF takes care of the fact that a PassTicket, rather than a password, is passed to it.

Results

If the PassTicket times out (because, for example, of a session failure), your application must generate another and try to sign on again. If sign-on continues to fail and the front-end and back-end are in different MVS systems, check that the system clocks are suitably synchronized. Too many failed sign-on attempts could result in the user ID being revoked.

What to do next

For detailed information about PassTickets, see Using the secured signon function in z/OS Security Server RACF Security Administrator's Guide.

The sign-on process

When users log-on to CICS through the z/OS Communications Server, 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

You can explicitly sign on either by using one of the CICS-supplied sign-on transactions CESL or CESN or by using an installation-provided sign-on transaction that uses the **EXEC CICS SIGNON** command.

OIDCARD users can use CESL or 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. When you sign 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 user ID you entered does not match a user ID already signed on in the scope of the **SNSCOPE** definition for the CICS system.

Identification
CICS calls RACF to confirm that a profile has been defined for your user ID.

Verification
CICS passes information to RACF to verify that the user ID is genuine. For RACF, this information is either a password or an OIDCARD or both. If the password entered is between 9 and 100 characters, RACF verifies it with the password phrase associated with your user ID. If the password is 8 characters or less, RACF verifies it with your standard password. A user ID can have both a standard password and a password phrase. If the password or password phrase has expired, CICS prompts you for a new password or password phrase. When the new password conforms to the RACF password formatting rules for an installation, the new password or password phrase and the date-of-change are recorded in your RACF user profile.

Passwords and password phrases operate independently of each other. You can sign on with a valid password phrase even if your standard password has expired. Similarly, you can sign on with a valid password if your password phrase has expired.

Immediately following the request to RACF for user ID and password verification, CICS clears the internal password field. This minimizes the possibility of the password or password phrase being revealed in any dump of the CICS address space that might be taken.

The CESL and CESN transactions do not warn users when a password or password phrase is about to expire. To display expiry warnings to users, you must replace CESL or CESN with your own transaction that calls a custom sign-on program. This program can use the **EXEC CICS VERIFY PASSWORD** or **EXEC CICS VERIFY PHRASE** commands to return expiry information from RACF, before using the **EXEC CICS SIGNON** command to make a full verification request for the user ID.

Authorization

RACF performs checks on the application name and the port of entry to verify that you are allowed to use the CICS system. In the application name check, RACF determines whether you are 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 43 for information about how to define profiles in the APPL resource class.)

With the port of entry check, RACF verifies that you are 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 57.

These checks restrict CICS users 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 CESL or CESN transaction, or the SIGNON command, is performed 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 API 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 timeout 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 57. 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 z/OS Communications Server.

In addition, if the terminal is auto-installed, 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.

SIGNOFF(NO)
CICS leaves the user signed on and the terminal remains logged on, effectively overriding the timeout 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: CESL or CESN will not sign the user off until a valid attempt has been made to use the CESL or CESN panel, even if the sign-on attempt subsequently fails. It is not recommended that CESL or CESN be used for the Goodnight transaction.

Implicit sign-on and implicit sign-off

Implicit sign-on means that all other user IDs added to the system by CICS are considered to be implicitly signed on without a password or password phrase.

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 61). 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 60.

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 59 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 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

You can control user access to a terminals using profiles in the TERMINAL and GTERMINL resource classes.

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 23. 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

You can 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 GTERMINL class

For CICS, the terminal profiles to define to RACF in the TERMINAL or GTERMINL class are used only for SNA LUs.

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-SNA LUs.

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

Using RACF, you can permit a user to access resources when that user is signed on a particular terminal or console, but not otherwise. Access that is restricted in this way is known as *conditional access*.

To grant conditional access to a resource, add

```
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 59:

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 z/OS MVS Planning: Operations 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 use this information in several ways: for example, as an audit trail, to identify possible attempts to breach security, and to help with capacity planning.

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)
```

Note: VTAM is now z/OS Communications Server (for SNA or IP).

For further information on preset-security terminals in the transaction routing environment, refer to “Preset-security terminals and transaction routing” on page 236 .

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 might 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 that is performing the installation. For more information, see Surrogate user security.

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:

  Restricting the use of the CEDA transactions to authorized users gives you control over who can define resources, such as terminals, to CICS. For information about protecting CICS-supplied transactions, see “Security for CICS transactions” on page 142
- The SURROGAT resource class:

If you restrict who can install terminals with preset security, even when such terminals are defined in the CSD, only authorized users can install them on CICS. This authority is in addition 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*

- 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 48 for information about defining this parameter.

- The LOCK command for locking CSD definitions

  CICS installs resource definitions in the CSD during an initial or cold start, from the list of groups that are defined on the **GRPLIST** system initialization parameter. To control the addition of resource groups to the CICS startup group list, use the CEDA LOCK command to lock the list. This lock 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 and .

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.

Other preset security considerations

If you intend to use preset security, you need to be aware of some additional considerations:

- 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 it is 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 with preset security:

  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 that are defined in the TCT:

  For terminals such as BSAM terminals that are defined in the terminal control table (TCT) by using DFHTCT macros, 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 might need authorization to use the MVS MODIFY command by using the OPERCMDS resource class.

You can 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 sign-on (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 or password phrase can generally be seen on the console and system log. However, if CICS is defined as an MVS subsystem in a JES2 system, you can use the HIDEPASSWORD=YES option of the DFHSSI*xx* member in SYS1.PARMLIB, which enables CICS to intercept the command and overwrite the password or password phrase with asterisks. For more information about defining CICS as an MVS subsystem, see Defining CICS as an MVS subsystem.

You can use the CESL or CESN commands to sign on to CICS from a console. With CESL, you can use either a standard password or a password phrase as authorization. CESL treats any password over 8 characters as a password phrase. CESN does not support the use of password phrases. The format of the CESL and CESN sign-on commands, when entered from a console, is as follows:

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

If a passphrase contains mixed-case characters, blanks, or both, the value of the PS parameter must be enclosed in single quotation marks, for example:

```
F JATP3250,CESL USERID=JAT232,PS='MOND July 17th'
```

The maximum length of the **MODIFY** command is 126 characters, including F *taskname*; the maximum length of a passphrase is 91 characters when it is specified in this way:

```
F JATP3250,CESL USERID=JAT284,PS='This is a valid long passphrase of
length 91 chars for CESL JAT testing 9XYZ@,#,.12345678!i'
```

For more information, see Guidance for issuing console commands in z/OS.

With CESL, the PS (password) and NEWPS (new password) parameters must match; they must be either both password phrases or both standard passwords. You cannot authorize a new password phrase that uses a password and you cannot authorize a new password by using a password phrase. The PS and NEWPS parameters can contain spaces or punctuation characters, including single quotation marks, but each of these characters must be enclosed in two single quotation marks.

If any of the data entered on the command is invalid, or if the password or password phrase is missing or expired, CICS terminates the sign-on attempt.

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

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

Users can sign on using CESL or CESN, or you might 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 user ID, by default, becomes a console name. However, you can change the console name to something else by 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 with the CONSNAME option in CICS (or if you autoinstall a terminal definition). If another name is 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 CESL or CESN transaction, their passwords might be exposed on the TSO screen and in the MVS system log. You can prevent these potential exposures by defining the terminal as having preset security. It is advisable to use automated preset security for several reasons:

- TSO users do not have to sign on, which avoids exposing their IDs and passwords on the log.
- You do not have to define a relationship in a CICS definition between a console name and a user. A relationship might change frequently or become invalid.
- You can define one autoinstall model that covers most 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 by using the user ID that first issues an MVSMVS 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 user ID imposes another overhead, which is equivalent to the console user signing on using CESL or 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, or 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 11 for details of the information that you can define in the CICS segment. See “The LANGUAGE segment” on page 15 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 for details of the information that you can define in the CICS segment. See The LANGUAGE segment for details of the LANGUAGE segment.
2. If RACF does not return the CICS segment or LANGUAGE segment data for the user, CICS uses the user attributes of the CICS default user, defined during system initialization. (See Obtaining CICS-related data for the default user.)

CICS obtains the national language attribute in the following order:

1. The LANGUAGE option on the CICS-supplied CESN transaction, or the LANGUAGECODE or NATLANG option of the **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.
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.

See National language codes 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 11 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 inquiries 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

Support for mixed-case passwords depends on your external security manager. All security managers that support password phrases, that is passwords between 9 and 100 characters in length, support mixed

case. However, not all security managers that support standard passwords, that is passwords up to 8 characters in length support mixed case.

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

You can enter a password using one of the two signon transactions, CESL and CESN or one of the following API commands:

CHANGE PASSWORD
CHANGE PHRASE
VERIFY PASSWORD
VERIFY PHRASE
SIGNON

CESL supports password phrases and standard passwords. CESN supports only standard passwords. These transaction have 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, CICS passes the password to the security manager unchanged.
- If not, 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 23.

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.

Transaction security

Transaction 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 46.)

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 46.)

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 19 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 2 on page 70 is an example which shows the main elements of CICS transaction security.

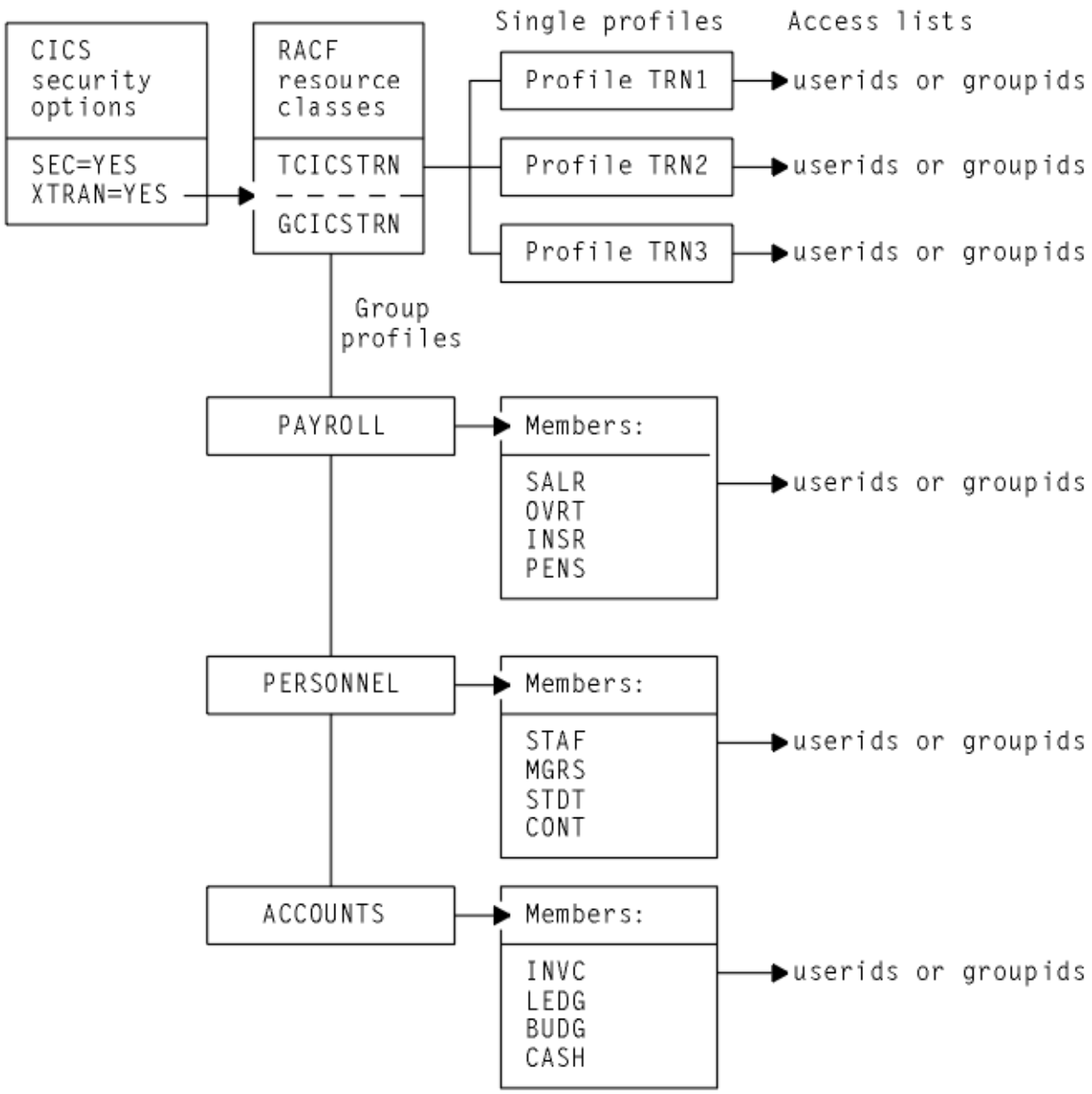


Figure 2. An example of the main elements of CICS transaction security

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

Every time a transaction is initiated at a CICS terminal, CICS issues an authorization request to determine whether the user associated with the terminal is authorized for that transaction.

CICS and RACF process the authorization request using the currently active transaction profiles in the RACF class identified by the XTRAN SIT parameter. (For more information, see "Refreshing resource profiles in main storage" on page 18.)

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.

About this task

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 19 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 in this document, 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.

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 z/OS Security Server RACF Messages and Codes.

For more information on resolving authorization problems, see Problem determination in a CICS-RACF security environment.

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 or an ICRX which maps to a userid 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.

Types of non-terminal transactions include the following:

- Transactions started by an **EXEC CICS START** command without a terminal identifier specified
- Transactions started by a **RUN TRANSID** command
- 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

Note: Distributed identity information is not preserved when a **START** command is run with any of the parameters **USERID** or **TERMID**, or is shipped across an LU61 or LU62 connection.

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 user IDs" on page 32. 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.

Resource security

Resource security provides a further level of security to transaction security, by controlling access to the resources used by the CICS transactions. A user who is authorized to invoke a particular CICS transaction might 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.

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.

For information about the access authorization levels for system programming commands, see "Resource and command check cross-reference" on page 96.

Security using the XRES resource security parameter

Use security profiles and the **XRES** system initialization parameter to security check a subset of CICS resources. An example of how to implement resource security for CICS document templates is provided.

About this task

A CICS security profile names consist of three parts: *security_prefix.resource_type.resource_name*. CICS profiles are passed to the security manager for checking. Security checking is case sensitive.

Security prefix

The *security_prefix* is the value specified on the **SECPRFX** system initialization parameter. The default value for the **SECPRFX** parameter is NO, which means that the *security_prefix* is omitted. If the value of

the **SECPRFX** parameter is YES, the *security_prefix* is the name of the region user ID. Alternatively, you can specify a 1- to 8-character value for the *security_prefix*.

Resource type

The *resource_type* specifies the type of resource against which checks are performed. In most instances, each CICS resource has a corresponding security profile; for example, the ATOMSERVICE resource has a security profile with a corresponding resource type:

```
security_prefix.ATOMSERVICE.resource_name
```

However, certain CICS resources do not have a corresponding security profile; for example, the BUNDLEPART, OSGIBUNDLE, and OSGISERVICE resources are checked using the BUNDLE security profile.

You can create security profiles that use the following resource types:

- ATOMSERVICE
- BUNDLE
- DOCTEMPLATE
- EPADAPTER
- EPADAPTERSET
- EVENTBINDING
- JVMSERVER
- XMLTRANSFORM

For a complete list of the CICS resources, resource types, and commands associated with security checks, see “Resource and command check cross-reference” on page 96.

Note: When you give a user authority to perform an action on a platform or application, you also give them authority to perform the same action on the dynamically generated resources for the platform or application. CICS resource security checks are not carried out when you create or operate on CICS bundles through an application or platform. However, CICS resource security checks do apply when you perform an action directly on an individual BUNDLE resource, or a resource that was defined in a CICS bundle, even if the bundle was created when you installed a platform or application. For more information, see Security for bundles.

Resource name

The *resource_name* specifies the name of the CICS resource.

Example

This example task explains how to implement resource security for CICS document templates.

CICS document templates are controlled in the following cases:

- Document templates delivered as a static response to a web client 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 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.

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. However, security checking for these commands uses the name of the DOCTEMPLATE resource that corresponds to the TEMPLATENAME attribute. So you can set up one profile name for each document template, using the name of the DOCTEMPLATE resource, 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.

Complete the following steps to implement resource security for CICS document templates:

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, refresh it using the SETROPTS command:

```
SETROPTS RACLIST(RCICSRES) REFRESH
```

2. Specify **SEC**=YES as a CICS system initialization parameter (and **SECPRFX**=YES 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).

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. If authentication is required for the connection but the client does not provide an authenticated user ID, the request is rejected.
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.

Resource security checking by CICS and RACF

CICS uses RACF to protect the resources that you can access through a CICS application program.

CICS and RACF resource checking summary

You use the CICS resource system initialization parameters to specify the RACF class name.

Each resource is described briefly in Table 8 on page 77, 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 “Resource and command check cross-reference” on page 96.

No authorization processing is done for BMS commands.

Table 8. General resource checking by CICS

CICS parameter	Resource protected	Further information
XAPPC	Partner logical units (LU6.2).	"Implementing LU6.2 security" on page 224.
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.	"CICS command security" on page 118
XDB2	Db2 resource classes for DB2ENTRY are specified to CICS on the XDB2 system initialization parameter.	"Resource classes for DB2ENTRY resources" on page 23
XDCT	CICS extrapartition and intrapartition transient data queues. Define profiles in the transient data class to control who is allowed to access CICS transient data queues.	"Security for transient data" on page 80.
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 82.
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 91.
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 83.
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 transactions" on page 84.
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 88.
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 93.
XRES	CICS resources that use the **XRES** parameter are: ATOMSERVICE, BUNDLE, DOCTEMPLATE, EPADAPTER, EPADAPTERSET, EVENTBINDING, JVMSERVER, and XMLTRANSFORM. For example, define profiles in the DOCTEMPLATE resource class to control who is allowed to access document templates.	"Security using the XRES resource security parameter" on page 73.

Table 8. General resource checking by CICS (continued)		
CICS parameter	**Resource protected**	**Further information**
XTRAN	CICS transactions.	“Transaction security” on page 69.
XTST	CICS temporary storage queues. Define profiles in the temporary storage class to control who is allowed to access CICS temporary storage queues.	“Security for temporary storage” on page 89.
XUSER	Surrogate user security.	Surrogate user security.

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 transaction-attach security.

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 queues (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)
- Resources subject to XRES security checks (XRES parameter), as listed in “Security using the XRES resource security parameter” on page 73.
- Temporary storage destinations (XTST parameter)

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 queue 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 "Security checking using the QUERY SECURITY command" on page 131), 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

Follow this procedure to implement security for transient data queues.

Procedure

1. Specify RESSEC(YES) in the 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, 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)
   ```

See also “Considerations for defining profiles for transient data queues” on page 81.

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).

Considerations for 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 “Resource and command check cross-reference” on page 96.

CICS-required transient data queue resource definitions

CICS itself uses a number of transient data queues. These queues are defined in group DFHDCTG, which is part of DFHLIST. 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 supplied resource definitions, most of the transient data queues are indirect, pointing to the transient data queues CSSL or CCSO. If you use the definitions as supplied, define to RACF only the queue names CSSL and CCSO, as follows:

```
RDEFINE  ECICSDCT  CICSQUEUES UACC(NONE)
                   ADDMEM(CSSL, CCSO)
                   NOTIFY(sys_admin_userid)
```

Considerations for triggered transactions

For intrapartition transient data queues with a trigger level greater than zero, CICS obtains the user ID associated with the triggered transaction from the following sources:

- For a transient data queue defined with ATIFACILITY(FILE), CICS uses the USERID parameter specified in the resource definition for the transient data queue.
- For a transient data queue defined with ATIFACILITY(TERMINAL), CICS uses the user ID associated with the terminal. This can be the CICS default userid if no user is signed on at the terminal.
- For a transient data queue defined with ATIFACILITY(SYSTEM), CICS uses the link user ID obtained by the CONNECTION resource definition.

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 “Resource and command check cross-reference” on page 96.

Security for journals and log streams

The CICS log manager provides facilities to write to and read from the CICS system log and CICS general logs. General logs comprise user journals, forward recovery logs, and autojournals. You can implement security for journals and log streams to protect them from unauthorized access.

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 YES on the **SEC** system initialization parameter, and YES on the **SECPRFX** if you define profiles with a prefix. CICS requires a minimum authorization of UPDATE for journal access. For information about the access authorization levels for system programming commands that apply to journals, see “Resource and command check cross-reference” on page 96.
4. Specify YES on the **XJCT** system initialization parameter for the default resource class names of JCICSJCT and KCICSJCT, or XJCT=*class_name* for user-defined resource class names. For more information, see XJCT system initialization parameter.

Transactions that use WRITE JOURNALNUM command

The WRITE JOURNALNUM command is supported in CICS Transaction Server for z/OS, Version 5 Release 6 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 transactions

A CICS transaction 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.

When a transaction issues an **EXEC CICS START TRANSID** command, CICS calls RACF to check that the user of the transaction that issues the command is authorized for the started transaction.

To implement security for started transactions and for transactions checked against the XPCT class:

1. Specify SEC=YES as a CICS system initialization parameter (and SECPRFX if you define profiles with a prefix).
2. Specify RESSEC(YES) in the CSD resource definition of the transactions that issue **START** commands.
3. Specify XPCT=YES for the default resource class names of ACICSPCT and BCICSPCT (or XPCT=class_name for user-defined resource class names).

   This ensures that when a transaction is started by a **START** command, CICS calls RACF to check that the userid associated with the transaction is authorized to attach the transaction.
4. 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)
   ```
5. Specify XTRAN=YES for the default resource class names of TCICSTRN and GCICSTRN (or XTRAN=class_name for user-defined resource class names).
6. Define profiles to RACF in the TCICSTRN or GCICSTRN resource classes (or their equivalent if you have user-defined resource class names) using the name of the started transaction to identify the profiles.

Transactions started at terminals

The START 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 61 for more information.

Transactions started without terminals

The EXEC CICS START command enables a CICS application program to start another transaction that is not associated with any terminal. When no TERMID is specified for the started transaction, the userid associated with the new transaction depends on whether you also specify the USERID option.

Userid of a non-terminal started transaction

The USERID option of the 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 222. For information about EDF authorization of surrogate users, see “Conditional access processing” on page 59.

Note: Distributed identity information is not preserved when a **START** command is run with any of the parameters: **USERID**, **TERMID**, or **RTERMID**, or is shipped across an LU61 or LU62 connection.

Access to resources by a non-terminal started transaction

If the USERID option is not specified on the STARTBROWSE PROCESS 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 59.

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 “Resource and command check cross-reference” on page 96.

Security for transactions started with EXEC CICS RUN TRANSID

A CICS transaction can initiate other transactions by means of an **EXEC CICS RUN TRANSID** command. You can use CICS RACF security to control who can initiate other transactions using the **RUN TRANSID** command.

When a transaction issues an **EXEC CICS RUN TRANSID** 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 asynchronous transactions, you need to do the following:

1. Specify SEC=YES as a CICS system initialization parameter (and SECPRFX if you define profiles with a prefix).
2. Specify RESSEC(YES) in the CSD resource definition of the parent transactions that issue **EXEC CICS RUN TRANSID** commands.

   This ensures that when a child transaction is started by an **EXEC CICS RUN TRANSID** command, CICS calls RACF to check that the userid associated with the transaction is authorized to attach the transaction.
3. Specify XPCT=YES for the default resource class names of ACICSPCT and BCICSPCT (or XPCT=class_name for user-defined resource class names).
4. 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 child transaction to identify the profiles.
5. Specify XTRAN=YES for the default resource class names of TCICSTRN and GCICSTRN (or XTRAN=class_name for user-defined resource class names).
6. Define profiles to RACF in the TCICSTRN or GCICSTRN resource classes (or their equivalent if you have user-defined resource class names) using the name of the started child transaction to identify the profiles.

Userid of a transaction started using EXEC CICS RUN TRANSID

A child transaction started by the **EXEC CICS RUN TRANSID** command runs under the USERID of the parent transaction which issued the command.

Access to resources by transactions started using EXEC CICS RUN TRANSID

- If a transaction-routed parent transaction executes an **EXEC CICS RUN TRANSID** command, the started child transaction is not subject to link security.
- If EDF is used in dual-screen mode for a transaction that issues an **EXEC CICS RUN TRANSID** command, the started transaction is not subject to resource access determined by the userid of the EDF terminal.

Access authorization levels

CICS requires a minimum authorization of READ for transactions started by an **EXEC CICS RUN TRANSID** command.

Security for XPCT-checked transactions

The **XPCT** system initialization parameter specifies whether CICS performs transaction resource security checking on transactions started by **EXEC CICS** commands, such as **EXEC CICS START** and **EXEC CICS RUN TRANSID**. Transactions defined in the ACICSPCT and BCICSPCT resource class profiles will be subjected to XPCT security checking.

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 **EXEC CICS** commands affected by XPCT-checking are:

- START

- COLLECT STATISTICS TRANSACTION
- DISCARD TRANSACTION
- INQUIRE TRANSACTION
- SET TRANSACTION
- INQUIRE REQID
- CANCEL
- RUN TRANSID

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 LINK, LOAD, XCTL, CICS RELEASE, ENABLE or DISABLE 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 “Resource and command check cross-reference” on page 96.

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 TSMODEL resources 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. 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” on page 89.
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).

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 for a TSMODEL resource definition that corresponds to the leading characters of the queue name.

Note that if you include 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 “Resource and command check cross-reference” on page 96.

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 that use the HFSFILE attribute, follow the steps listed in this topic.

Before you begin

The CICS region user ID must always have a minimum of **read** and **execute** permission 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 is unable to view it. This is the case even when the file is defined as a CICS document template.

About this task

Procedure

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 preferred 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 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 by using an analyzer program, as you can with application-generated responses.)

   Authenticated user IDs 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 - 16 777 216. To assign UIDs, specify the UID value in the OMVS segment of the user profile for each user ID.

   “RACF user profiles” on page 11 tells you how to update a RACF user profile by 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 based on the UID. z/OS UNIX System Services Planning 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 must 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, by using the `setfacl` command in the z/OS UNIX System Services shell environment.

   z/OS UNIX System Services Planning 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 must 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 must 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 before 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 and z/OS UNIX System Services User's Guide have information about using this command.

   (Note that as group permissions only can 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.

Results

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 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 Security checking with DBCTL for information on defining security in a CICS-DBCTL environment.

Security checking of transactions running under CEDF, CEDG, CEDX, or CEDY

When a transaction is run under one of the four EDF transactions, CICS checks the security settings for the target transaction.

The IBM-supplied definitions of CEDF, CEDG, CEDX, and CEDY in the DFHEDF group specify RESSEC(YES). Definitions in the IBM-supplied groups cannot be modified, so to change the definition, copy the transaction to another group.

When CEBR and CECI are invoked from within CEDF they are transaction-attach checked.

When one of the four EDF transactions is used to test a transaction, the authority of the user executing the transaction being tested is checked. For each resource accessed by the tested transaction, the user must have access authority, otherwise a NOTAUTH condition is raised. This requirement 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)
```

This 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.

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)
```

This 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)
```

Resource and command check cross-reference

Use this cross-reference to check commands against resources.

Table 9. 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" on page 112	XFCT	UPDATE	BTS REPOSITORY FILE		
ADDRESS					
ALLOCATE					
ASKTIME					
BIF DEEDIT					
BIF DIGEST					
BUILD ATTACH					
CANCEL See note "1" on page 112	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					

Table 9. Resource and command check cross-reference (continued)

EXEC CICS COMMAND	RESOURCE CHECK			CHECK CLASS=XCMD	
	CLASS	ACCESS	RESOURCE	ACCESS	RESOURCE
CONVERSE					
CREATE ATOMSERVICE	XRES	ALTER	ATOMSERVICE.*resource_name*	ALTER	ATOMSERVICE
CREATE BUNDLE	XRES	ALTER	BUNDLE.*resource_name*	ALTER	BUNDLE
CREATE CONNECTION See note "2" on page 112				ALTER	CONNECTION
CREATE DB2CONN See note "3" on page 112		ALTER			DB2CONN
CREATE DB2ENTRY See note "3" on page 112	XDB2	ALTER	DB2ENTRY	ALTER	DB2ENTRY
CREATE DB2TRAN See note "3" on page 112	XDB2	ALTER	DB2TRAN	ALTER	DB2TRAN
CREATE DOCTEMPLATE	XRES	ALTER	DOCTEMPLATE.*resource_name*	ALTER	DOCTEMPLATE
CREATE ENQMODEL				ALTER	ENQMODEL
CREATE FILE	XFCT	ALTER	FILE	ALTER	FILE
CREATE IPCONN				ALTER	IPCONN
CREATE JOURNALMODEL				ALTER	JOURNALMODEL
CREATE JVMSERVER	XRES	ALTER	JVMSERVER.*resource_name*	ALTER	JVMSERVER
CREATE LIBRARY				ALTER	LIBRARY
CREATE LSRPOOL				ALTER	LSRPOOL
CREATE MAPSET	XPPT	ALTER	MAPSET	ALTER	MAPSET
CREATE MQCONN				ALTER	MQCONN
CREATE MQMONITOR				ALTER	MQMON
CREATE PARTITIONSET	XPPT	ALTER	PARTITIONSET	ALTER	PARTITIONSET
CREATE PARTNER				ALTER	PARTNER
CREATE PIPELINE				ALTER	PIPELINE

Table 9. Resource and command check cross-reference (continued)

EXEC CICS COMMAND	RESOURCE CHECK			CHECK CLASS=XCMD	
	CLASS	ACCESS	RESOURCE	ACCESS	RESOURCE
CREATE PROCESSTYPE See note "10" on page 112				ALTER	PROCESSTYPE
CREATE PROFILE				ALTER	PROFILE
CREATE PROGRAM	XPPT	ALTER	PROGRAM	ALTER	PROGRAM
CREATE SESSIONS See note "3" on page 112				ALTER	SESSIONS
CREATE TCPIPSERVICE				ALTER	TCPIPSERVICE
CREATE TDQUEUE See note "3" on page 112	XDCT	ALTER	TDQUEUE	ALTER	TDQUEUE
CREATE TERMINAL See note "3" on page 112				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
CSD ADD				UPDATE	CSD
CSD ALTER				UPDATE	CSD
CSD APPEND				UPDATE	CSD
CSD COPY				UPDATE	CSD
CSD DEFINE				UPDATE	CSD
CSD DELETE				UPDATE	CSD
CSD DISCONNECT				READ	CSD
CSD ENDBRGROUP				READ	CSD
CSD ENDBRLIST				READ	CSD

Table 9. Resource and command check cross-reference (continued)

EXEC CICS COMMAND	RESOURCE CHECK			CHECK CLASS=XCMD	
	CLASS	ACCESS	RESOURCE	ACCESS	RESOURCE
CSD ENDBRRSRCE				READ	CSD
CSD GETNEXTGROUP				READ	CSD
CSD GETNEXTLIST				READ	CSD
CSD GETNEXTRSRCE				READ	CSD
CSD INQUIREGROUP				READ	CSD
CSD INQUIRELIST				READ	CSD
CSD INQUIRERSRCE				READ	CSD
CSD INSTALL				ALTER	CSD
CSD LOCK				UPDATE	CSD
CSD REMOVE				UPDATE	CSD
CSD RENAME				UPDATE	CSD
CSD STARTBRGROUP				READ	CSD
CSD STARTBRLIST				READ	CSD
CSD STARTBRRSRCE				READ	CSD
CSD UNLOCK				UPDATE	CSD
CSD USERDEFINE				UPDATE	CSD
DEFINE ACTIVITY See notes “7” on page 112 and “9” on page 112	XFCT	UPDATE	BTS REPOSITORY FILE		
DEFINE PROCESS See notes “7” on page 112 and “9” on page 112	XFCT	UPDATE	BTS REPOSITORY FILE		
DELAY					
DELETE	XFCT	UPDATE	FILE		
DELETE ACTIVITY See note “9” on page 112	XFCT	UPDATE	BTS REPOSITORY FILE		
DELETEQ TD	XDCT	UPDATE	TDQUEUE		
DELETEQ TS See note “4” on page 112	XTST	UPDATE	TSQUEUE		
DEQ					

Table 9. Resource and command check cross-reference (continued)

EXEC CICS COMMAND	RESOURCE CHECK			CHECK CLASS=XCMD	
	CLASS	ACCESS	RESOURCE	ACCESS	RESOURCE
DISABLE PROGRAM	XPPT	UPDATE	PROGRAM	UPDATE	EXITPROGRAM
DISCARD ATOMSERVICE	XRES	ALTER	ATOMSERVICE.*resource_name*	ALTER	ATOMSERVICE
DISCARD AUTINSTMODEL				ALTER	AUTINSTMODEL
DISCARD BUNDLE	XRES	ALTER	BUNDLE.*resource_name*	ALTER	BUNDLE
DISCARD CONNECTION					
DISCARD DB2CONN				ALTER	DB2CONN
DISCARD DB2ENTRY	XDB2	ALTER	DB2ENTRY	ALTER	DB2ENTRY
DISCARD DB2TRAN	XDB2	ALTER	DB2TRAN	ALTER	DB2TRAN
DISCARD DOCTEMPLATE	XRES	ALTER	DOCTEMPLATE.*resource_name*	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 JVMSERVER	XRES	ALTER	JVMSERVER.*resource_name*	ALTER	JVMSERVER
DISCARD LIBRARY				ALTER	LIBRARY
DISCARD MQCONN				ALTER	MQCONN
DISCARD MQMONITOR				ALTER	MQMON
DISCARD PARTNER				ALTER	PARTNER
DISCARD PIPELINE				ALTER	PIPELINE
DISCARD PROCESSTYPE See note “10” on page 112				ALTER	PROCESSTYPE
DISCARD PROFILE				ALTER	PROFILE
DISCARD PROGRAM	XPPT	ALTER	PROGRAM	ALTER	PROGRAM
DISCARD TCPIPSERVICE				ALTER	TCPIPSERVICE

Table 9. Resource and command check cross-reference (continued)

EXEC CICS COMMAND	RESOURCE CHECK			CHECK CLASS=XCMD	
	CLASS	ACCESS	RESOURCE	ACCESS	RESOURCE
DISCARD TDQUEUE	XDCT	ALTER	TDQUEUE	ALTER	TDQUEUE
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 See note 14	XRES	READ	DOCTEMPLATE.*resource_name*		
DOCUMENT INSERT See note 14	XRES	READ	DOCTEMPLATE.*resource_name* See note 13		
DUMP TRANSACTION					
ENABLE PROGRAM	XPPT	UPDATE	PROGRAM	UPDATE	EXITPROGRAM
ENDBR See note “5” on page 112					
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					

Table 9. Resource and command check cross-reference (continued)

EXEC CICS COMMAND	RESOURCE CHECK			CHECK CLASS=XCMD	
	CLASS	ACCESS	RESOURCE	ACCESS	RESOURCE
HANDLE CONDITION					
IGNORE CONDITION					
INQUIRE ACTIVITYID See note "9" on page 112	XFCT	READ	BTS REPOSITORY FILE		
INQUIRE ASSOCIATION				READ	ASSOCIATION
INQUIRE ATOMSERVICE	XRES	READ	ATOMSERVICE.*resource_name*	READ	ATOMSERVICE
INQUIRE AUTINSTMODEL				READ	AUTINSTMODEL
INQUIRE AUTOINSTALL				READ	AUTOINSTALL
INQUIRE BRFACILITY				READ	BRFACILITY
INQUIRE BUNDLE	XRES	READ	BUNDLE.*resource_name*	READ	BUNDLE
INQUIRE BUNDLEPART	XRES	READ	BUNDLE.*resource_name*	READ	BUNDLEPART
INQUIRE CAPDATAPRED	XRES	READ	EVENTBINDING.*resource_name*	READ	CAPDATAPRED
INQUIRE CAPINFOSRCE	XRES	READ	EVENTBINDING.*resource_name*	READ	CAPINFOSRCE
INQUIRE CAPOPTPRED	XRES	READ	EVENTBINDING.*resource_name*	READ	CAPOPTPRED
INQUIRE CAPTURESPEC	XRES	READ	EVENTBINDING.*resource_name*	READ	CAPTURESPEC
INQUIRE CFDTPOOL				READ	CFDTPOOL
INQUIRE CONNECTION				READ	CONNECTION
INQUIRE CONTAINER See note "9" on page 112	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

Table 9. Resource and command check cross-reference (continued)

EXEC CICS COMMAND	RESOURCE CHECK			CHECK CLASS=XCMD	
	CLASS	ACCESS	RESOURCE	ACCESS	RESOURCE
INQUIRE DOCTEMPLATE	XRES	READ	DOCTEMPLATE.*resource_name*	READ	DOCTEMPLATE
INQUIRE DSNAME				READ	DSNAME
INQUIRE DUMPDS				READ	DUMPDS
INQUIRE ENQMODEL				READ	ENQMODEL
INQUIRE EPADAPTER	XRES	READ	EPADAPTER.*resource_name*	READ	EPADAPTER
INQUIRE EPADAPTERSET	XRES	READ	EPADAPTERSET.*resource_name*	READ	EPADAPTERSET
INQUIRE EPADAPTINSET	XRES	READ	EPADAPTERSET.*resource_name*	READ	EPADAPTINSET
INQUIRE EVENT See note “9” on page 112	XFCT	READ	BTS REPOSITORY FILE		
INQUIRE EVENTBINDING	XRES	READ	EVENTBINDING.*resource_name*	READ	EVENTBINDING
INQUIRE EVENTPROCESS				READ	EVENTPROCESS
INQUIRE EXCI				READ	EXCI
INQUIRE EXITPROGRAM	XPPT	READ	PROGRAM	READ	EXITPROGRAM
INQUIRE FEATUREKEY				READ	SYSTEM
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 JVMENDPOINT				READ	JVMENDPOINT
INQUIRE JVMSERVER	XRES	READ	JVMSERVER.*resource_name*	READ	JVMSERVER
INQUIRE LIBRARY				READ	LIBRARY
INQUIRE MODENAME				READ	MODENAME
INQUIRE MONITOR				READ	MONITOR

Table 9. Resource and command check cross-reference (continued)

EXEC CICS COMMAND	RESOURCE CHECK			CHECK CLASS=XCMD	
	CLASS	ACCESS	RESOURCE	ACCESS	RESOURCE
INQUIRE MQCONN				READ	MQCONN
INQUIRE MQINI				READ	MQMON
INQUIRE MQMONITOR				READ	MQMON
INQUIRE MVSTCB				READ	MVSTCB
INQUIRE NETNAME				READ	TERMINAL
INQUIRE NODEJSAPP	XRES	READ	BUNDLE.*resource_name*	READ	NODEJSAPP
INQUIRE OSGIBUNDLE	XRES	READ	BUNDLE.*resource_name*	READ	OSGIBUNDLE
INQUIRE OSGISERVICE	XRES	READ	BUNDLE.*resource_name*	READ	OSGISERVICE
INQUIRE PARTNER				READ	PARTNER
INQUIRE PIPELINE				READ	PIPELINE
INQUIRE PROCESS See note "9" on page 112	XFCT	READ	BTS REPOSITORY FILE		
INQUIRE PROCESSTYPE See note "10" on page 112	XPTT			READ	PROCESSTYPE
INQUIRE PROFILE				READ	PROFILE
INQUIRE PROGRAM	XPPT	READ	PROGRAM	READ	PROGRAM
INQUIRE REQID See note "8" on page 112	XPCT	READ	TRANSID	READ	REQID
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

Table 9. Resource and command check cross-reference (continued)

EXEC CICS COMMAND	RESOURCE CHECK			CHECK CLASS=XCMD	
	CLASS	ACCESS	RESOURCE	ACCESS	RESOURCE
INQUIRE TASK				READ	TASK
INQUIRE TCLASS				READ	TCLASS
INQUIRE TCPIP				READ	TCPIP
INQUIRE TCPIPSERVICE				READ	TCPIPSERVICE
INQUIRE TDQUEUE	XDCT	READ	TDQUEUE	READ	TDQUEUE
INQUIRE TEMPSTORAGE				READ	TEMPSTORAGE
INQUIRE TERMINAL				READ	TERMINAL
INQUIRE TIMER See note “9” on page 112	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	TRANSID	READ	TRANSACTION
INQUIRE TSMODEL				READ	TSMODEL
INQUIRE TSPOOL				READ	TSPOOL
INQUIRE TSQNAME See note “4” on page 112	XTST	READ	TSQNAME	READ	TSQUEUE
INQUIRE TSQUEUE See note “4” on page 112	XTST	READ	TSQUEUE	READ	TSQUEUE
INQUIRE UOW				READ	UOW
INQUIRE UOWDSNFAIL				READ	UOWDSNFAIL
INQUIRE UOWENQ				READ	UOWENQ
INQUIRE UOWLINK				READ	UOWLINK
INQUIRE URIMAP				READ	URIMAP

Table 9. Resource and command check cross-reference (continued)

EXEC CICS COMMAND	RESOURCE CHECK			CHECK CLASS=XCMD	
	CLASS	ACCESS	RESOURCE	ACCESS	RESOURCE
INQUIRE VTAM				READ	VTAM (now the z/OS Communications Server)
INQUIRE WEB				READ	WEB
INQUIRE WEBSERVICE				READ	WEBSERVICE
INQUIRE WLMHEALTH				READ	WLMHEALTH
INQUIRE XMLTRANSFORM	XRES	READ	XMLTRANSFORM.*resource_name*	READ	XMLTRANSFORM
ISSUE					
LINK	XPPT	READ	PROGRAM		
LINK ACQPROCESS See note “9” on page 112	XFCT	UPDATE	BTS REPOSITORY FILE		
LINK ACTIVITY / ACQACTIVITY See note “9” on page 112	XFCT	UPDATE	BTS REPOSITORY FILE		
LOAD	XPPT	READ	PROGRAM		
MONITOR					
PERFORM DELETSHIPPED				UPDATE	DELETSHIPPED
PERFORM DUMP				UPDATE	DUMP
PERFORM JVMSERVER				UPDATE	JVMSERVER
PERFORM PIPELINE				UPDATE	PIPELINE
PERFORM RESETTIME				UPDATE	RESETTIME
PERFORM SECURITY REBUILD				UPDATE	SECURITY
PERFORM SHUTDOWN				UPDATE	SHUTDOWN
PERFORM SSL REBUILD				UPDATE	SECURITY
PERFORM STATISTICS				UPDATE	STATISTICS

Table 9. Resource and command check cross-reference (continued)

EXEC CICS COMMAND	RESOURCE CHECK			CHECK CLASS=XCMD	
	CLASS	ACCESS	RESOURCE	ACCESS	RESOURCE
POINT					
POP HANDLE					
POST					
PURGE MESSAGE					
PUSH HANDLE					
QUERY SECURITY See note “6” on page 112					
QUERY SECURITY USERID See note “17” on page 113					
READ	XFCT	READ	FILE		
READ NEXT See note “5” on page 112					
READ PREV See note “5” on page 112					
READQ TD	XDCT	UPDATE	TDQUEUE		
READQ TS See note “4” on page 112	XTST	READ	TSQUEUE		
RECEIVE					
RELEASE	XPPT	READ	PROGRAM		
RESET ACTIVITY See note “9” on page 112	XFCT	UPDATE	BTS REPOSITORY FILE		
RESET ACQPROCESS See note “9” on page 112	XFCT	UPDATE	BTS REPOSITORY FILE		
RESETBR See note “5” on page 112					

Table 9. Resource and command check cross-reference (continued)

EXEC CICS COMMAND	RESOURCE CHECK			CHECK CLASS=XCMD	
	CLASS	ACCESS	RESOURCE	ACCESS	RESOURCE
RESYNC ENTRYNAME				UPDATE	EXITPROGRAM
RETRIEVE					
RETURN / RETURN ENDACTIVITY See note "9" on page 112 and "11" on page 112	XFCT	UPDATE	BTS REPOSITORY FILE		
REWRITE	XFCT	UPDATE	FILE		
ROUTE					
RUN See note "9" on page 112 and "11" on page 112	XFCT	UPDATE	BTS REPOSITORY FILE		
RUN / ASYNCH SYNC See note "9" on page 112	XFCT	UPDATE	DFHLRQ FILE		
SEND					
SET ATOMSERVICE	XRES	UPDATE	ATOMSERVICE.*resource_name*	UPDATE	ATOMSERVICE
SET AUTOINSTALL				UPDATE	AUTOINSTALL
SET BRFACILITY				UPDATE	BRFACILITY
SET BUNDLE	XRES	UPDATE	BUNDLE.*resource_name*	UPDATE	BUNDLE
SET CONNECTION				UPDATE	CONNECTION
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.*resource_name*	UPDATE	DOCTEMPLATE
SET DSNAME				UPDATE	DSNAME
SET DUMPDS				UPDATE	DUMPDS
SET ENQMODEL				UPDATE	ENQMODEL
SET EPADAPTER	XRES	UPDATE	EPADAPTER.*resource_name*	UPDATE	EPADAPTER
SET EPADAPTERSET	XRES	UPDATE	EPADAPTERSET.*resource_name*	UPDATE	EPADAPTERSET
SET EVENTBINDING	XRES	UPDATE	EVENTBINDING.*resource_name*	UPDATE	EVENTBINDING
SET EVENTPROCESS	XRES	UPDATE	EVENTPROCESS.*resource_name*	UPDATE	EVENTPROCESS

Table 9. Resource and command check cross-reference (continued)

EXEC CICS COMMAND	RESOURCE CHECK			CHECK CLASS=XCMD	
	CLASS	ACCESS	RESOURCE	ACCESS	RESOURCE
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 JVMENDPOINT				UPDATE	JVMENDPOINT
SET JVMSERVER	XRES	UPDATE	JVMSERVER.*resource_name*	UPDATE	JVMSERVER
SET LIBRARY				UPDATE	LIBRARY
SET MODENAME				UPDATE	MODENAME
SET MONITOR				UPDATE	MONITOR
SET MQCONN See note “13” on page 112				UPDATE	MQCONN
SET MQMONITOR				UPDATE	MQMON
SET NETNAME				UPDATE	TERMINAL
SET PIPELINE				UPDATE	PIPELINE
SET PROCESSTYPE See note “10” on page 112	XPTT			UPDATE	PROCESSTYPE
SET PROGRAM	XPPT	UPDATE	PROGRAM	UPDATE	PROGRAM
SET PROGRAM REPLICATION See note “16” on page 113				ALTER	REPLICATION
SET STATISTICS				UPDATE	STATISTICS
SET SYSDUMPCODE See note 15				UPDATE	SYSDUMPCODE
SET SYSTEM				UPDATE	SYSTEM
SET TASK				UPDATE	TASK
SET TCLASS				UPDATE	TCLASS
SET TCPIP				UPDATE	TCPIP
SET TCPIPSERVICE				UPDATE	TCPIPSERVICE

Table 9. Resource and command check cross-reference (continued)

EXEC CICS COMMAND	RESOURCE CHECK			CHECK CLASS=XCMD	
	CLASS	ACCESS	RESOURCE	ACCESS	RESOURCE
SET TDQUEUE See note "3" on page 112	XDCT	UPDATE	TDQUEUE	UPDATE	TDQUEUE
SET TEMPSTORAGE				UPDATE	TEMPSTORAGE
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 See note "4" on page 112	XTST	UPDATE	TSQUEUE	UPDATE	TSQUEUE
SET TSQUEUE See note "4" on page 112	XTST	UPDATE	TSQNAME	UPDATE	TSQUEUE
SET UOW				UPDATE	UOW
SET UOWLINK				UPDATE	UOWLINK
SET URIMAP				UPDATE	URIMAP
SET VTAM				UPDATE	VTAM
SET WEB				UPDATE	WEB
SET WEBSERVICE				UPDATE	WEBSERVICE
SET WLMHEALTH				UPDATE	WLMHEALTH
SET XMLTRANSFORM	XRES	UPDATE	XMLTRANSFORM.*resource_name*	UPDATE	XMLTRANSFORM
SIGNOFF					
SIGNON					
SPOOLCLOSE					
SPOOLOPEN					
SPOOLREAD					
SPOOLWRITE					

Table 9. Resource and command check cross-reference (continued)

EXEC CICS COMMAND	RESOURCE CHECK			CHECK CLASS=XCMD	
	CLASS	ACCESS	RESOURCE	ACCESS	RESOURCE
START See note "7" on page 112	XPCT	READ	TRANSID		
STARTBR	XFCT	READ	FILE		
STARTBROWSE ACTIVITY See note "9" on page 112	XFCT	READ	BTS REPOSITORY FILE		
STARTBROWSE CONTAINER (BTS) See note "9" on page 112	XFCT	READ	BTS REPOSITORY FILE		
STARTBROWSE EVENT See note "9" on page 112	XFCT	READ	BTS REPOSITORY FILE		
STARTBROWSE PROCESS See note "9" on page 112	XFCT	READ	BTS REPOSITORY FILE		
SUSPEND See note "9" on page 112	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		

Table 9. Resource and command check cross-reference (continued)					
EXEC CICS COMMAND	**RESOURCE CHECK**			**CHECK CLASS=XCMD**	
	CLASS	**ACCESS**	**RESOURCE**	**ACCESS**	**RESOURCE**
WRITE JOURNALNUM	XJCT	UPDATE	DFHJNN		
WRITE OPERATOR					
WRITEQ TD	XDCT	UPDATE	TDQUEUE		
WRITEQ TS See note "4" on page 112	XTST	UPDATE	TSQUEUE|TSQNAME		
XCTL	XPPT	READ	PROGRAM		

Notes:

1. The **CANCEL** command can perform two checks. One check is performed against the transaction specified on the **CANCEL** command. The other check is performed against the transaction associated with the REQID that 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** commands. The **COMPLETE** and **DISCARD** commands can be used only 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. For more information, see "Situations where surrogate user checking applies" on page 113.
4. A security check is performed for temporary storage queues when SECURITY(YES) is specified in the relevant TSMODEL resource definition, or, if you use a temporary storage table (TST), when a DFHTST TYPE=SECURITY macro is coded for the temporary storage queue.
5. No security check is performed, because the **STARTBR** command must be issued before this command. A security check is issued on the **STARTBR** command.
6. The **QUERY SECURITY** command is not controlled by resource or command checks, but it can cause them to be issued.
7. A start surrogate user check can also occur.
8. The resource check for the transid is performed only if the REQID is associated with a transaction.
9. CICS BTS API commands.
10. CICS BTS commands that are subject to command security. All other CICS BTS commands are not subject to command-level security.
11. Any BTS commands that use timing operands access the BTS LRQ file.
12. 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.
13. When command security is active for your CICS region, to use the **EXEC CICS SET MQCONN** command to start or stop the connection to IBM MQ, users must have authority to use not only the **EXEC CICS SET MQCONN** command, but also the **EXEC CICS EXTRACT EXIT** command. If a user attempts to start or stop the connection without the authority to use the **EXEC CICS EXTRACT EXIT** command, CICS issues messages DFHXS1111 and DFHMQ0302. When resource security for

programs (XPPT) is active for your CICS region, when using SET to set the Connection Status to Connected, the user must have sufficient authority to run the supplied CICS MQ programs.

14. 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 **EXEC CICS DOCUMENT CREATE** commands with the TEMPLATE option.
15. For CHECK CLASS=XCMD, you must have at least CONTROL access to update the **JOBLIST** or **DSPLIST** option on **SET SYSDUMPCODE**.
16. If command security checking is active, besides UPDATE access already required by **SET PROGRAM**, to issue **SET PROGRAM REPLICATION**, you must also have ALTER access to the RACF REPLICATION resource. REPLICATION is treated as a CICS resource in the RACF definitions.
17. A surrogate check for userid.DFHQUERY. Task user must have at least READ access.

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.

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 following items:

- The CICS default user
- PLT post-initialization processing
- Preset terminal security
- Started transactions
- The user ID associated with a CICS business transaction services (BTS) process or activity that is started by a RUN command
- The user ID associated with a transient data destination
- The user ID supplied as a parameter on an EXCI call
- The user ID supplied on the AUTHID and COMAUTHID attributes of the DB2CONN and DB2ENTRY resource definitions
- The user ID supplied on the USERID attribute of URIMAP resource definitions
- The user ID supplied on the transaction user ID of an event processing transaction start adapter.
- A CICSPlex SM MAS agent started with the COLM transaction
- A CICSPlex SM local MAS agent started with the CORM transaction
- The user IDs involved in JCL job submissions to the JES internal reader

Note: When you use the CICSPlex SM interface to install a resource definition that is subject to surrogate user security, the surrogate user that is checked by CICS is the user ID that started the CICSPlex SM agent in the region where the resource is installed.

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 user ID is authorized as a surrogate user of the user ID specified in the PLTPIUSR system initialization parameter.

The **PLTPIUSR** system initialization parameter specifies the user ID that CICS is to use for PLT programs that run during CICS initialization. All PLT programs run under the authority of the specified user ID, including the CPLT transaction, which must be authorized to all the resources referenced by the programs.

The scope of PLT security checking is defined by the PLTPISEC system initialization parameter. This parameter 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 user ID, in which case CICS does not perform a surrogate user check. However, the CICS region user ID must be authorized to all the resources referenced by the PLT programs. Furthermore, the CICS region user ID 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 installation is authorized as a surrogate user of the preset userid.

This is discussed in "Preset terminal security" on page 61.

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 user ID under which the transaction issuing the START command runs is called the *starting-userid*, and the user ID 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 user IDs associated with the transaction issuing the START are surrogates of the *started-userid*. CICS also assumes that any user ID is always a surrogate of itself. So user IDs 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 user IDs that are different from *starting-userid* when it is using CICS intercommunication, and when it is using EDF in the two-terminal mode.

If neither **USERID** nor **TERMID** is specified on the **START** command, surrogate checking is not done, because the *starting-userid* and the *started-userid* are assumed to be the same. If an ICRX is available, CICS passes it to the started task and the started task inherits the distributed identity used by the ICRX.

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 222.

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.

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 DEFINE PROCESS or DEFINE ACTIVITY. 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 user ID 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 user ID, which must be authorized to all the resources used by the transaction.

If you omit the user ID from a qualifying trigger-level entry, CICS uses the default user ID specified on the DFLTUSER system initialization parameter. Ensure that the user ID of any CICS region in which the transient data queue definition is installed is defined as a surrogate of all the user IDs 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 user ID against all the user IDs 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.

Default job userid for TDQ

When a transient data queue is defined with a **JOBUSERID** parameter, a surrogate user security check is performed against the user that installed the definition.

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, it is authorized as a surrogate of the userid specified on the DPL_request call).

External CICS interface (EXCI) client jobs are subject to surrogate user checking. So you must 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 *execution_userid*.DFHEXCI in the SURROGAT general resource class (where *execution_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 execution_userid.DFHEXCI UACC(NONE)
PERMIT   execution_userid.DFHEXCI CLASS(SURROGAT) ID(batch_region_userid)
                  ACCESS(READ)
```

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

For more information about these attributes, see DB2CONN resources and DB2ENTRY resources.

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. For more information, see Security in a CICS DB2 environment.

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 this resource definition, see URIMAP resources.

RACF definitions for surrogate user checking

To enable CICS surrogate user checking, you define the appropriate SURROGAT class profiles for CICS in the RACF database and you authorize CICS surrogate users to the appropriate SURROGAT profiles.

You can define three forms of surrogate class profile names 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 user ID under which a started transaction is to run
- The user ID 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 user ID specified on the PLTPIUSR system initialization parameter
- The user ID associated with a trigger-level transaction
- The CICS default user ID specified on the DFLTUSER system initialization parameter
- The user ID specified for preset terminal security
- The user ID specified on the AUTHID or COMAUTHID parameter of a Db2 resource definition
- The user ID supplied on the USERID attribute of URIMAP resource definitions
- The user ID supplied on the transaction user ID of an event processing transaction start adapter.

If the user ID that is associated with a task issuing either a CREATE IPCONN or CREATE CONNECTION command is not an authorized surrogate of the user specified in the SECURITYNAME option, a NOTAUTH error is returned.

***userid*.DFHQUERY**
userid represents the user ID of the user whose access to a resource is to be queried.

You can also define a form of surrogate class profile 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 user ID of the EXCI batch region READ access.

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 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 its own surrogate. In this situation, CICS bypasses the surrogate check.

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 make 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 user ID must be authorized as a surrogate of the PLT user ID that is defined on the **PLTPIUSR** system initialization parameter.

You must grant the CICS region user ID access to a SURROGAT resource class profile owned by the PLT user ID, as shown in the following example. In the example, the CICS region user ID is CICSHT01 and the PLT security user ID 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 user ID to the access lists of all the resources accessed by PLT programs. For example, if you specify PLTPISEC=RESSEC, ensure that the PLT user ID 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 114 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 141.

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 main terminal transaction.

Table 10 on page 119 shows the commands that are subject to command security checking:

Table 10. Access required for system programming commands	
Command name	**Access required**
COLLECT CSD DISCONNECT CSD ENDBRGROUP CSD ENDBRLIST CSD ENDBRRSRCE CSD GETNEXTGROUP CSD GETNEXTLIST CSD GETNEXTRSRCE CSD INQUIREGROUP CSD INQUIRELIST CSD INQUIRERSRCE CSD STARTBRGROUP CSD STARTBRLIST CSD STARTBRRSRCE EXTRACT STATISTICS INQUIRE	READ
DISABLE CSD ADD CSD ALTER CSD APPEND CSD COPY CSD DEFINE CSD DELETE CSD LOCK CSD REMOVE CSD RENAME CSD UNLOCK CSD USERDEFINE ENABLE EXTRACT (but not EXTRACT STATISTICS) PERFORM RESYNC SET	UPDATE
CREATE CSD INSTALL DISCARD	ALTER
Note: Because the PERFORM CORBASERVER SCAN might 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 that 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.

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. It 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 that you have assigned to them; for example, file names, queue names, and transaction names. 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 with the associated commands is shown in Table 11 on page 120. Most of these commands are common to both the CEMT and EXEC CICS interfaces; commands that are specific to CEMT have the CEMT prefix.

If you use prefixing, the value specified by the SECPRFX SIT parameter must be prefixed to the command resource name.

Table 11. Resource identifiers and associated commands for CICS resources subject to command security checking

Resource identifier	Related CICS commands
ASSOCIATION	**INQUIRE ASSOCIATION**
ATOMSERVICE	**CREATE ATOMSERVICE** **DISCARD ATOMSERVICE** **INQUIRE ATOMSERVICE** **SET ATOMSERVICE**
AUTINSTMODEL	**DISCARD AUTINSTMODEL** **INQUIRE AUTINSTMODEL**
AUTOINSTALL	**INQUIRE AUTOINSTALL** **SET AUTOINSTALL**
BRFACILITY	**INQUIRE BRFACILITY** **SET BRFACILITY**
BUNDLE	**CREATE BUNDLE** **DISCARD BUNDLE** **INQUIRE BUNDLE** **SET BUNDLE**
BUNDLEPART	**INQUIRE BUNDLEPART**
CAPDATAPRED	**INQUIRE CAPDATAPRED**
CAPINFOSRCE	**INQUIRE CAPINFOSRCE**
CAPOPTPRED	**INQUIRE CAPOPTPRED**
CAPTURESPEC	**INQUIRE CAPTURESPEC**
CFDTPOOL	**INQUIRE CFDTPOOL**

Table 11. Resource identifiers and associated commands for CICS resources subject to command security checking (continued)

Resource identifier	Related CICS commands
CONNECTION	**CREATE CONNECTION** **DISCARD CONNECTION** **INQUIRE CONNECTION** **SET CONNECTION**
CSD	**CSD ADD** **CSD ALTER** **CSD APPEND** **CSD COPY** **CSD DEFINE** **CSD DELETE** **CSD DISCONNECT** **CSD ENDBRGROUP** **CSD ENDBRLIST** **CSD ENDBRRSRCE** **CSD GETNEXTGROUP** **CSD GETNEXTLIST** **CSD GETNEXTRSRCE** **CSD INQUIREGROUP** **CSD INQUIRELIST** **CSD INQUIRERSRCE** **CSD INSTALL** **CSD LOCK** **CSD REMOVE** **CSD RENAME** **CSD STARTBRGROUP** **CSD STARTBRLIST** **CSD STARTBRRSRCE** **CSD UNLOCK** **CSD USERDEFINE**
DB2CONN	**CREATE DB2CONN** **DISCARD DB2CONN** **INQUIRE DB2CONN** **SET DB2CONN**
DB2ENTRY	**CREATE DB2ENTRY** **DISCARD DB2ENTRY** **INQUIRE DB2ENTRY** **SET DB2ENTRY**
DB2TRAN	**CREATE DB2TRAN** **DISCARD DB2TRAN** **INQUIRE DB2TRAN** **SET DB2TRAN**
DELETSHIPPED	**INQUIRE DELETSHIPPED** **PERFORM DELETSHIPPED** **SET DELETSHIPPED**

Table 11. Resource identifiers and associated commands for CICS resources subject to command security checking (continued)

Resource identifier	Related CICS commands
DISPATCHER	**INQUIRE DISPATCHER** **SET DISPATCHER**
DOCTEMPLATE	**CREATE DOCTEMPLATE** **DISCARD DOCTEMPLATE** **INQUIRE DOCTEMPLATE** **SET DOCTEMPLATE**
DSNAME	**INQUIRE DSNAME** **SET DSNAME**
DUMP	**CEMT PERFORM SNAP** **PERFORM DUMP**
DUMPCODE	**CREATE DUMPCODE**
DUMPDS	**INQUIRE DUMPDS** **SET DUMPDS**
ENQMODEL	**CREATE ENQMODEL** **INQUIRE ENQMODEL** **SET ENQMODEL**
EPADAPTER	**INQUIRE EPADAPTER** **SET EPADAPTER** Bundle command security applies when you use SPI commands to perform an action on a BUNDLE resource, and in that process you install, enable, disable, or discard a dynamically generated resource of this type that was defined in the CICS bundle. No CICS command security applies when you install, enable, disable, or discard a dynamically generated resource of this type through an application or platform. For more information, see Security for bundles.
EPADAPTERSET	**INQUIRE EPADAPTERSET** **SET EPADAPTERSET** Bundle command security applies when you use SPI commands to perform an action on a BUNDLE resource, and in that process you install, enable, disable, or discard a dynamically generated resource of this type that was defined in the CICS bundle. No CICS command security applies when you install, enable, disable, or discard a dynamically generated resource of this type through an application or platform. For more information, see Security for bundles.

Table 11. Resource identifiers and associated commands for CICS resources subject to command security checking (continued)

Resource identifier	Related CICS commands
EPADAPTINSET	**INQUIRE EPADAPTINSET** Bundle command security applies when you use SPI commands to perform an action on a BUNDLE resource, and in that process you install, enable, disable, or discard a dynamically generated resource of this type that was defined in the CICS bundle. No CICS command security applies when you install, enable, disable, or discard a dynamically generated resource of this type through an application or platform. For more information, see Security for bundles.
EVENTBINDING	**INQUIRE EVENTBINDING** **SET EVENTBINDING** Bundle command security applies when you use SPI commands to perform an action on a BUNDLE resource, and in that process you install, enable, disable, or discard a dynamically generated resource of this type that was defined in the CICS bundle. No CICS command security applies when you install, enable, disable, or discard a dynamically generated resource of this type through an application or platform. For more information, see Security for bundles.
EVENTPROCESS	**INQUIRE EVENTPROCESS** **SET EVENTPROCESS**
EXCI	**INQUIRE EXCI**
EXITPROGRAM	**DISABLE PROGRAM** **ENABLE PROGRAM** **EXTRACT EXIT** **RESYNC ENTRYNAME** **INQUIRE EXITPROGRAM**
FEPIRESOURCE	Certain **FEPI** commands
FILE	**CREATE FILE** **DISCARD FILE** **INQUIRE FILE** **SET FILE**
HOST	**INQUIRE HOST** **SET HOST**
IPCONN	**CREATE IPCONN** **DISCARD IPCONN** **INQUIRE IPCONN** **SET IPCONN**
IRC	**INQUIRE IRC** **SET IRC**

Table 11. Resource identifiers and associated commands for CICS resources subject to command security checking (continued)

Resource identifier	Related CICS commands
JOURNALMODEL	**CEMT INQUIRE JMODEL** **CREATE JOURNALMODEL** **DISCARD JOURNALMODEL** **INQUIRE JOURNALMODEL**
JOURNALNAME	**INQUIRE JOURNALNAME** **SET JOURNALNAME**
JVMENDPOINT	**INQUIRE JVMENDPOINT** **SET JVMENDPOINT**
JVMSERVER	**CREATE JVMSERVER** **DISCARD JVMSERVER** **INQUIRE JVMSERVER** **PERFORM JVMSERVER** **SET JVMSERVER**
LIBRARY	**CREATE LIBRARY** **DISCARD LIBRARY** **INQUIRE LIBRARY** **SET LIBRARY** Bundle command security applies when you use SPI commands to perform an action on a BUNDLE resource, and in that process you install, enable, disable, or discard a dynamically generated resource of this type that was defined in the CICS bundle. No CICS command security applies when you install, enable, disable, or discard a dynamically generated resource of this type through an application or platform. For more information, see Security for bundles.
LINE	**CEMT INQUIRE LINE** **CEMT SET LINE**
LSRPOOL	**CREATE LSRPOOL**
MAPSET	**CREATE MAPSET**
MODENAME	**INQUIRE MODENAME** **SET MODENAME**
MONITOR	**INQUIRE MONITOR** **SET MONITOR**
MQCONN	**CREATE MQCONN** **DISCARD MQCONN** **INQUIRE MQCONN** **SET MQCONN**

Table 11. Resource identifiers and associated commands for CICS resources subject to command security checking (continued)

Resource identifier	Related CICS commands
MQMON	**CREATE MQMONITOR** **DISCARD MQMONITOR** **INQUIRE MQMONITOR** **SET MQMONITOR**
MVSTCB	**COLLECT STATISTICS** **INQUIRE MVSTCB**
NODEJSAPP	**INQUIRE NODEJSAPP** Bundle command security applies when you use SPI commands to perform an action on a BUNDLE resource, and in that process you install, enable, disable, or discard a dynamically generated resource of this type that was defined in the CICS bundle. No CICS command security applies when you install, enable, disable, or discard a dynamically generated resource of this type through an application or platform. For more information, see Security for bundles.
OSGIBUNDLE	**INQUIRE OSGIBUNDLE** Bundle command security applies when you use SPI commands to perform an action on a BUNDLE resource, and in that process you install, enable, disable, or discard a dynamically generated resource of this type that was defined in the CICS bundle. No CICS command security applies when you install, enable, disable, or discard a dynamically generated resource of this type through an application or platform. For more information, see Security for bundles.
OSGISERVICE	**INQUIRE OSGISERVICE** Bundle command security applies when you use SPI commands to perform an action on a BUNDLE resource, and in that process you install, enable, disable, or discard a dynamically generated resource of this type that was defined in the CICS bundle. No CICS command security applies when you install, enable, disable, or discard a dynamically generated resource of this type through an application or platform. For more information, see Security for bundles.
PARTITIONSET	**CREATE PARTITIONSET**
PARTNER	**CREATE PARTNER** **DISCARD PARTNER** **INQUIRE PARTNER**
PIPELINE	**CREATE PIPELINE** **DISCARD PIPELINE** **INQUIRE PIPELINE** **PERFORM PIPELINE** **SET PIPELINE**
PROCESSTYPE	**CEMT INQUIRE PROCESSTYPE** **CEMT SET PROCESSTYPE** **CREATE PROCESSTYPE** **DISCARD PROCESSTYPE**
PROFILE	**CREATE PROFILE** **DISCARD PROFILE** **INQUIRE PROFILE**

Table 11. Resource identifiers and associated commands for CICS resources subject to command security checking (continued)

Resource identifier	Related CICS commands
PROGRAM	**CREATE PROGRAM** **DISCARD PROGRAM** **INQUIRE PROGRAM** **SET PROGRAM** Bundle command security applies when you use SPI commands to perform an action on a BUNDLE resource, and in that process you install, enable, disable, or discard a dynamically generated resource of this type that was defined in the CICS bundle. No CICS command security applies when you install, enable, disable, or discard a dynamically generated resource of this type through an application or platform. For more information, see Security for bundles. **SET PROGRAM REPLICATION**. SET PROGRAM REPLICATION has additional command security checking beyond SET PROGRAM. For more information, see “Resource and command check cross-reference” on page 96.
REQID	**INQUIRE REQID**
RESETTIME	**PERFORM RESETTIME**. See “CEMT considerations” on page 131.
RRMS	**INQUIRE RRMS**
SECURITY	**PERFORM SECURITY REBUILD** **PERFORM SSL REBUILD**
SESSIONS	**CREATE SESSIONS**
SHUTDOWN	**PERFORM SHUTDOWN**. Be particularly cautious when authorizing access to these and any other CICS commands that include a SHUTDOWN option.
STATISTICS	**COLLECT STATISTICS** **EXTRACT STATISTICS** **PERFORM STATISTICS RECORD** **INQUIRE STATISTICS** **SET STATISTICS**
STORAGE	**INQUIRE STORAGE**
STREAMNAME	**INQUIRE STREAMNAME**
SUBPOOL	**INQUIRE SUBPOOL**
SYSDUMPCODE	**INQUIRE SYSDUMPCODE** **SET SYSDUMPCODE** See “CEMT considerations” on page 131.
SYSTEM	**INQUIRE SYSTEM** **SET SYSTEM** **INQUIRE FEATUREKEY**
TASK	**INQUIRE TASK** **SET TASK**

Table 11. Resource identifiers and associated commands for CICS resources subject to command security checking (continued)

Resource identifier	Related CICS commands
TCLASS	**CREATE TRANCLASS** **DISCARD TRANCLASS** **INQUIRE TRANCLASS** **SET TRANCLASS** **INQUIRE TCLASS** **SET TCLASS**
TCPIP	**INQUIRE TCPIP** **SET TCPIP**
TCPIPSERVICE	**CREATE TCPIPSERVICE** **DISCARD TCPIPSERVICE** **INQUIRE TCPIPSERVICE** **SET TCPIPSERVICE**
TDQUEUE	**CREATE TDQUEUE** **DISCARD TDQUEUE** **INQUIRE TDQUEUE** **SET TDQUEUE**
TEMPSTORAGE	**INQUIRE TEMPSTORAGE** **SET TEMPSTORAGE**
TERMINAL	**INQUIRE NETNAME** **SET NETNAME** **CREATE TERMINAL** **DISCARD TERMINAL** **INQUIRE TERMINAL** **SET TERMINAL**
TRACEDEST	**INQUIRE TRACEDEST** **SET TRACEDEST**
TRACEFLAG	**INQUIRE TRACEFLAG** **SET TRACEFLAG**
TRACETYPE	**INQUIRE TRACETYPE** **SET TRACETYPE**
TRANDUMPCODE	**INQUIRE TRANDUMPCODE** **SET TRANDUMPCODE** See “CEMT considerations” on page 131.

Table 11. Resource identifiers and associated commands for CICS resources subject to command security checking (continued)

Resource identifier	Related CICS commands
TRANSACTION	**CREATE TRANSACTION** **DISCARD TRANSACTION** **INQUIRE TRANSACTION** **SET TRANSACTION** Bundle command security applies when you use SPI commands to perform an action on a BUNDLE resource, and in that process you install, enable, disable, or discard a dynamically generated resource of this type that was defined in the CICS bundle. No CICS command security applies when you install, enable, disable, or discard a dynamically generated resource of this type through an application or platform. For more information, see Security for bundles.
TSMODEL	**CREATE TSMODEL** **DISCARD TSMODEL** **INQUIRE TSMODEL**
TSPOOL	**INQUIRE TSPOOL**
TSQUEUE	**INQUIRE TSQUEUE**
TSQNAME	**INQUIRE TSQNAME** **SET TSQNAME**
TYPETERM	**CREATE TYPETERM**
UOW	**INQUIRE UOW** **SET UOW**
UOWDSNFAIL	**INQUIRE UOWDSNFAIL**
UOWENQ	**INQUIRE UOWENQ**
UOWLINK	**SET UOWLINK** **INQUIRE UOWLINK**
URIMAP	**CREATE URIMAP** **DISCARD URIMAP** **INQUIRE URIMAP** **SET URIMAP** Bundle command security applies when you use SPI commands to perform an action on a BUNDLE resource, and in that process you install, enable, disable, or discard a dynamically generated resource of this type that was defined in the CICS bundle. No CICS command security applies when you install, enable, disable, or discard a dynamically generated resource of this type through an application or platform. For more information, see Security for bundles.
VTAM	**INQUIRE VTAM** **SET VTAM**

Table 11. Resource identifiers and associated commands for CICS resources subject to command security checking (continued)	
Resource identifier	**Related CICS commands**
WEB	**INQUIRE WEB** **SET WEB**
WEBSERVICE	**CREATE WEBSERVICE** **DISCARD WEBSERVICE** **INQUIRE WEBSERVICE** **SET WEBSERVICE**
WLMHEALTH	**INQUIRE WLMHEALTH** **SET WLMHEALTH**
XMLTRANSFORM	**INQUIRE XMLTRANSFORM** **SET XMLTRANSFORM**

Resource profile examples

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 that are 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 "Resource and command check cross-reference" on page 96.

Parameters for specifying command security

In addition to the **SEC** and **SECPRFX** system initialization parameters, 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.

These parameters are summarized as follows:

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.

To view all of the attributes of this resource, see TRANSACTION resources

Security checking of transactions running under CEDF

When a transaction is run under the CEDF transaction, CICS takes into account the security settings for the target transaction only.

When a transaction runs under the CEDF transaction, CICS uses the CMDSEC attribute in the definition of the target transaction. The values of CMDSEC and RESSEC for CEDF are not considered.

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.

When CEDF is used to test a transaction, the authority of the user executing the transaction being tested is checked. For each resource accessed by the tested transaction, the user must have access authority, otherwise a NOTAUTH condition is raised. This applies to all resource checks:

- Transaction-attach
- CICS resource
- CICS command
- Non-CICS resources accessed through the QUERY SECURITY command
- Surrogate user

Note: When one of the following EXEC CICS commands are issued by a transaction running under CEDF, the password or password phrase (and new password or password phrase, where applicable) is blanked out:

CHANGE PASSWORD
CHANGE PHRASE
SIGNON
VERIFY PASSWORD
VERIFY PHRASE

CEMT considerations

In general, the resources that the CICS-supplied CEMT main terminal transaction operates on are the same as the equivalent system programming commands.

The equivalent system programming commands are 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)
```

Authorization failures in application programs

CICS returns a RESP2 value of 100 for command security failures, a value of 101 for resource security failures, and a value of 102 for surrogate security failures.

If you are running with CICS command security, CICS returns the NOTAUTH condition, RESP value 70, to your application, which is the same condition as for a resource security failure. CICS also issues message DFHXS1111 to the CICS security transient data destination CSCS. To test for this value in your application, it is recommended that you code DFHRESP(NOTAUTH) rather than explicitly coding a value.

To distinguish between a command security failure, a resource security failure, and a surrogate 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 a surrogate security failure, a value of 102 is returned in RESP2

Security checking using the QUERY SECURITY command

Use the **QUERY SECURITY** command in an application program to determine the level of access that a 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 several factors.

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

- Which resource classes are active
- Whether the transaction that is 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**

The SEC system initialization parameter

The SEC system initialization parameter specifies what level of external security you want CICS to use.

Table 12 on page 132 assumes that the relevant resource class is active; for example, that XFCT=YES is specified when QUERY SECURITY RESTYPE('FILE') is issued.

Table 12. 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

The 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 that is 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 12 on page 132 shows how the **QUERY SECURITY RESTYPE** command works if the system initialization parameter for the relevant resource class (for example, XFCT) is active. If, however, the relevant X*name* parameter is **not** active (for example, if XFCT=NO was specified), the resource is READABLE, UPDATABLE, CTRLABLE and ALTERABLE.

Transaction routing

When the **QUERY SECURITY** command without the USERID option is issued from a transaction that was 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.

However, when the **QUERY SECURITY** command is issued with the USERID option, CICS performs a surrogate user check, which might include the link user and the terminal user. In this case, the link user and the terminal user must have surrogate user authority to the user ID specified in the USERID option. After a successful surrogate user check, CICS only checks whether the user ID specified in the USERID option has access to the resource.

For more information, see IPIC link security, Link security with LU6.2, or Link security with MRO, depending upon the environment you are using.

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.

RESTYPE can take any of the values shown in Table 13 on page 133.

Table 13. QUERY SECURITY RESTYPE values

RESTYPE value	**X*name* parameter**
ATOMSERVICE	XRES
BUNDLE	XRES
DB2ENTRY	XDB2
DOCTEMPLATE	XRES
EPADAPTER	XRES
EPADAPTERSET	XRES
EVENTBINDING	XRES
FILE	XFCT
JOURNALNAME	XJCT
JOURNALNUM 1	XJCT
JVMSERVER	XRES
PROGRAM	XPPT
PSB	XPSB

Table 13. QUERY SECURITY RESTYPE values (continued)	
RESTYPE value	**X*name* parameter**
SPCOMMAND (a resource type that you can use to specify a RESID for a command.)	XCMD
TDQUEUE	XDCT
TRANSACTION	XPCT
TRANSATTACH	XTRAN
TSQUEUE	XTST
TSQNAME	XTST
XMLTRANSFORM	XRES

1. Supported for compatibility with earlier releases.
2. SPCOMMAND is a resource type that you can use to specify a RESID for a command.

The **XHFS** system initialization parameter controls resource security for z/OS UNIX files. This parameter does not have a corresponding RESTYPE value on the **QUERY SECURITY** command. 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
- ATOMSERVICE
- AUTINSTMODEL
- AUTOINSTALL
- BRFACILITY
- BUNDLE
- BUNDLEPART
- CAPDATAPRED
- CAPINFOSRCE
- CAPOPTPRED
- CAPTURESPEC
- CFDTPOOL
- CONNECTION
- CSD
- DB2CONN
- DB2ENTRY

- DB2TRAN
- DISPATCHER
- DOCTEMPLATE
- DSNAME
- DUMP
- DUMPDS
- ENQUEUE
- EPADAPTER
- EPADAPTERSET
- EPADAPTINSET
- EVENTBINDING
- EVENTPROCESS
- EXCI
- EXITPROGRAM
- FEPIRESOURCE
- FILE
- HOST
- IPCONN
- IRC
- JOURNALMODEL
- JOURNALNAME
- JVMSERVER
- LIBRARY
- MODENAME
- MONITOR
- MQCONN
- MQMON
- MVSTCB
- NODEJSAPP
- OSGIBUNDLE
- OSGISERVICE
- PARTNER
- PIPELINE
- PROCESS
- PROFILE
- PROGRAM
- REQID
- REQUEST
- RESETTIME
- RRMS
- SECURITY
- SHUTDOWN
- STATISTICS

- STORAGE
- SUBPOOL
- SYSDUMPCODE
- SYSTEM
- TASK
- TCLASS
- TCPIP
- TCPIPSERVICE
- TDQUEUE
- TEMPSTORAGE
- TERMINAL
- TRACEDEST
- TRACEFLAG
- TRACETYPE
- TRANDUMPCODE
- TRANSACTION
- TSQUEUE
- TSMODEL
- TSPOOL
- TYPETERM
- UOW
- UOWDSNFAIL
- UOWENQ
- UOWLINK
- URIMAP
- VOLUME
- VTAM
- WEB
- WEBSERVICE
- XMLTRANSFORM

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 20. 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 DB2ENTRY resources” on page 23 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 139 for details.

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:

1. Add entries to the RACF class descriptor table (CDT) and to the RACF router table.
2. Activate the new classes, define your resources in the new classes, and grant your users access to the resources.

Note: To improve the performance of **QUERY SECURITY**, 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.

Procedure

1. 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

   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
```

Note: By default, uppercase profile names are used for the class specified by the CLASS operand. If you want to allow mixed-case profile names and preserve the case of profile names for the specified class, specify CASE=ASIS when you code the ICHERCDE macro.

2. 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 z/OS Security Server RACF Macros and Interfaces.

3. When you have re-created 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
```

Note: By default, uppercase profile names are used for the class specified by the CLASS operand. If you want to allow mixed-case profile names and preserve the case of profile names for the specified class, specify CASE=ASIS when you code the ICHERCDE macro.

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 134. 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. When logging is in effect, if 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 can 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-value data areas (CVDAs).

QUERY SECURITY logging options

Specify one of the following options to control logging:

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

Examples: Using the QUERY SECURITY command for resource security checking

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. Typically, you can use **QUERY SECURITY** to check whether a user is authorized to use a particular transaction before displaying the transaction code in a menu. You can also use **QUERY SECURITY** to control access to data at the record or field level.

Example: 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 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 this example—are bypassed.

Example: Checking which transactions to offer a user

You can use **QUERY SECURITY** 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: Controlling access to data at the record or field level

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 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.

Security for CICS transactions

The RACF profile definitions for the transactions that are supplied with CICS are described in three categories. Each transaction is identified within a category that describes its use within CICS. Each category specifies the recommended security specifications, in terms of both the CICS transaction definitions and the corresponding RACF profiles. 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). This initialization does not include the CICS sample transactions (the transactions that are in groups that start with DFH$). Some category 1 transactions are not in the CSD. They are defined by CICS during installation.

See “Implementing CICSPlex SM security” on page 163 for details of transactions that are related to CICSPlex SM.

The three categories are:

Category 1 transactions
: Transactions that are for CICS internal use only and must not run from a user terminal. See “Category 1 transactions” on page 143 for details.

Category 2 transactions
: Transactions that must be restricted to specific signed-on users; for example, you might want to limit access to transactions that define and install CICS resources. See “Category 2 transactions” on page 143 for details.

Category 3 transactions
: Transactions that are available to all users, whether signed-on or not. These transactions are not subject to security checking. Any security definitions for these transactions are redundant. See “Category 3 transactions” on page 148 for details.

By default, all CICS transactions (except for category 3 transactions) are subject to RACF protection, unless you run your CICS regions with transaction security switched off. You can switch off transaction security 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.

If you run with transaction security (SEC=YES and XTRAN=YES), CICS issues a security check for each transaction attach, other than a transaction within category 3. This check establishes whether the user is permitted to run that transaction.

Category 1 transactions

Category 1 transactions are never associated with a terminal. They are for CICS internal use only, and must not be started from a user terminal. Do not modify the CSD definitions of these transactions. For a list of category 1 transactions, see List of CICS transactions.

CICS checks that the region user ID has the authority to attach these transactions. If the region user ID is not authorized to access any of the category 1 transactions, CICS issues message DFHXS1113 for each unauthorized category 1 transaction and fails to initialize.

Customize and run the sample CLIST DFH$CAT1 to create the RACF definitions. You need to provide an owner of the profile, the CLASSNAME of the transaction, and the list of CICS region user IDs.

The sample CLIST is in library CICSTS56.CICS.SDFHSAMP.

Category 2 transactions

Category 2 transactions are initiated by CICS users, or are associated with CICS users. Some are associated with signed-on users. Others are associated with task user IDs that are associated with adapters. Restrict authorization to initiate these transactions to user IDs that belong to specific RACF groups. For a list of category 2 transactions, see List of CICS transactions.

The supplied transactions are defined with the recommended RESSEC and CMDSEC options. Make sure that the user groups that are authorized to use the transactions are also authorized to access the CICS resources and commands that these transactions use. You can change these definitions but, if you do so, copy them to another group.

For most category 2 transactions, specify them to RACF as follows:

- UACC(NONE) and AUDIT(FAILURES) in the transaction profile. AUDIT(FAILURES) is the default and does not have to be specified.
- Access list.

If you protect a resource with a resource group profile, be careful if you protect the same resource with another profile and a user can have different access to both profiles. If the profiles are different (for example, if they have different access lists), RACF merges the profiles that are used during authorization checking. The merging might incur higher cost in checking, and it can be difficult to determine exactly which access authority applies to a particular user. For more information, see z/OS Security Server RACF Security Administrator's Guide.

It is unlikely that users require access to all the transactions in this category, so consider defining the transactions in several subcategories. You can choose to group CICS transactions in the way that best suits the needs of your installation. The sample CLIST DFH$CAT2 (in library CICSTS56.CICS.SDFHSAMP) shows one way that you might group the category 2 transactions. To use a different setup, adapt the CLIST, or provide your own.

Table 14. Suggested subcategories for Category 2 transactions.. To see which transactions are in each subcategory, see List of CICS transactions or the DFH$CAT2 CLIST.

Subcategory	Contains	Notes
For all users:		

Table 14. Suggested subcategories for Category 2 transactions.. To see which transactions are in each subcategory, see List of CICS transactions or the DFH$CAT2 CLIST. *(continued)*

Subcategory	Contains	Notes
ALLUSER	Transactions that are used by all users.	Add to the list of transactions your "good morning" transaction (defined on the **GMTRAN** system initialization parameter) and your "good night" transaction in this group (defined on the **GNTRAN** system initialization parameter).
For operators:		
SYSADM	Transactions that are used by system programmers who need full access to the system.	
INQUIRE	Transactions that are used by operators or other users who need only to inquire on resources.	
OPERATOR	Transactions that are used by operators.	
DBCTL	Transactions that are used by operators of the DBCTL interface to IMS.	
CMCIUSER	Transactions that are used by operators who use the CICS Explorer® and other users of the CMCI.	
For application developers:		
DEVELOPER	Transactions that are used by application developers on a non-Production system.	
For application users:		

Table 14. Suggested subcategories for Category 2 transactions.. To see which transactions are in each subcategory, see List of CICS transactions or the DFH$CAT2 CLIST. *(continued)*

Subcategory	Contains	Notes
JVMUSER	Transactions that are used by users of Liberty applications.	Depending on your security configuration, transactions CJSA and CJSU might require the CICS default user to have access. For more information, see Security for Java applications, and Configuring permissions for z/OS Connect Services and APIs. As a security best practice, turn on Liberty security and avoid using the CICS default user to run application tasks. For the z/OS Connect Service, you can install a URIMAP resource that CICS uses to associate the work for the Service with a specific transaction ID in CICS and with an initial user ID. CJSA is the default transaction ID for any web request that does not have a matching URI. Consider restricting access to CJSA to prevent any arbitrary application being run.

Table 14. Suggested subcategories for Category 2 transactions.. To see which transactions are in each subcategory, see List of CICS transactions or the DFH$CAT2 CLIST. *(continued)*

Subcategory	Contains	Notes
INTERCOM	Transactions that are used by application users who run transactions on multiple regions.	If you are using function shipping, the mirror transactions must be available to remote users in a function shipping environment. When a database or file is on another CICS region, CICS function ships the request to access the data. The request runs under one of the CICS-supplied mirror transactions. In this situation, the following conditions apply: • The terminal user that runs the application must be authorized to use the mirror transaction. See “Transaction security” on page 69. • The terminal user must also be authorized to use the data that the mirror transaction accesses. See “Resource security” on page 73. 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. For a deferred **START** request, if the user transaction to be started is eligible for dynamic routing, system transaction CDFS will run and start the user transaction at the specified time. Ensure that security for CDFS is correctly configured.

Table 14. Suggested subcategories for Category 2 transactions.. To see which transactions are in each subcategory, see List of CICS transactions or the DFH$CAT2 CLIST. *(continued)*

Subcategory	Contains	Notes
WEBUSER	Transactions that are used by application users who run transactions using the CICS web interface.	The CICS default user requires access to the CWBA transaction initially, even if an analyzer program is then used to assign another user ID to the task. Make sure 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.
RPCUSER	Transactions that are used by application users who run transactions using ONC RPC.	
PIPEUSER	Transactions that are used by application users who run web services transactions.	
EVENTUSER	Transactions that are used by EP adapters.	If the RESSEC and CMDSEC options for these transactions are not the ones you want, you can specify your own transaction IDs in the adapter tab **Advanced Options** section of the Event binding editor. For more information, see Specifying EP adapter and dispatcher information.
For users of IBM MQ:		
MQADMIN	CKAM CKCN CKDL CKRS CKSD CKSQ	
MQBRIDGE	CKBC CKBP CKBR	
MQMONITOR	CKTI	
MQSTATUS	CKQC CKBM CKRT CKDP	

Category 3 transactions

Category 3 transactions are either initiated by the terminal user or associated with a terminal. For a list of category 3 transactions, see List of CICS transactions.

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.

These transactions can be defined to RACF. Although this definition does not affect task attach-time processing, it is needed to support the QUERY SECURITY command.

Security for submitting a JCL job to the internal reader

For CICS to be able to secure who can submit JCL, there are a number of configuration and security definitions required.

JCL jobs can be submitted by writing JCL in two ways:

- Using **WRITEQ TD** commands to an extrapartition TDQ defined to the internal reader
- Using **SPOOLWRITE** commands when a `USERID("INTRDR")` was specified on the **SPOOLOPEN** command

There are three user IDs involved in job submission:

- Region user ID.
- Signed-on user ID. This is the user ID that the task is running under (the default user ID if not logged on).
- Job user ID. This is specified by a USER parameter on the JOB card. If the JOB statement doesn't contain a USER parameter, the default will depend on the security settings. If this isn't the region user ID, CICS will add USER=*job_userid* to the JOB card. In releases earlier than CICS TS 5.5, the default is always the region user ID.

Protection for JCL jobs that are submitted through the TDQ is provided by resource security on the TDQ. Additional protection is provided by surrogate user checking if the USER parameter is specified on the JOB card. For details, see "Security when using a TDQ to submit JCL" on page 148.

Protection for JCL jobs that are submitted by using spool commands is provided by surrogate user checking. For details, see "Security when using the SPOOLWRITE commands to submit JCL" on page 149.

In addition, it is necessary to have a profile for the region user ID in the JESSPOOL class to give the region user ID authority to submit jobs for the job user IDs. See "z/OS surrogate user checking" on page 149 for details of the RACF definitions and error messages if the surrogate check fails.

Security when using a TDQ to submit JCL

Protection is provided by a resource security check of the TDQ. For details, see "Security for transient data" on page 80.

Additional security configuration and checking depends on whether a user ID is specified on the job card written to the TDQ by using the **WRITEQ TD** command.

If a user ID is specified

Example:

```
//JOBNAME JOB USER=JOBUSER
```

CICS performs an additional surrogate check when writing the job card to the TDQ. This surrogate check verifies whether the task user ID has permission to submit jobs on behalf of the job user ID (JOBUSER in the above example).

To enable this security check, you must set the following options:

- CICS surrogate user checking is enabled by the system initialization parameter **XUSER=YES**.
- The feature toggle for surrogate user checking for spool commands is enabled:

```
com.ibm.cics.spool.surrogate.check=true
```

If the surrogate check fails, a NOTAUTH response will be returned from the **WRITEQ TD** command. See “CICS surrogate user checking” on page 150 for details of the RACF definitions and error messages if the surrogate check fails.

If a user ID is not specified

Example:

```
//JOBNAME JOB
```

CICS will add a USER parameter to the statement written to the TDQ. The job user ID added will be set to the value specified by the JOBUSERID option in the TDQ definition.

```
//JOBNAME JOB USER=jobuserid
```

If JOBUSERID is not defined in the TDQ definition, the job user ID will be set to the CICS region user ID. No additional surrogate checking will be done by CICS.

```
//JOBNAME JOB USER=regionuserid
```

Security when using the SPOOLWRITE commands to submit JCL

To use spool commands, CICS must be started with the system initialization parameter **SPOOL=YES**.

If a JOB card that is written using a **SPOOLWRITE** command has a USER parameter, protection is provided by a surrogate check will be made if the following options are defined:

- CICS surrogate user checking is enabled by the system initialization parameter **XUSER=YES**.
- The feature toggle for surrogate user checking for spool commands is enabled:

```
com.ibm.cics.spool.surrogate.check=true
```

If the surrogate check fails, a NOTAUTH response will be returned from the **SPOOLWRITE** command. See “CICS surrogate user checking” on page 150 for details of the RACF definitions and error messages if the surrogate check fails.

If a JOB card that is written using a **SPOOLWRITE** command doesn't have a USER parameter, the job user ID will default to the region user ID. This will be subject to a surrogate check. You can change the default to job user ID to be the signed-on user ID if you set the following feature toggle:

```
com.ibm.cics.spool.defaultjobuser=TASK
```

Specifying the region user ID on the JOB card

If you want a job to be able to run under the region user ID without needing to change the JCL dynamically, you can specify USER=&SYSUID on the JOB card. This will run the job under whatever region user ID the job was submitted from. This applies to JCL jobs written using the **SPOOLWRITE** command or using a TDQ. A surrogate security check, if applicable, will be made when jobs are submitted using &SYSUID to represent the region user ID.

z/OS surrogate user checking

Although the **SPOOLWRITE** and **WRITEQ TD** commands are issued by the signed-on user ID, the job is submitted by the region user ID. Therefore, if the job user ID is not the region user ID and no password is

supplied, it is necessary to grant the region user ID surrogate authority to submit jobs on behalf of the job user ID. This is required regardless of whether CICS surrogate user checking is active or not.

```
RDEFINE SURROGAT job_userid.SUBMIT UACC(NONE) OWNER(sysadmin)
PERMIT job_userid.SUBMIT CLASS(SURROGAT) ID(region_userid) ACCESS(READ)
```

If you configure your region user IDs to be able to submit jobs on behalf of any user, it is recommended that the region user IDs are not used for any other purpose than a user ID for running CICS.

The check takes place after the job is submitted. If the check fails, the job will fail with an ICH408I message issued to the console and job log, but no response will be returned to the application.

```
ICH408I USER(job_userid) GROUP(group) NAME(username)
SUBMITTER(region_userid)
LOGON/JOB INITIATION - SUBMITTER IS NOT AUTHORIZED BY USER
```

CICS surrogate user checking

CICS surrogate user checking will be made if the following options are defined:

- CICS surrogate user checking is enabled by the system initialization parameter **XUSER=YES**.
- The feature toggle for surrogate user checking for spool commands is enabled:

```
com.ibm.cics.spool.surrogate.check=true
```

If the job user ID is not the signed-on user ID and no password is supplied, it is necessary to grant the signed-on user ID surrogate authority to submit jobs on behalf of the job user ID.

```
RDEFINE SURROGAT job_userid.SUBMIT UACC(NONE) OWNER(sysadmin)
PERMIT job_userid.SUBMIT CLASS(SURROGAT) ID(signed_on_userid) ACCESS(READ)
```

If a surrogate check is applicable it will take place when the **SPOOLWRITE** or **WRITEQ TD** command writes the JOB card. If the check fails, the command will fail with a NOTAUTH response. In addition, a DFHXS1111 message will be issued to the CICS log, and an ICH408I message will be issued to the console and job log.

```
ICH408I USER(job_userid) GROUP(group) NAME(username) SUBMITTER(signed_on_userid)
LOGON/JOB INITIATION - SUBMITTER IS NOT AUTHORIZED BY USER
```

Learn more

If you want to migrate to using CICS surrogate user checking, follow the instructions in Upgrading security.

When and how CICS determines the user ID of the CICS region

CICS makes use of the user ID of the region in which it runs for a variety of purposes.

CICS uses the region user ID in the following situations:

- 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 46.
- As the user to be checked for category 1 transactions.
- 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 “Implementing MRO security” on page 268.

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.

Chapter 3. Identity propagation and distributed security

Identity propagation provides a mechanism to allow a user identity from an external security realm to be preserved, regardless of where the identity information was created, strengthening accountability across distributed environments.

In an external computing environment, for example, WebSphere® Application Server, the identity of a user is authenticated using a user identification that applies to that environment. Applications like WebSphere Application Server often use a separate, shared external security manager user ID when communicating with a CICS system. The original identity of the user is not passed to CICS and therefore cannot be passed onto the external security manager, making it difficult to determine the initial user identity and impacting the audit trail of the request.

In the identity propagation topics, RACF represents the external security manager. If you are not using RACF with CICS, consult your system administrator to configure the settings for your external security manager.

The term *distributed identity* represents user identity information, for example, an X.500 distinguished name and associated LDAP realm, that originates from a remote system. The distributed identity is created in one system and is passed to one or more other systems over a network. A distributed identity originates outside of CICS only; CICS is never the source of a distributed identity, but is capable of propagating the distributed identity onwards.

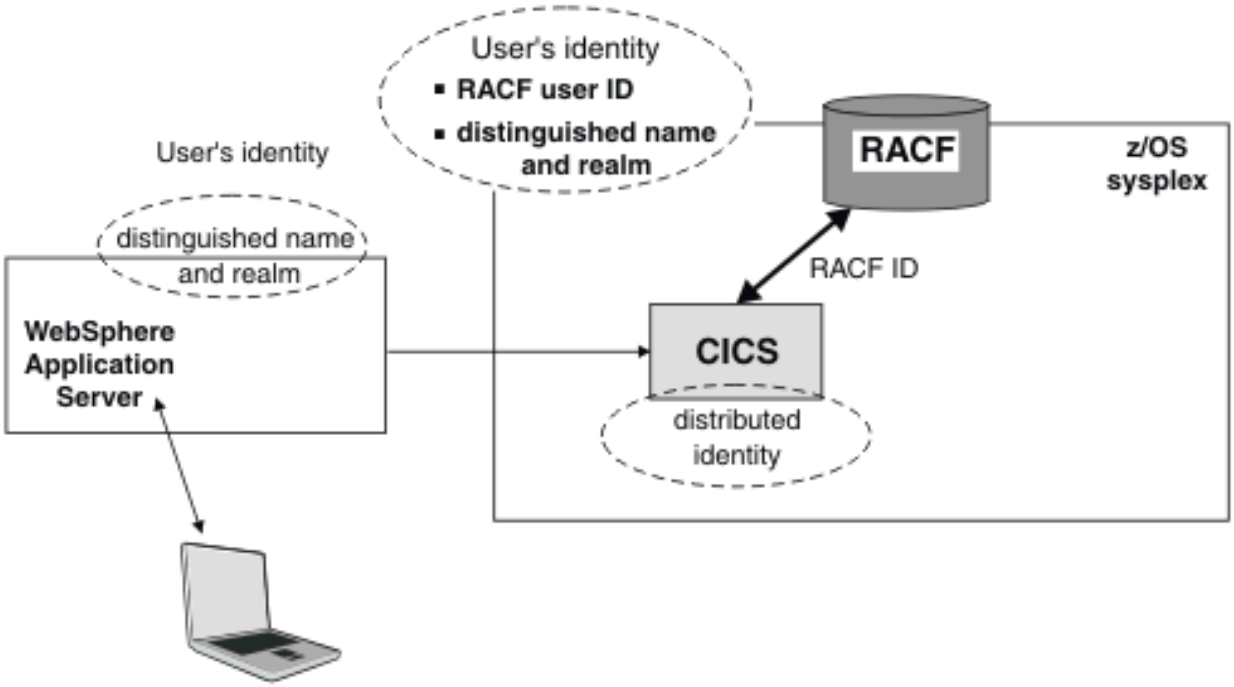
The following does not apply to use of identity propagation CICS Liberty. For information on identity propagation mapping in Liberty, see Configuring security for a Liberty JVM server by using distributed identity mapping.

When a distributed identity enters the sysplex over MRO and IPIC connections, it is automatically propagated in the sysplex, regardless of connection settings. CICS security handles the distributed identity as additional information relating to the user ID, and a distributed identity cannot exist without a user ID.

Outside the sysplex, the distributed identity can be propagated on, depending on support. Receipt of a distributed identity by another party depends on whether the party is participating in identity propagation. See “Support and requirements for identity propagation” on page 154.

Identity propagation allows the distributed identity to be preserved, regardless of where the identity information was created, for use during authorization and for auditing purposes. The distributed identity is part of the identity context that is carried with a request from the external client application to the server, and it is incorporated in the access control of the server as part of the authorization process; for example, CICS Transaction Gateway on behalf of WebSphere Application Server.

The diagram shows how the X.500 distinguished name and associated LDAP realm which identify the user externally are passed with the request from WebSphere Application Server to a CICS system. The distinguished name and realm, which are known in CICS as a distributed identity when they are transmitted across a network, are propagated into the z/OS security context and are associated with the RACF user ID. With the z/OS RACF command, **RACMAP**, you can use mapping filters to correlate the distinguished name and realm to a RACF user ID, preserve the distributed credential information and fulfill governance and auditing requirements. RACF provides information to CICS about the distinguished name and realm, allowing retrieval in CICS of the identity of the initial user.



Sample network topologies for using identity propagation expands on this diagram and shows how different network and product combinations provide support for distributed identities.

Support and requirements for identity propagation

CICS provides support for identity propagation between a range of products. Make sure that you meet these requirements to allow participation in identity propagation.

Summary of identity propagation participation

Identity propagation is a form of asserted identity and as a result, participation requires a number of factors:

- All parties involved must be able to process distributed identities
- A trusted connection must connect all external parties

If a system does not participate in identity propagation, distributed identity information is ignored, and user ID information is used as before.

For propagation, the outbound request must be made from a participating task, that is; a task whose user ID has an associated distributed identity.

Support for identity propagation

CICS provides support for identity propagation in the following situations:

- Inbound requests to CICS from WebSphere Application Server using the CICS ECI resource adapters over an IPIC connection.
- Using a WS-Security Header element in a web service request. Routed inbound web service requests do not support identity propagation.
- Using IPIC and MRO connections between CICS systems. The distributed identity is used by CICS only if it is passed to the MRO or IPIC connection from a participating task.
- Transactions issuing local or function shipped **START** commands. Exceptions to this support are the following situations, where the distributed identity is not propagated:

  - If a **START** command specifies a USERID or TERMID.
  - If a **START** command is shipped to a remote region across a LU61 or LU62 connection.
  - If a dynamically routed **START** command is delayed.
- Using Liberty. For information on identity propagation mapping in Liberty, see Configuring security for a Liberty JVM server by using distributed identity mapping.

Sample network topologies for using identity propagation provides diagrams and examples to explain how user security information is passed using the supported requests and connections.

RACF requirements for identity propagation

You must configure your RACF settings for the RACMAP and SETR RACLIST(IDIDMAP) commands before you update clients and CICS configuration definitions for identity propagation.

CICS requirements for identity propagation

CICS has a number of requirements to allow distributed identities to flow:

- Security must be enabled, by specifying the SEC=YES system initialization parameter.
- The external security manager, for example, RACF, must be configured to accept distributed identities.
- All partner systems must be able to process distributed identities.
- CICS supports distinguished names up to 246 bytes in length, and realm names up to 252 bytes in length.
- IPIC connections are limited to supporting identity contexts (ICRX identity tokens), the total size of which must not exceed 2000 bytes.

Sample network topologies for using identity propagation

Identity propagation is supported on network topologies that use either IPIC connections or web service requests to CICS. Use these sample topologies to help you plan your configuration.

Sample configuration using an IPIC connection and CICS Transaction Gateway

If you are using IPIC connections into CICS, you might be using CICS Transaction Gateway as an interface between WebSphere Application Server and the CICS ECI resource adapters. If CICS Transaction Gateway and any communicating CICS systems are not on the same sysplex, SSL is also required. The following sample topology shows how the X.500 distinguished name and associated LDAP realm are passed with the request from WebSphere Application Server through CICS Transaction Gateway over an IPIC connection to a CICS system on the same sysplex. The distinguished name and realm, which are known in CICS as a distributed identity when they are transmitted across a network, are then passed between CICS systems using either MRO or IPIC in a sysplex or between CICS systems in different sysplexes using IPIC over SSL. The multiple CICS systems shown in this scenario show how you can connect CICS systems in and outside of a sysplex, however multiple CICS systems are not a required part of the configuration to allow identity propagation. The distributed identity is propagated into the z/OS security context, also known as the Accessor Environment Element (ACEE), and is associated with the RACF user ID using the mapping rules specified in the RACF **RACMAP** command. RACF provides information to CICS about the distinguished name and realm as well as the RACF user ID, allowing retrieval in CICS of the identity of the initial user.

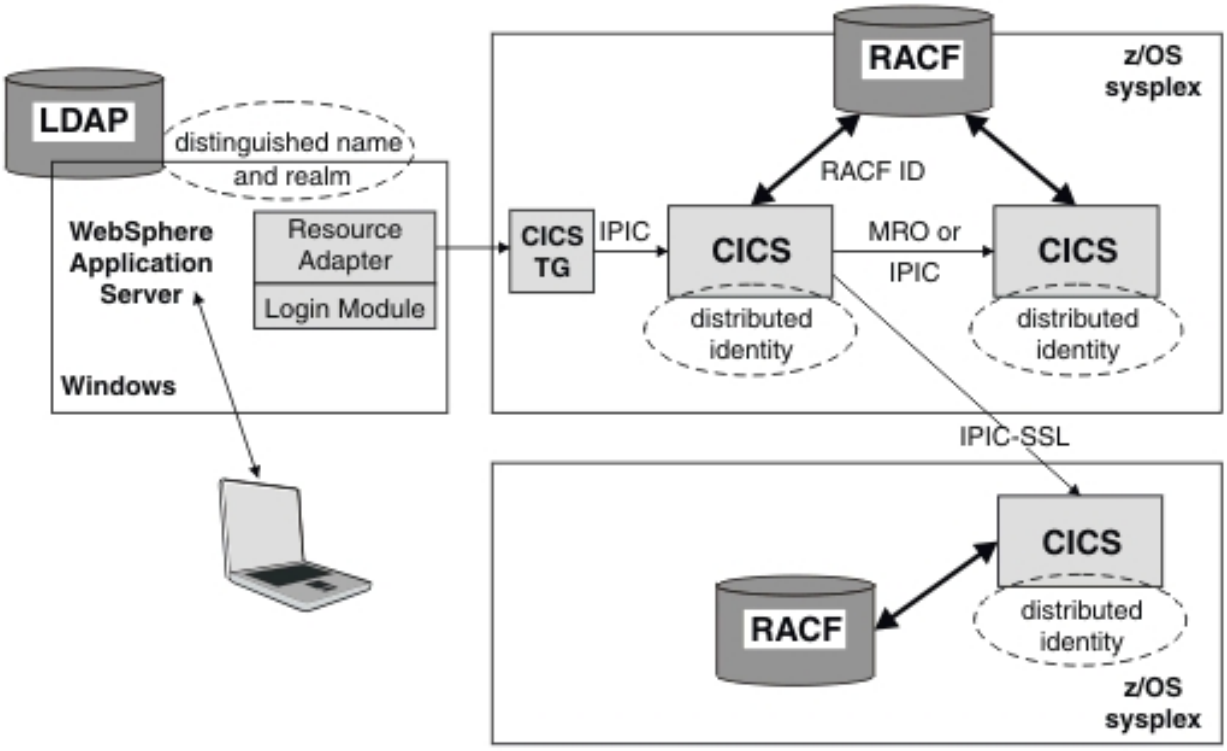


Figure 3. Example of identity propagation using WebSphere Application Server and CICS Transaction Gateway, communicating with CICS over an IPIC connection.

For more information about CICS Transaction Gateway and identity propagation, see CICS Transaction Gateway for z/OS or CICS Transaction Gateway for Multiplatforms.

Sample configuration using a web service request and IBM DataPower

If you are sending web service requests to CICS, you might be using IBM DataPower® as an interface between WebSphere Application Server and CICS. You can use the IBM DataPower appliance with CICS web services WS-Security support to process the XML digital signature and perform mappings to a predefined RACF user ID.

The following sample topology shows how the X.500 distinguished name and associated LDAP realm are passed with the request from WebSphere Application Server through IBM DataPower. The distinguished name and realm are sent in the Extended Identity Context Reference WS-Security Header element of a web service request to a CICS system. For more information about ICRX identity tokens, see z/OS Security Server RACF Data Areas. The distinguished name and realm, which are known in CICS as a distributed identity when they are transmitted across a network, are then passed between CICS systems using either MRO or IPIC in a sysplex or between CICS systems in different sysplexes using IPIC over SSL. The multiple CICS systems shown in this scenario show how you can connect CICS systems in and outside of a sysplex, however multiple CICS systems are not a required part of the configuration to allow identity propagation. The distributed identity is propagated into the z/OS security context, also known as the Accessor Environment Element (ACEE), and is associated with the RACF user ID using the mapping rules specified in the RACF **RACMAP** command. RACF provides information to CICS about the distinguished name and realm as well as the RACF user ID, allowing retrieval in CICS of the identity of the initial user.

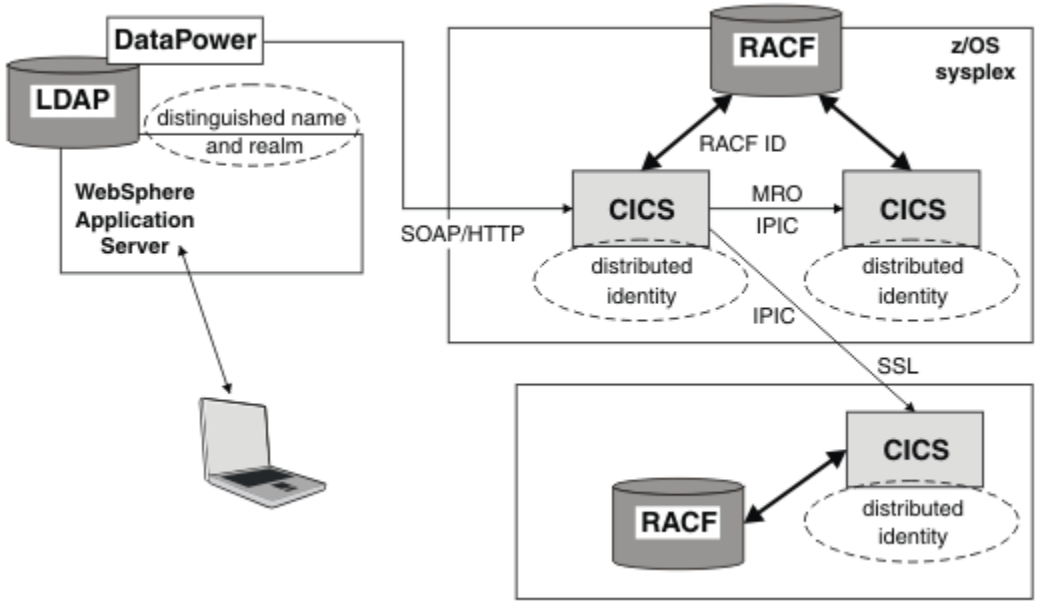


Figure 4. Example of identity propagation using WebSphere Application Server and IBM DataPower, communicating with CICS using a web service request.

Configuring identity propagation

You must configure RACF to be able to map distributed identities. You can configure your CICS system to accept distributed identity information from IPIC connections or from web service requests. Your CICS Liberty server can also use distributed identity mapping.

If you are using MRO connections, you can accept distributed identity information between CICS systems only. You must ensure that ATTACHSEC(IDENTIFY) is specified, but no other configuration changes are required.

Configuring RACF for identity propagation

Use the RACF **RACMAP** command to create, delete, and list a distributed identity filter. Use the RACF **SETR RACLIST(IDIDMAP)** command to refresh the IDIDMAP resource profile that contains the distributed identity filter. You must configure your RACF settings for identity propagation before you update clients and CICS configuration definitions.

Before you begin

Ensure that you have the required access to update RACF profiles for your z/OS system.

About this task

RACF uses the term *distributed identity filter* to describe a mapping association between a RACF user ID and one or more distributed identities. When you define the filter using the **RACMAP** command, you associate (or map) a distributed identity with a RACF user ID. If the distributed identity is not mapped in the **RACMAP** command, the default CICS user ID is not used and a security error is issued. Each distributed identity filter is stored in a general resource profile in the IDIDMAP class.

The owner of an IDIDMAP profile is the user ID of the **RACMAP MAP** command issuer.

When the **SETR RACLIST(IDIDMAP)** command is issued, the IDIDMAP class becomes active. When a distributed user is authenticated and the supported transaction is sent to the z/OS system, RACF receives the distinguished name and realm of the user as character strings of UTF-8 data. RACF has a number of

restrictions as a result of UTF-8 data encoding; for example, if the distinguished name exceeds 246 bytes or the realm exceeds 252 bytes, the **RACMAP MAP** command fails. See the topic about restrictions for UTF-8 data values in the z/OS Security Server RACF Security Administrator's Guide.

RACF uses the distinguished name and realm to search IDIDMAP profiles for the distributed identity filter that contains the name values best matching the data. When the best matching filter is found, RACF assigns a RACF user ID.

Procedure

1. Create a generic profile called `IRR.IDIDMAP.*` in the FACILITY class in RACF.
2. Grant UPDATE access to your user group to enable general access to this profile.

   Here is an example, using `dev` as the name of the user group:

   ```
   RDEFINE FACILITY IRR.IDIDMAP.** OWNER(cpssing)
   PERMIT IRR.IDIDMAP.** CLASS(FACILITY) ID(dev) ACCESS(UPDATE)
   SETROPTS RACLIST(FACILITY) REFRESH
   ```

3. You can grant specific users authority to manage the IDIDMAP class by granting CLAUTH access.

   Here is an example, using `usera`, `userb`, and `userc` as the users that are granted access:

   ```
   ALTUSER (usera,userb,userc) CLAUTH(IDIDMAP)
   ```

4. Issue the following command to activate the changes:

   ```
   SETROPTS CLASSACT(IDIDMAP) RACLIST(IDIDMAP)
   ```

5. If you are changing RACMAP definitions on a running system, the cached copy might not map to the same user ID as the new RACMAP definition. To reset the mapping you must wait until the cached copy is deleted. The copy is deleted if the distinguished name and realm pair (the distributed user ID) is not used for the length of time that is specified in the USRDELAY SIT parameter.

   For performance reasons, CICS caches distinguished names and realms so that the first requests only from a distinguished name and realm require RACF to build an ACEE using the RACMAP definitions.

Results

The distributed user information is mapped in RACF. RACF is configured to map the distributed identity to a RACF user ID.

What to do next

You are now ready to configure your IPIC connections, web service connections or CICS Liberty server to CICS. For more information about the RACF **RACMAP** and **SETR RACLIST(IDIDMAP)** commands, see the information about distributed identity filters in the z/OS Security Server RACF Security Administrator's Guide.

Chapter 4. Invoking an external security manager

CICS provides an interface to an external security manager (ESM), which can be the Resource Access Control Facility (RACF), a vendor product, or user-written. CICS security uses the MVS system authorization facility (SAF) interface to route authorization requests to the external security manager (ESM).

Typically, 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-written or vendor-supplied ESM. If you want to use your own security manager, you must supply an MVS router exit routine.

Note: This information is intended primarily for non-RACF users. For definitive information about security processing using RACF, refer to RACF facilities.

Overview of 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, for example 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 might 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. For more information, see Passing control to a user-supplied ESM.

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 z/OS Security Server RACF Messages and Codes.

For more information about the MVS router, see System authorization facility (SAF) in z/OS Security Server RACROUTE Macro Reference and ICHRTX00 - MVS Router Exit in z/OS MVS Installation Exits.

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 whether or not your ESM is RACF.

If you use an ESM exit program that issues a **RACROUTE REQUEST=AUTH** and the exit is running because of a security request from CICS, you must ensure that any CLASS which is referenced is listed in the RACLIST.

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 15 on page 160.

Table 15. 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	ICHPWX01	ICHPWXP	PWXINSTL
	ICHPWX11	ICHPWX2	PWX2INST
	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 by 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 RACF installation exits.

Guidelines on using RACF exits for CICS RACROUTE calls

CICS runs many RACROUTE requests, in particular those that create or delete an RACF access control environment element (ACEE) such as SIGNON, CHANGE PASSWORD, and CHANGE PHRASE commands on the RO TCB. Requests run serially on the RO TCB. Therefore, the exits should not run anything with a large elapse time; otherwise, other security requests or usage of the TCB would be blocked.

The installation data parameter list

The installation data parameter list gives your external security manager (ESM) exit programs access to the CICS security event that is being processed and details about the current CICS environment.

Details of the current CICS environment might include, as available:

- APPLID of the CICS region
- Common work area
- Transaction being invoked
- Program being run
- CICS terminal identifier
- SNA LUname
- Terminal user area
- An 8-byte communication area

For the format of the data parameter list, see Mapping the installation data parameter list.

Chapter 5. Security for CICSPlex SM

This part describes how to implement security for CICSPlex SM.

Implementing CICSPlex SM security

To implement CICSPlex SM security using RACF, you must determine who needs access to the various CICSPlex SM functions. You must also perform a number of tasks to define CICSPlex SM class names and resource names, as well as activating security and refreshing RACF profiles.

About this task

If you are using a SAF-compliant external security manager (ESM) other than RACF, refer to "Invoking a user-supplied external security manager" on page 212. The following steps summarize how to set up security for CICSPlex SM. Each of the steps are explained in detail in the subsequent topics.

Procedure

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 Controlling access to Web User Interface resources for more information.
9. Optional: 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.

About this task

You can control access to CICSPlex SM resources in two ways:

- By restricting access to the objects managed by CICSPlex SM views. This restriction 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 restriction does not affect access to the objects that are being managed but prevents access to the view sets, menus, and View Editor themselves.

Procedure

1. Answer the following questions to determine who requires access to the CICSPlex SM resources:

   What groups of users use CICSPlex SM?
   Your enterprise probably already has several user groups that are 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 required for your enterprise.)

Which CICSPlex SM views do each group need to use?
CICSPlex SM manages CICS resources that use views. Views are grouped by function: 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 only use 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 you decide 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, by 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.

Tip: For application programmers, if you need to control who is allowed to use the CPSM option on the CICS translator, you can use RACF to control who is allowed to load the DFHSMTAB table at translation time. For a description of RACF program control, see the z/OS Security Server RACF Security Administrator's Guide. DFHSMTAB is the language definition table that defines the CICSPlex SM API commands. It is loaded only on demand.

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 16. 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					

Table 16. Security matrix (continued)

RACF group → CICSPlex SM view group ↓	System Programming ID( )	Operations ID( )	Help Desk ID( )	Application Programming ID( )	Performance ID( )
Operations OPERATE					
Monitor MONITOR					
Business Application Services BAS					
CSD					

Table 17 on page 165 is a sample of a completed security matrix for a production CICSplex:

Table 17. 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	
CSD	ALTER	ALTER		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 a 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 systems programming group requires UPDATE access to all of the view groups and ALTER access to the OPERATE, BAS, and CSD views.

   To start, allow UPDATE access to all possible profiles:

```
PERMIT ** CLASS(CPSMOBJ)  ID(SYSPGRP) ACCESS(UPDATE)
```

   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(SYSPGRP) ACCESS(ALTER)
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(SYSPGRP) ACCESS(ALTER)
PERMIT BAS.** CLASS(CPSMOBJ) ID(OPSGRP) ACCESS(ALTER)
PERMIT BAS.** CLASS(CPSMOBJ) ID(APPLGRP) ACCESS(UPDATE)
```

   For the CSD views:

```
PERMIT CSD.** CLASS(CPSMOBJ) ID(SYSPGRP) ACCESS(ALTER)
PERMIT CSD.** CLASS(CPSMOBJ) ID(OPSGRP) ACCESS(ALTER)
PERMIT CSD.** CLASS(CPSMOBJ) ID(APPLGRP) ACCESS(UPDATE)
```

Results

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. Refer to “Specifying CICSPlex SM resource names in profiles” on page 176 for details.

What to do next

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. “Example tasks: security” on page 213 provides detailed profile examples.

General requirements for CICSPlex SM security

Review your RACF configurations to ensure that your systems meet the minimum requirements.

- The IDs for all users expected to use CICSPlex SM must 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 must be defined to RACF in a consistent manner in all MVS systems used by CICSPlex SM.

In addition, 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.

Procedure

1. Prohibit universal access by specifying UACC(NONE).
2. Ensure that minimum access to the data sets is authorized for the RACF USERID assigned to each of the following:
   - Every CMAS job or started task.
   - Every MAS.
   - All individuals allowed to use CICSPlex SM from the CICSPlex SM WUI and API (both system administrators and users).

   Table 18 on page 167 lists the CICSPlex SM data sets and the minimum access that should be granted to each type of user ID.

Table 18. Access by user ID for CICSPlex SM data sets

Data set name	CMAS	MAS	System Admin.	Individual User
SYS1.CICSTS56.CPSM.SEYULPA	NONE	READ	UPDATE	NONE
SYS1.CICSTS56.CPSM.SEYULINK	READ	NONE	UPDATE	NONE
CICSTS56.CPSM.SEYUAUTH	READ	READ	UPDATE	READ
CICSTS56.CPSM.SEYULOAD	READ	READ	UPDATE	NONE
CICSTS56.CPSM.SEYUPARM	READ	READ	UPDATE	NONE
CICSTS56.CPSM.SEYUCMOD	NONE	NONE	UPDATE	NONE
CICSTS56.CPSM.SEYUCOB	NONE	NONE	UPDATE	READ
CICSTS56.CPSM.SEYUC370	NONE	NONE	UPDATE	READ
CICSTS56.CPSM.SEYUDEF	READ	READ	UPDATE	READ
CICSTS56.CPSM.SEYUCLIB	NONE	NONE	UPDATE	READ
CICSTS56.CPSM.SEYUMLIB	NONE	NONE	UPDATE	READ
CICSTS56.CPSM.SEYUPLIB	NONE	NONE	UPDATE	READ
CICSTS56.CPSM.SEYUTLIB	NONE	NONE	UPDATE	READ

Table 18. Access by user ID for CICSPlex SM data sets (continued)

Data set name	CMAS	MAS	System Admin.	Individual User
CICSTS56.CICS.SDFHINST	NONE	NONE	UPDATE	NONE
CICSTS56.CPSM.SEYUMAC	NONE	NONE	UPDATE	READ
CICSTS56.CPSM.SEYUOS2	NONE	NONE	UPDATE	NONE
CICSTS56.CPSM.SEYUPL1	NONE	NONE	UPDATE	READ
CICSTS56.CPSM.SEYUPROC	NONE	NONE	UPDATE	READ
CICSTS56.CPSM.SEYUSAMP	NONE	NONE	UPDATE	READ
CICSTS56.CPSM.EYUSDEF	NONE	NONE	UPDATE	UPDATE
CICSTS56.CPSM.EYUDREP	UPDATE	NONE	UPDATE	NONE
CICSTS56.CPSM.EYUIPRM	NONE	NONE	UPDATE	NONE

What to do next

If you require more details about RACF data set protection, see the z/OS Security Server RACF Security Administrator's Guide.

Determining the agent user ID for CICSPlex SM components

You need to know the user ID that is used by components of CICSPlex SM, such as the CMAS, MAS, WUI, and SMSS so that you can set up the right authorization in RACF.

What governs the agent user ID?

The agent user ID depends on the following factors:

- The release of CICS, but irrespective of the release of CICSPlex SM
- How the agent was initialized.
- Whether the **PLTPIUSR** system initialization parameter is set (to initialize the agent at the initialization of the CICS region)

The agent user IDs for the CMAS, MAS, or WUI run under the **PLTPIUSR** user ID only when the SIT parameter **SEC=YES** *and* at least one of the following SIT parameter values are set. Otherwise, the agent user ID continues to run under the CICS region user ID.

- **PLTPI=YES** or **PLTPI=***name*
- **MQCONN=YES**
- **DB2CONN=YES**
- **DBCTLCON=YES**

For CMAS and WUI regions, use the **CPSMCONN** SIT parameter instead of the **PLTPI** SIT parameter. Do not specify the **MQCONN=YES**, **DB2CONN=YES**, or **DBCTLCON=YES** SIT parameters in these regions.

For details of SIT parameters, see System initialization parameter descriptions and summary.

Use the tables to determine which ID is used:

- CMAS agent user ID
- MAS agent user ID
- WUI agent user ID

Table 19. CMAS agent user ID

CICSPlex SM release	How was the CMAS agent initialized?	PLTPIUSR in system initialization table?	CMAS agent user ID
Any in-service release	At initialization of the CICS region (CPSMCONN only)	Not applicable (no effect)	CICS region user ID
5.5 or earlier	At initialization of the CICS region (PLT entry only)	No	CICS region user ID
5.5 or earlier	At initialization of the CICS region (PLT entry only)	Yes	PLTPIUSR (see details in "What governs the agent user ID?" on page 168)

Table 20. MAS agent user ID

CICSPlex SM release	How was the MAS agent initialized?	PLTPIUSR in system initialization table?	MAS agent user ID
Any in-service release	At initialization of the CICS region (CPSMCONN only)	Not applicable	CICS region user ID
5.4 or later	Using the COLM transaction	Not applicable	CICS region user ID
5.3 or earlier	Using the COLM transaction	Not applicable	Signed-on user ID that initiated COLM
5.5 or earlier	At initialization of the CICS region (PLT entry only)	No	CICS region user ID
5.5 or earlier	At initialization of the CICS region (PLT entry only)	Yes	PLTPIUSR (see details in "What governs the agent user ID?" on page 168)

Table 21. WUI agent user ID

CICSPlex SM release	How was the WUI agent initialized?	PLTPIUSR in system initialization table?	WUI agent user ID
Any in-service release	At initialization of the CICS region (CPSMCONN only)	Not applicable	CICS region user ID
Any in-service release	Using the COVC transaction	Not applicable	Signed-on user ID that initiated COVC
5.5 or earlier	At initialization of the CICS region (PLT entry only)	No	CICS region user ID
5.5 or earlier	At initialization of the CICS region (PLT entry only)	Yes	PLTPIUSR (see details in "What governs the agent user ID?" on page 168)

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.

About this task

This is usually 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.

Procedure

1. You must define the following CICSPlex SM transactions to RACF:

BMLT	LPLT	PRLT	WMWC
LCPP	LPRT	PRPR	WMWD
LCMU	LPSC	PSLT	WMWT
LECI	LPSM	TICT	WSCL
LECR	LRLT	TIRT	WSLW
LECS	LSGT	TIST	XDBM
LEEI	LSRT	TSMH	XDNC
LEER	LWTM	TSPD	XDND
LEMI	MCCM	TSSC	XDNE
LEMS	MCTK	TSSJ	XDNR
LENS	MMEI	WMCC	XDNS
LMIR	MMIS	WMGR	XDSR
LNCI	MMST	WMLA	XLEV
LNCS	PEAD	WMQB	XLNX
LNMI	PELT	WMQM	XLST
LNMS	PMLT	WMQS	XMLT
LPDG	PNLT	WMSC	XQST
LPLK	PPLT		XZLT

   A list of these transactions is also contained in the EYU$CDEF member of SEYUSAMP sample library.

   a) Ensure that the CICS region user ID has authority to access these transactions.

      - If the CMAS starts by using the SIT parameter CPSMCONN, and a user ID is specified on a PLTPIUSR system initialization parameter, ensure that the PLTPIUSR user ID has authority to attach these transactions.

- If the CMAS starts by using the SIT parameter CPSMCONN, and no user ID is specified on a PLTPIUSR system initialization parameter, ensure that CICS region user ID 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 a connected CMAS have the authority to attach these transactions.

2. Define the following debugging transactions to RACF if transaction security is active, regardless of the CICS release that runs 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 CICSPlex SM 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 transactions in a managed CICS region

For CICS regions that are capable of running with an external security manager, it might be necessary to define the transactions that run in a CICS region managed by CICSPlex SM to the ESM.

About this task

If transaction–attach security is active in a CICS region (that is, SEC=YES and XTRAN=YES|*classname* are specified in the system initialization parameters), you must define the following transactions to RACF in the appropriate class:

COHT
COIE
COIR
COI0
CONA
COND
CONH
CONL
CONM
COWC

The region user ID must be given READ access to these transactions. These transactions are in category 1 of CICS-supplied transactions, and the necessary access can be defined together with all transactions in this category. For more information about category 1 of CICS-supplied transactions, see Security for CICS-supplied transactions.

If CICSPlex SM status probes (STATDEFs) run in a MAS, the region user ID must be given READ access to transaction CORT.

Users who might invoke the 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 “Overview of intercommunication security” on page 221 for information on intercommunication security.

The COSH transaction allows a terminal user to stop agent code execution. Access to this transaction should be restricted to those users who might need to stop the agent code in this way.

Setting up CICSPlex SM Web User Interface security

You can set Web User Interface security requirements for CICS security, Secure Sockets Layer (SSL) support, and access to MVS data sets.

User security access summary

Table 22 on page 172 summarizes the security accesses required by users of the Web User Interface.

Table 22. Security accesses required by users of the Web User Interface

User Roles	CICS Web Support	Administrator	User	View Editor
Transactions	COVP COVE COVU	COVG COVC	COVA	COVA
CICS surrogate user security		Yes		
View Editor profile				Yes
CICSPlex SM and CICS security			As appropriate for individual users	As appropriate for individual users

CICS security in your Web User Interface server region

If your Web User Interface server region is running with CICS security active, you must define the security access required for the CICS Web Support, by the administrator and by the users of the View Editor.

You can use CICS transaction security to limit the users who are allowed to control the Web User Interface server using the COVC transaction.

Security access for the CICS Web Interface

If CICS transaction security is in use, the CICS DFLTUSER must be given access to the COVP, COVU, and COVE transactions.

Security access for the administrator

The user ID that starts the Web User Interface (the terminal user of COVC or PLTPIUSR, if started automatically using PLTPI) must have access to the COVC and COVG transactions. If CICS surrogate user security checking is active in the Web User Interface server region, the user ID that started the Web User Interface (the terminal user of COVC or PLTPIUSR, if started automatically using PLTPI) must have READ access to wui-userid.DFHSTART in the SURROGAT class for all Web User Interface users.

Security access for users of the View Editor

Users of the Web User Interface require access to the COVA transaction and CICSPlex SM. Users of the View Editor require access to the COVA transaction, CICSPlex SM, and the View Editor profile.

All users who are successfully signed on to the Web User Interface have access to all of the customizable view and menu help pages, if the customizable view and menu help is served by the Web User Interface.

Secure Sockets Layer support

You can provide secure connections by using the Secure Sockets Layer (SSL) support to provide encryption on the connection. For information about SSL support, see Web User Interface server initialization parameters for information about the **TCPIPSSL** and **TCPIPSSLCERT** Web User Interface server initialization parameters that you must specify for SSL support and for more guidance on SSL, see Configuring CICS to use SSL.

Web User Interface SSL support uses server authentication only. User authentication is by the external security manager (ESM) user ID and password.

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, it is necessary to define the CICSPlex SM transactions that run in the region to the ESM.

Before you begin

This is necessary when transaction attach security is active in the WUI region, where SEC=YES and XTRAN=YES system initialization parameters are specified.

Procedure

1. Create the same definitions as you would for a MAS region.
   See “Defining the transactions in a managed CICS region” on page 171.
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 users.

Example

A list of these transactions is also contained in the CSD group EYU$WDEF.

Authorizing access to MVS data sets

In addition to standard CICS and CICSPlex SM requirements, the CICS region user ID must have the authority to access the data sets associated with the DD names described in the table.

Table 23. Security access required for MVS data sets

DDnames	Access required
EYUWUI	READ
DFHHTML	READ
EYUCOVI (and clones)	READ
EYUWREP	UPDATE
EYULOG	UPDATE
EYUCOVE (and clones)	UPDATE

Running your Web User Interface server with security active

If the Web User Interface server is running with the CICS system initialization parameter **SEC** set to YES, you can control who can access the Web User Interface, what resources they can see, what actions they can perform, and the use of the view editor.

If you have already set up CICSPlex SM security for use with the CICSPlex SM API, users have the same level of access with the Web User Interface as they do with the API.

When you attempt to connect to a Web User Interface server, the CICSPlex SM Web User Interface Signon Panel is displayed. The user ID and password entered in this panel are passed to the Web User Interface server, in plain text over the TCP/IP connection, unless you are using SSL support, and then verified by the external security manager. If the external security manager supports mixed case passwords, and this feature is active, an icon is displayed next to the password field when you sign on.

All users who are successfully signed on to the Web User Interface have access to all of the customizable view and menu help pages, if the customizable view and menu help is served by the Web User Interface.

If security is active, messages produced by auditing system programming interface commands contain the user ID used to log on to the WUI. See SPI commands that can be audited .

To control who is allowed to sign onto the Web User Interface server, you can protect the Web User Interface CICS application ID by using RACF APPL checking. See RACF classes for protecting system resources.

Access to managed resources uses standard CICSPlex SM security using profiles in the CPSMOBJ class. For example, to see a CICS region view, the Web User Interface user needs READ authority through the CPSMOBJ class profile OPERATE.REGION.context.scope.

Access to CICS resources, and actions on resources in a view, use the simulated CICS security checking of CICSPlex SM, which uses the normal CICS RACF resource and command security profiles. For example, to issue the shutdown action against a CICS region, if command security is active in the target CICS region, the Web User Interface user would need UPDATE authority to the SHUTDOWN command in the CCICSCMD class.

Controlling access to Web User Interface resources

You can use your external security manager to control user access to views, menus, editors, and help information and control the import and export of views, menus, maps, user groups, and user objects that use COVC.

The navigation frame is exempt from security checks. To control user access, you need to create the appropriate profiles in the FACILITY class. The following ESM FACILITY profiles are available:

EYUWUI.wui_server_applid.VIEW.viewsetname
- used to protect view sets.

EYUWUI.wui_server_applid.MENU.menuname
- used to protect menus.

EYUWUI.wui_server_applid.MAP.mapname
- used to protect maps.

EYUWUI.wui_server_applid.HELP.helpmembername
- used to protect help pages.

EYUWUI.wui_server_applid.EDITOR
- used to protect the View Editor.

EYUWUI.wui_server_applid.USER
- used to protect the User Editor and user and user group objects.

where `wui_server_applid` is the CICS APPLID of the server.

Users can be given read or update access to WUI resources:

- Read -- to use the views, menus, maps and Help information in the main interface, or to export views, menus, maps, user groups, or user objects that use COVC.
- Update -- to access the editors and create, update, or remove items that use them, or to import views, menus, maps, user groups, or user objects that use COVC.

To support AUTOIMPORT functions, the user ID that runs the COVG transaction must also be given update access to these FACILITY profiles. This is either the region user ID, or the PLTPIUSR value if a PLTPI table is in use.

If the ESM that you are using, neither grants nor refuses access to a profile (for example, if no RACF profile is defined), all users who are successfully signed on to the Web User Interface have access to the resources. You can make `not authorized` the default by setting up a generic profile.

Note: This security is designed to protect the views and menus themselves and not the objects they manage, which is covered by normal CICSPlex SM security.

When selecting a view set, map or menu to edit or delete within the view editor, only items for which you have update access are listed. However, when selecting an item to copy, all items for which you have read access are shown. This allows you to copy any object for which you have read only access to a private copy in your updateable namespace .

When browsing for views that are accessible, no security exceptions are logged. Users are presented with a list that has been filtered to remove the views that are not accessible. However, when a user attempts an unauthorized action; for example, creating a view in a denied namespace, the EYULOG security exception message EYUVS1100E is issued.

Examples of WUI security profiles

The following examples use the RACF TSO command syntax and assume that the default CICS RACF classes and no security prefixing is in use.

This is not the only way that suitable profiles can be defined. These examples can be adapted to suit the installations requirements and standards.

In the examples, lower case strings should be replaced with the appropriate use ID or resource.

- Example 1

  Create Web User Interface user groups:

  ```
  ADDGROUP (WUISERV,WUIADM,WUIUSER,WUIEDIT)
  ```

- Example 2

  Define profiles to protect Web User Interface transactions:

  ```
  RDEFINE GCICSTRN WUISYS UACC(NONE) ADDMEM(COVP,COVU,COVE)
  RDEFINE GCICSTRN WUIADMIN UACC(NONE) ADDMEM(COVG,COVC)
  RDEFINE TCICSTRN COVA UACC(NONE)
  ```

- Example 3

  Authorize user groups to appropriate profiles:

  ```
  PERMIT WUISYS CLASS(GCICSTRN) ID(WUISERV) ACCESS(READ)
  PERMIT WUIADMIN CLASS(GCICSTRN) ID(WUIADM) ACCESS(READ)
  PERMIT COVA CLASS(TCICSTRN) ID(WUIUSER,WUIEDIT) ACCESS(READ)
  ```

- Example 4

  Refresh transaction security profiles:

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

- Example 5

  Define View Editor profile and give user group appropriate access:

```
RDEFINE FACILITY EYUWUI.wui_server_applid.EDITOR UACC(NONE)
PERMIT EYUWUI.wui_server_applid.EDITOR CLASS(FACILITY) ID(WUIEDIT) ACCESS(UPDATE)
```

- Example 6

  Define view set profile and give user group appropriate access:

```
RDEFINE FACILITY EYUWUI.wui_server_applid.VIEW.viewsetname UACC(NONE)
PERMIT EYUWUI.wui_server_applid.VIEW.viewsetname CLASS(FACILITY) ID(WUIUSER)
ACCESS(READ)
```

- Example 7

  Connect users to appropriate Web User Interface groups:

```
CONNECT wui_server_dfltuser GROUP(WUISERV)
CONNECT (wui_server_pltpiusr,wui_administrator) GROUP(WUIADM)
CONNECT (wui_administrator,wui_view_designer) GROUP(WUIEDIT)
CONNECT (wui_operator1,wui_operator2...) GROUP(WUIUSER)
```

- Example 8

  If CICS surrogate user security is active in the Web User Interface region, definitions similar to the following are required:

```
DEFINE SURROGAT wui_administrator.DFHSTART UACC(NONE)
PERMIT wui_administrator.DFHSTART CLASS(SURROGAT) ID(WUIADM) ACCESS(READ)
DEFINE SURROGAT wui_view_designer.DFHSTART UACC(NONE)
PERMIT wui_view_designer.DFHSTART CLASS(SURROGAT) ID(WUIADM) ACCESS(READ)
DEFINE SURROGAT wui_operator1.DFHSTART UACC(NONE)
PERMIT wui_operator1.DFHSTART CLASS(SURROGAT) ID(WUIADM) ACCESS(READ)
DEFINE SURROGAT wui_operator2.DFHSTART UACC(NONE)
PERMIT wui_operator2.DFHSTART CLASS(SURROGAT) ID(WUIADM) ACCESS(READ)

SETROPTS RACLIST(SURROGAT) REFRESH
```

Running your Web User Interface server with security not active

If the Web User Interface server is running with the CICS system initialization parameter **SEC=NO**, users of the Web User Interface must provide a user ID to identify 'sessions' in the COVC transaction. Users are not required to provide a password.

The user ID does not need to be defined to the external security manager. Access checking of access to views, and CICS resources are based on the **DFLTUSER** for the Web User Interface server CICS region. All Web User Interface users have access to the View Editor and all menus, view sets and help members.

If security is not active, messages produced by auditing system programming interface commands contain the default user ID of either the CMAS or the MAS. See Changes to the SPI in the Upgrading documentation.

Specifying CICSPlex SM resource names in profiles

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.

About this task

The lists contain the resource names for CICSPlex SM views to be used in RACF profiles.

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 or platform. 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:

ANALYSIS
Real-time analysis

BAS
Business application services

CLOUD
Platforms and applications

CONFIG
CMAS configuration

CSD
CICS System Definition

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:

AIMODEL
CICS AIMODEL

APPLICTN
CICS bundle resources and resources only installable by CICS bundles

APPLICATION
Applications deployed in platforms

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

PLATFORM
Platforms

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 24 on page 180 lists the valid `function.type` combinations.

`context`
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, the `context` can be a CMAS or a CICSplex. For all other functions, the `context` must be a CICSplex.

`scope`
Scope is the specific or generic name of a CICS system in the CICSplex identified as the context. Do not specify `scope` when the context is a CMAS, the type is DEF (except when handling CSD resources) or when the function is CLOUD.

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 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 function and type combinations for CICSPlex SM resource profiles

You control access to a specific set of CICSPlex SM views and their associated action commands by identifying the set in a RACF resource profile with a defined resource name format, such as *function.type.context*. The *type* in this format must be valid for the specified *function*. This reference summary lists the valid function and type combinations and the resource names that are associated with each combination.

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 24. Function and type combinations for resources accessible using CICSPlex SM

Function.type combination	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.
BAS.APPLICTN	**BUNDDEF** Install applications that are deployed as bundles.

Table 24. Function and type combinations for resources accessible using CICSPlex SM (continued)

Function.type combination	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.

Table 24. Function and type combinations for resources accessible using CICSPlex SM (continued)

Function.type combination	Resource Table Name and Usage
BAS.DEF	**ATOMDEF** Create, display, and maintain Atom service definitions. **BUNDDEF** Create, display, and maintain bundle definitions. **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. **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. **IPCONDEF** Create, display, and maintain IPIC connection definitions. **JRNMDEF** Create, display, and maintain journal model definitions. **JVMSVDEF** Create, display, and maintain JVM server definitions. **LIBDEF** Create, display, and maintain LIBRARY definitions. **LSRDEF** Create, display, and maintain LSR pool definitions. **MAPDEF** Create, display, and maintain mapset definitions. **MQCONDEF** Create, display, and maintain IBM MQ connection definitions. **MQMONDEF** Create, display, and maintain IBM MQ monitor definitions. **PARTDEF** Create, display, and maintain partner definitions. **PIPEDEF** Create, display, and maintain pipeline definitions. **PROCDEF** Create, display, and maintain process type definitions.

Table 24. Function and type combinations for resources accessible using CICSPlex SM (continued)

Function.type combination	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 24. Function and type combinations for resources accessible using CICSPlex SM (continued)

Function.type combination	Resource Table Name and Usage
BAS.DEF (continued)	**ATMINGRP** Describe the membership of an Atom service definition in a resource group. **BUNINGRP** Describe the membership of a bundle definition in a resource group. **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. **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. **JMSINGRP** Describe the membership of a JVM server 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. **MQCINGRP** Display IBM MQ connection definitions in groups.

Table 24. Function and type combinations for resources accessible using CICSPlex SM (continued)

Function.type combination	Resource Table Name and Usage
BAS.DEF (continued)	**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. **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	**JVMSVDEF** Install JVM server definitions.
BAS.FILE	**FILEDEF** Install file definitions.

Table 24. Function and type combinations for resources accessible using CICSPlex SM (continued)

Function.type combination	Resource Table Name and Usage
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. **MQCONDEF** Install IBM MQ connection definitions. **MQMONDEF** Install IBM MQ monitor definitions. **TRNCLDEF** Install transaction class definitions.
BAS.TCPIPS	**ATOMDEF** Install Atom service definitions. **PIPEDEF** Install pipeline definitions. **TCPDEF** Install TCPIP service 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.

Table 24. Function and type combinations for resources accessible using CICSPlex SM (continued)

Function.type combination	Resource Table Name and Usage
BAS.TRAN	**TRANDEF** Install transaction definitions.
CLOUD.APPLICATION	**APPLDEF** Install application definitions in platforms. **APPLCTN** Display and manage applications deployed in platforms.
CLOUD.DEF	**APPLDEF** Create, display, and maintain application definitions in platforms. **PLATDEF** Create, display, and maintain platform definitions.
CLOUD.PLATFORM	**PLATDEF** Install platform definitions. **PLATFORM** Display and manage platforms. **MGMTPART** Create, display, and maintain management parts for applications and platforms.
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.

Table 24. Function and type combinations for resources accessible using CICSPlex SM (continued)

Function.type combination	Resource Table Name and Usage
CSD.DEF	**ATOMDEF** Create, install, display, and maintain Atom service definitions. **BUNDDEF** Create, install, display, and maintain bundle definitions. **CONNDEF** Create, install, display, and maintain MRO and ISC over SNA connection definitions. **CSDGROUP** Create, install, display, and maintain CSD resource group definitions. **CSDINGRP** Describe the membership of a CSD resource group. **CSDINLST** Describe the membership of a CSD resource group list. **CSDLIST** Create, install, display, and maintain CSD resource group list definitions. **DB2CDEF** Create, install, display, and maintain Db2 connection definitions. **DB2EDEF** Create, install, display, and maintain Db2 entry definitions. **DB2TDEF** Create, install, display, and maintain Db2 transaction definitions. **DOCDEF** Create, install, display, and maintain document template definitions. **ENQMDEF** Create, install, display, and maintain enqueue models definitions. **FILEDEF** Create, install, display, and maintain file definitions. **IPCONDEF** Create, install, display, and maintain IPIC connection definitions. **JRNMDEF** Create, install, display, and maintain journal model definitions. **JVMSVDEF** Create, install, display, and maintain JVM server definitions. **LIBDEF** Create, install, display, and maintain LIBRARY definitions. **LSRDEF** Create, install, display, and maintain LSR pool definitions. **MAPDEF** Create, install, display, and maintain mapset definitions. **MQCONDEF** Create, install, display, and maintain IBM MQ connection definitions. **MQMONDEF** Create, install, display, and maintain IBM MQ monitor definitions.

Table 24. Function and type combinations for resources accessible using CICSPlex SM (continued)

Function.type combination	Resource Table Name and Usage
CSD.DEF (continued)	**PARTDEF** Create, install, display, and maintain partner definitions. **PIPEDEF** Create, install, display, and maintain pipeline definitions. **PROCDEF** Create, install, display, and maintain process type definitions. **PROFDEF** Create, install, display, and maintain profile definitions. **PROGDEF** Create, install, display, and maintain program definitions. **PRTNDEF** Create, install, display, and maintain partition set definitions. **SESSDEF** Create, display, and maintain session definitions. **TCPDEF** Create, install, display, and maintain TCP/IP service definitions. **TDQDEF** Create, install, display, and maintain transient data queue definitions. **TERMDEF** Create, install, display, and maintain terminal definitions. **TRANDEF** Create, install, display, and maintain transaction definitions. **TRNCLDEF** Create, install, display, and maintain transaction class definitions. **TSMINGRP** Describe the membership of a temporary storage model definition in a resource group. **TYPTMDEF** Create, install, display, and maintain typeterm definitions. **URIMPDEF** Create, install, display, and maintain URI map definitions. **WEBSVDEF** Create, install, display, and maintain web services definitions.
MONITOR.CONNECT	**MCONNECT** ISC and MRO connections **MMODENAME** LU 6.2 modenames
MONITOR.DB2DBCTL	**MDB2THRD** Db2 threads

Table 24. Function and type combinations for resources accessible using CICSPlex SM (continued)

Function.type combination	Resource Table Name and Usage
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
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

Table 24. Function and type combinations for resources accessible using CICSPlex SM (continued)

Function.type combination	Resource Table Name and Usage
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 in a CICS system.

Table 24. Function and type combinations for resources accessible using CICSPlex SM (continued)

Function.type combination	Resource Table Name and Usage
OPERATE.APPLICTN	**BUNDLE** A CICS resource object that provides information about bundles. **BUNDPART** A CICS resource object that provides information about a bundle part. **CRESBUND** Describes an instance of a bundle in a CICS system. **CRESXMLT** Describes an instance of an XMLTRANSFORM resource in a CICS system. **EPADAPT** A CICS resource object that provides information about EP adapters. **EPADSET** A CICS resource object that provides information about EP adapter sets. **EPAINSET** A CICS resource object that provides the name of event processing adapters in an event processing adapter set. **EVCSDATA** A CICS resource object that provides information about application data predicates. **EVCSINFO** A CICS resource object that provides information about information sources. **EVCSOPT** A CICS resource object that provides information about application option predicates. **EVCSPEC** A CICS resource object that provides information about capture specifications. **EVNTBIND** A CICS resource object that provides information about event bindings. **EVNTGBL** A CICS resource object that provides information about event processing. **NODEJSAP** A CICS resource object that provides information about a Node.js application. **OSGIBUND** A CICS resource object that provides information about an OSGi bundle. **OSGISERV** A CICS resource object that provides information about an OSGi service. **XMLTRANS** XMLTRANSFORM resources.
OPERATE.BRFACIL	**BRFACIL** LINK3270 bridge facility

Table 24. Function and type combinations for resources accessible using CICSPlex SM (continued)

Function.type combination	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 in a CICS system. **CRESIPCN** Describe an instance of an IPIC connection in a CICS system. **CRESMODE** Describe an instance of an LU6.2 modename in a CICS system. **IPCONN** IPIC connections **MODENAME** LU 6.2 modenames
OPERATE.DB2DBCTL	**CRESDB2C** Describe an instance of a Db2 connection in a CICS system. **CRESDB2E** Describe an instance of a Db2 entry in a CICS system. **CRESDB2P** Describe an instance of a Db2 package set in a CICS region. **CRESDB2T** Describe an instance of a Db2 transaction in a CICS system. **DB2CONN** Db2 connection **DB2ENTRY** Db2 entry **DB2PKGST** Db2 package set **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 in a CICS system. **DOCTEMP** Document templates
OPERATE.ENQMODEL	**CRESENQM** Describe an instance of an ENQ/DEQ model in a CICS system. **ENQMODEL** Enqueue models

Table 24. Function and type combinations for resources accessible using CICSPlex SM (continued)

Function.type combination	Resource Table Name and Usage
OPERATE.ENTJAVA	**CLCACHE** Shared class caches **CRESJVMS** Describe an instance of a JVM server in a CICS region. **ENDPOINT** JVM Endpoints **JVM** Java™ virtual machines **JVMPOOL** JVM pools **JVMPROF** JVM profiles **JVMSERV** JVM servers
OPERATE.EXIT	**CRESGLUE** Describe an instance of a global user exit in a CICS system. **CRESTRUE** Describe an instance of a task-related user exit in 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 in a CICS system. **CRESFEND** Describe an instance of a FEPI node in a CICS system. **CRESFEPO** Describe an instance of a FEPI pool in a CICS system. **CRESFETR** Describe an instance of a FEPI target in a CICS system. **FEPICONN** FEPI connections **FEPINODE** FEPI nodes **FEPIPOOL** FEPI pools **FEPIPROP** FEPI property sets **FEPITRGT** FEPI targets

Table 24. Function and type combinations for resources accessible using CICSPlex SM (continued)

Function.type combination	Resource Table Name and Usage
OPERATE.FILE	**CFDTPOOL** Coupling facility data tables pools **CMDT** Data tables **CRESDSNM** Describe an instance of a data set in a CICS system. **CRESFILE** Describe an instance of a file in a CICS system. **DSNAME** Data sets **LOCFILE** Local files **REMFILE** Remote files
OPERATE.JOURNAL	**CRESJRNL** Describe an instance of a journal in a CICS system. **CRESJRNM** Describe an instance of a journal name in 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 in a CICS system. **PARTNER** CICS partners
OPERATE.PROCTYPE	**CRESPRTY** Describe an instance of a process type in a CICS system. **PROCTYP** Process types
OPERATE.PROFILE	**CRESPROF** Describe an instance of a profile in a CICS system. **PROFILE** CICS profiles

Table 24. Function and type combinations for resources accessible using CICSPlex SM (continued)

Function.type combination	Resource Table Name and Usage
OPERATE.PROGRAM	**CRESPRGM** Describe an instance of a program in a CICS system. **PROGRAM** Programs **LIBDSN** LIBRARY data set names **LIBRARY** LIBRARY data sets **RPLLIST** DFHRPL data sets

Table 24. Function and type combinations for resources accessible using CICSPlex SM (continued)

Function.type combination	Resource Table Name and Usage
OPERATE.REGION	**CICSDSA** Dynamic storage areas **CICSPAGP** CICS page pools **CICSRGN** CICS systems **CICSSTOR** All CICS dynamic storage areas **CRESMQMN** Describe an instance of an IBM MQ monitor in a CICS system **CRESSDMP** Describe an instance of a system dump code in a CICS system **CRESTDMP** Describe an instance of a transaction dump code in 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 **FEATURE** Feature toggles **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 **MQCON** IBM MQ connection **MQCONN** IBM MQ connection statistics **MQINI** IBM MQ initiation queue **MQMON** IBM MQ monitor **MVSESTG** MVS storage element **MVSTCB** MVS TCB

Table 24. Function and type combinations for resources accessible using CICSPlex SM (continued)

Function.type combination	Resource Table Name and Usage
OPERATE.REGION (continued)	**RECOVERY** CICS recovery manager **REQID** Timed requests **SYSDUMP** System dump codes **SYSPARM** A CICS Resource that describes a system initialization parameter or a system initialization parameter override in an active CICS system being managed by CICSPlex SM. **TRANCLAS** Transaction classes **TRANDUMP** Transaction dump codes
OPERATE.RQMODEL	**CRESRQMD** Describe an instance of a request in a CICS system.
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

Table 24. Function and type combinations for resources accessible using CICSPlex SM (continued)

Function.type combination	Resource Table Name and Usage
OPERATE.TCPIPS	**ATOMSERV** Atom services **CRESATOM** Describe an instance of an Atom service in a CICS system. **CRESTCPS** Describe an instance of a TCPIP service in 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 in 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 in a CICS system. **TERMNL** Terminals

Table 24. Function and type combinations for resources accessible using CICSPlex SM (continued)

Function.type combination	Resource Table Name and Usage
OPERATE.TRAN	**CRESTRAN** Describe an instance of a transaction in a CICS system. **LOCTRAN** Local transactions **REMTRAN** Remote transactions **TASKASSC** Task association data
OPERATE.TSQUEUE	**CRESTSMD** Describe an instance of a temporary storage queue in 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

Table 24. Function and type combinations for resources accessible using CICSPlex SM (continued)

Function.type combination	Resource Table Name and Usage
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 24. Function and type combinations for resources accessible using CICSPlex SM (continued)

Function.type combination	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 TORs 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.

Security for platforms and applications

You can secure resources for applications that are deployed on platforms by creating RACF security profiles for CICSPlex SM to cover platforms and applications in a CICSplex.

Security for platforms and applications is set up in a similar way to security for other CICSPlex SM components. You control access to a specific set of views (and their associated action commands) by identifying the set in a security profile. With these security profiles, you can give users authority to install, enable or disable, make available or unavailable, inquire on, or discard platforms and applications, and ensure that unauthorized users cannot create and administer these resources.

When you give a user authority to perform an action on a platform or application, you also give them authority to perform the same action on the dynamically generated resources for the platform or application. For example, a user who has authority to enable an application also has authority to enable the CICS bundles for the application that were installed in CICS regions in all the platforms in

the CICSplex. CICS command and resource security checks, and simulated CICS security checking in CICSPlex SM, are not carried out when you operate on CICS bundles through an application or platform.

You can secure a platform and its deployed applications by setting up security profiles with the following function and type combinations:

CLOUD.DEF.*context*
This security profile covers the PLATDEF and APPLDEF resource tables, which contain the definitions for platforms and applications. *context* is the specific or generic name of the CICSplex that is covered by the security profile.

Users with UPDATE access for this security profile can create, update, and remove definitions for platforms and applications in the CICSPlex SM data repository. Users with READ access can view those definitions in the CICSPlex SM data repository.

CLOUD.PLATFORM.*context*
This security profile covers the installation of PLATDEF resources and operations on PLATFORM resources. It also allows users to view management parts (MGMTPART resources). *context* is the specific or generic name of the CICSplex that is covered by the security profile.

Users with ALTER access for this security profile can install platforms in the CICSplex and discard them. (To install a platform, users also need READ access for the CLOUD.DEF profile that covers the PLATDEF resource.) Users with UPDATE access can enable and disable platforms. Users with UPDATE access can also add CICS regions to region types in the platform and remove CICS regions from region types in the platform. Users with READ access can view PLATFORM resources and MGMTPART resources. These permissions apply for all platforms that exist in the CICSplex.

CLOUD.APPLICATION.*context*
This security profile covers the installation of APPLDEF resources and operations on APPLCTN resources. *context* is the specific or generic name of the CICSplex that is covered by the security profile.

Users with ALTER access for this security profile can install applications in the CICSplex and discard them. (To install an application, users also need READ access for the CLOUD.DEF profile that covers the APPLDEF resource.) Users with UPDATE access can enable and disable applications and make them available or unavailable. Users with READ access can view APPLCTN resources. These permissions apply for all applications in all platforms that exist in the CICSplex. If you require different security permissions for certain applications, use a different CICSplex to host the platform where you deploy the application.

Note: These security profiles are only checked in the maintenance point CMAS. Security checks are reported by message EYUCR0009I in the EYULOG of the maintenance point CMAS. To receive message EYUCR0009I for violations you must set the CICSPlex SM system parameter (EYUPARM) **SECLOGMSG** to YES. For more information about **SECLOGMSG**, see CICSPlex SM system parameters.

Although the CLOUD security profiles cover actions on the dynamically generated resources for the platform or application, users may still carry out a limited set of actions directly on individual resources in the CICSplex and CICS regions where they are installed:

- You can modify the CICSPlex SM topology definition (CSYSDEF, or CICS system definition) for a CICS region that is part of a platform. Attribute values that you specified at the region type level are locked and cannot be changed, but other attribute values can be changed.
- You can make available or unavailable, enable or disable, or inquire on, a BUNDLE resource that was dynamically created when you installed a platform or application. You cannot discard an individual bundle directly if it was created when you installed a platform or application.
- You can inquire on a dynamically created resource, such as a PROGRAM resource, that was defined inside a CICS bundle and created when you installed a platform or application. You cannot enable, disable, or discard these resources directly if they were created when you installed a platform or application.

CICS command and resource security checks, and simulated CICS security checking in CICSPlex SM, do apply when you perform an action directly on a CICS region that is part of a platform, or on an individual

CICS bundle, or a resource defined in a CICS bundle, that was created when you installed a platform or application.

- A TOPOLOGY.DEF.*context* security profile covers actions on the CICSPlex SM topology definitions for individual CICS regions that are part of a platform. *context* is the specific or generic name of the CICSplex that is covered by the security profile. Users with UPDATE access can modify the CSYSDEF for a CICS region that is part of a platform, with the exception of attribute values that are locked by the platform itself.
- CICS bundles created when you install a platform or application have a unique generated name beginning with the $ character. To provide security for actions on individual CICS bundles that were dynamically created in this way, you can set up a security profile specifying the BUNDLE resource type and the resource name $*. Users with UPDATE access for BUNDLE.$* can make available or unavailable, or enable or disable, BUNDLE resources created for platforms and applications, and users with READ access can inquire on those BUNDLE resources.

If you apply security measures to individual PROGRAM resources, for applications that are deployed on platforms, secure the programs that are declared as application entry points, but do not secure other programs in the applications. The security settings that you specify for a program that is part of an application deployed on a platform apply to both public and private programs, and do not take into account the version of the application. Programs that are declared as an application entry point must have a unique PROGRAM resource name in your environment. However, if you secure programs that run at a lower level in the application, programs with the same names might be running in different applications, which can lead to unforeseen consequences. In this situation, a user might have permission to access a program that is declared as an application entry point, but not have permission to access a program that runs at a lower level in the application, because the security settings from another instance of the program name are in effect. Consider the security measures that you apply to a program that is declared as an application entry point program, as applying to the whole application.

If you used CICS bundles in earlier CICS releases, check the security permissions that you gave to users for those bundles. Depending on the way in which you set up security for CICS bundles, users with authority to take actions on individual CICS bundles might now be able to act on resources that are dynamically created as part of the installation of a bundle. Ensure that the levels of authority for BUNDLE resources are still appropriate.

Table 25 on page 204 summarizes the security checks that apply to actions performed on a platform, an application, an individual CICS bundle that was dynamically created when you installed a platform or application, or a resource defined in a CICS bundle for a platform or application.

Table 25. Security checks for operations on platforms, applications, and generated CICS bundles

Operation	**Platforms, including their CICS bundles**	**Applications, including their bundles**	**Dynamically-created CICS bundles**	**Resources defined in dynamically-created CICS bundles**
Define	CLOUD.DEF profile (UPDATE, or READ to view definitions); also TOPOLOGY.DEF profile (UPDATE to modify individual CICS region CSYSDEF after platform install)	CLOUD.DEF profile (UPDATE, or READ to view definitions)	Cannot manage resource definitions individually	Cannot manage resource definitions individually
Install	CLOUD.PLATFORM profile (ALTER) and CLOUD.DEF profile (READ)	CLOUD.APPLICATION profile (ALTER) and CLOUD.DEF profile (READ)	Cannot install individually	Cannot install individually
Enable or disable	CLOUD.PLATFORM profile (UPDATE)	CLOUD.APPLICATION profile (UPDATE)	CICS command and resource security, and simulated CICS security checking in CICSPlex SM; use BUNDLE.$* profile	Cannot enable or disable individually

Table 25. Security checks for operations on platforms, applications, and generated CICS bundles (continued)				
Operation	**Platforms, including their CICS bundles**	**Applications, including their bundles**	**Dynamically-created CICS bundles**	**Resources defined in dynamically-created CICS bundles**
Make available or unavailable	Not applicable	CLOUD.APPLICATION profile (UPDATE)	CICS command and resource security, and simulated CICS security checking in CICSPlex SM; use BUNDLE.$* profile	Cannot make available or unavailable individually
Inquire	CLOUD.PLATFORM profile (READ); also allows viewing of management parts	CLOUD.APPLICATION profile (READ)	CICS command and resource security, and simulated CICS security checking in CICSPlex SM; use BUNDLE.$* profile	CICS command and resource security, and simulated CICS security checking in CICSPlex SM
Discard	CLOUD.PLATFORM profile (ALTER)	CLOUD.APPLICATION profile (ALTER)	Cannot discard individually	Cannot discard individually

For more information on setting up security for CICSPlex SM and creating security profiles, see Implementing CICSPlex SM security.

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.

About this task

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, CICS commands, or both. 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.

Note:

1. See “Activating security for CICSPlex SM” on page 207 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.
3. CICSPlex SM simulated CICS security does not include the simulation of CICS surrogate security. If CICS surrogate security checking is required, see “Considerations for CICS surrogate security checks” on page 206 for guidance on how to secure your CICS regions correctly.

Procedure

- 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.
- To activate or deactivate simulated security checking temporarily for an active CICS system, use the MAS view.

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 176.
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 do not want to bypass security checking, specify `ACCESS(NONE)`.
   - If you want to bypass security checking of INQUIRE level commands, specify `ACCESS(READ)`.
   - If you want to bypass security checking of INQUIRE, SET, and PERFORM level commands, specify `ACCESS(UPDATE)`.
   - If you want to bypass security checking of all commands, including DISCARD level commands, specify `ACCESS(ALTER)`.
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  */
```

Considerations for CICS surrogate security checks

If CICS surrogate security checking is required, you need to work out which user ID against which CICS surrogate security checks will be performed and plan for that in the RACF definitions.

In some situations, CICS issues a surrogate security check on a user that requests an action that is required to be run on behalf of another user. To allow this request to be processed, the user that requests the action must have surrogate access for the other user ID.

However, CICSPlex SM simulated CICS security does not include the simulation of CICS surrogate security.

For CICS surrogate security, the checks are performed against the MAS agent user ID rather than the user that issues the request. To determine the MAS agent user ID, follow the instructions in Determining the agent user ID for CICSPlex SM components.

Activating security for CICSPlex SM

To activate security in the CMASs and MASs, you must set the CICSPlex SM and CICS security-related system initialization parameters.

Procedure

1. Specify the CICSPlex SM parameter SEC in the EYUPARM data set or member defined in the JCL used to start the CMAS and MAS.
2. Specify the CICS parameter SEC= in the CICS system initialization parameters used to start the MAS.

Results

Together these parameters determine what security checking is performed. Table 26 on page 207 explains the possible parameter combinations.

Table 26. 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.

About this task

This includes refreshing general resource profiles and user profiles in the cache.

Procedure

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 from the cache.

4. 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.
5. 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 5 on page 210 and Figure 6 on page 211 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" on page 209.
3. CICSPlex SM determines whether the user is exempt from simulated CICS security checking.
   - If so, processing continues at "9" on page 209.
4. CICSPlex SM determines whether simulated CICS command checking is to be performed.
   - If not, processing continues at "6" on page 209.
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" on page 209.
7. CICSPlex SM determines whether simulated CICS resource checking is to be performed.
   - If not, processing continues at "9" on page 209.
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" on page 209.
    - Otherwise, CICSPlex SM returns the requested information in the appropriate view.

No further security checking is required.

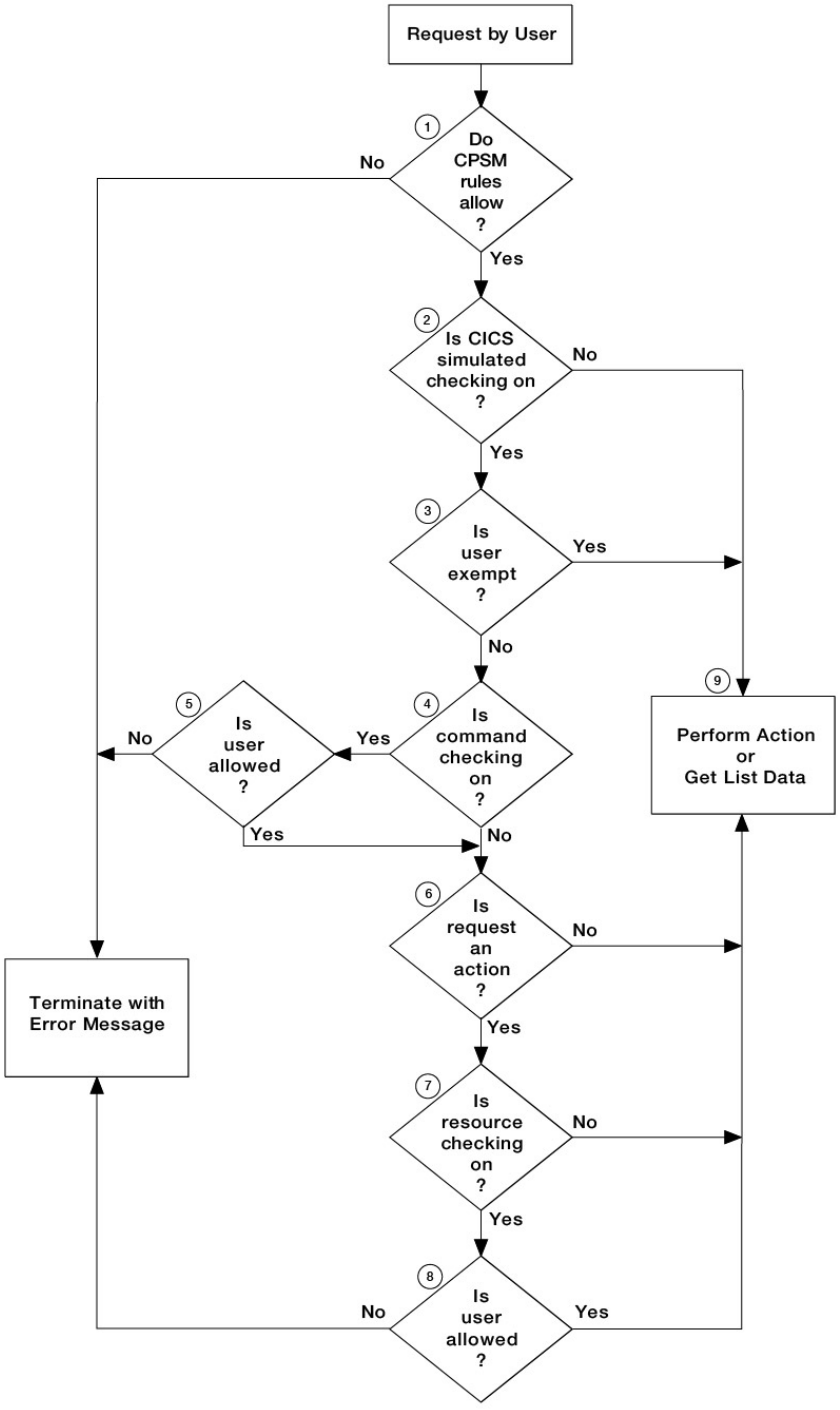


Figure 5. Flowchart of CICSPlex SM security checking sequence - part 1

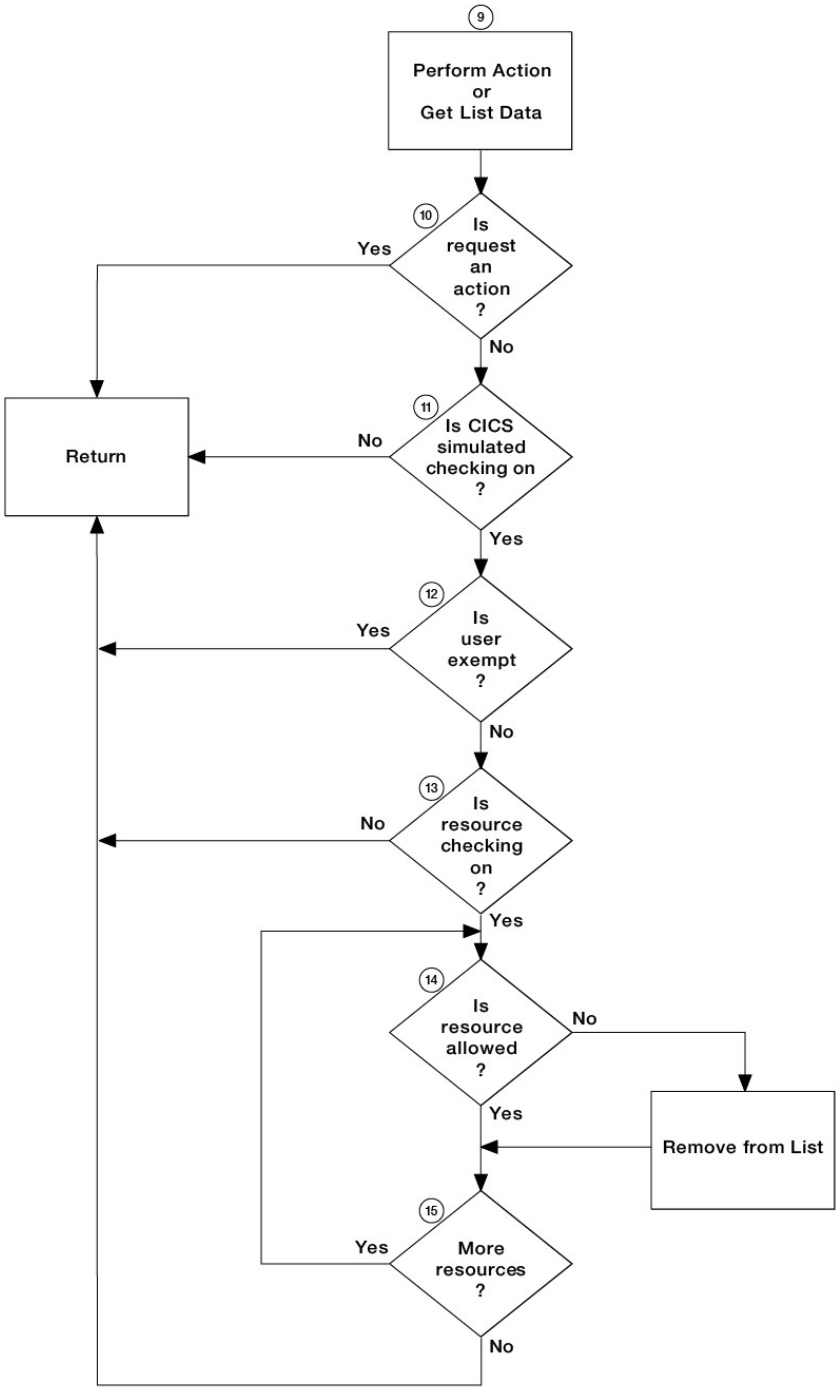


Figure 6. Flowchart of CICSPlex SM security checking sequence - part 2

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).

On the 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, see z/Architecture Principles of Operation.

Note: This information is intended primarily for non-RACF users. For information about security processing using RACF, refer to "Implementing CICSPlex SM security" on page 163.

An overview of the CICSPlex SM ESM interface

CICSPlex SM security uses the RACROUTE macro to get the MVS system authorization facility (SAF) interface to route authorization requests to the ESM.

Usually, 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 212.

Overview of 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, for example 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 might 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. For more information, see Passing control to a user-supplied ESM.

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 z/OS Security Server RACF Messages and Codes.

For more information about the MVS router, see System authorization facility (SAF) in z/OS Security Server RACROUTE Macro Reference and ICHRTX00 - MVS Router Exit in z/OS MVS Installation Exits.

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, 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 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 security action command (SEC) is issued.

For a detailed description of these macros, see z/OS Security Server RACF Macros and Interfaces.

Example tasks: security

The following information 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 do not use this convention 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.

The first example describes creating some RACF profiles to protect all CICSPlex SM functions and resources. The next step is to 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" on page 214: 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 cannot replace Step "2" on page 214 with multiple CICSplex-specific profiles: such profiles will not 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 unprotected if you did not also perform Step "2" on page 214.

4. In Step "3" on page 214 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)
```

Chapter 6. Security for platforms and applications

You can secure resources for applications that are deployed on platforms by creating RACF security profiles for CICSPlex SM to cover platforms and applications in a CICSplex.

Security for platforms and applications is set up in a similar way to security for other CICSPlex SM components. You control access to a specific set of views (and their associated action commands) by identifying the set in a security profile. With these security profiles, you can give users authority to install, enable or disable, make available or unavailable, inquire on, or discard platforms and applications, and ensure that unauthorized users cannot create and administer these resources.

When you give a user authority to perform an action on a platform or application, you also give them authority to perform the same action on the dynamically generated resources for the platform or application. For example, a user who has authority to enable an application also has authority to enable the CICS bundles for the application that were installed in CICS regions in all the platforms in the CICSplex. CICS command and resource security checks, and simulated CICS security checking in CICSPlex SM, are not carried out when you operate on CICS bundles through an application or platform.

You can secure a platform and its deployed applications by setting up security profiles with the following function and type combinations:

CLOUD.DEF.*context*
: This security profile covers the PLATDEF and APPLDEF resource tables, which contain the definitions for platforms and applications. *context* is the specific or generic name of the CICSplex that is covered by the security profile.

  Users with UPDATE access for this security profile can create, update, and remove definitions for platforms and applications in the CICSPlex SM data repository. Users with READ access can view those definitions in the CICSPlex SM data repository.

CLOUD.PLATFORM.*context*
: This security profile covers the installation of PLATDEF resources and operations on PLATFORM resources. It also allows users to view management parts (MGMTPART resources). *context* is the specific or generic name of the CICSplex that is covered by the security profile.

  Users with ALTER access for this security profile can install platforms in the CICSplex and discard them. (To install a platform, users also need READ access for the CLOUD.DEF profile that covers the PLATDEF resource.) Users with UPDATE access can enable and disable platforms. Users with UPDATE access can also add CICS regions to region types in the platform and remove CICS regions from region types in the platform. Users with READ access can view PLATFORM resources and MGMTPART resources. These permissions apply for all platforms that exist in the CICSplex.

CLOUD.APPLICATION.*context*
: This security profile covers the installation of APPLDEF resources and operations on APPLCTN resources. *context* is the specific or generic name of the CICSplex that is covered by the security profile.

  Users with ALTER access for this security profile can install applications in the CICSplex and discard them. (To install an application, users also need READ access for the CLOUD.DEF profile that covers the APPLDEF resource.) Users with UPDATE access can enable and disable applications and make them available or unavailable. Users with READ access can view APPLCTN resources. These permissions apply for all applications in all platforms that exist in the CICSplex. If you require different security permissions for certain applications, use a different CICSplex to host the platform where you deploy the application.

Note: These security profiles are only checked in the maintenance point CMAS. Security checks are reported by message EYUCR0009I in the EYULOG of the maintenance point CMAS. To receive message EYUCR0009I for violations you must set the CICSPlex SM system parameter (EYUPARM) **SECLOGMSG** to YES. For more information about **SECLOGMSG**, see CICSPlex SM system parameters.

Although the CLOUD security profiles cover actions on the dynamically generated resources for the platform or application, users may still carry out a limited set of actions directly on individual resources in the CICSplex and CICS regions where they are installed:

- You can modify the CICSPlex SM topology definition (CSYSDEF, or CICS system definition) for a CICS region that is part of a platform. Attribute values that you specified at the region type level are locked and cannot be changed, but other attribute values can be changed.
- You can make available or unavailable, enable or disable, or inquire on, a BUNDLE resource that was dynamically created when you installed a platform or application. You cannot discard an individual bundle directly if it was created when you installed a platform or application.
- You can inquire on a dynamically created resource, such as a PROGRAM resource, that was defined inside a CICS bundle and created when you installed a platform or application. You cannot enable, disable, or discard these resources directly if they were created when you installed a platform or application.

CICS command and resource security checks, and simulated CICS security checking in CICSPlex SM, do apply when you perform an action directly on a CICS region that is part of a platform, or on an individual CICS bundle, or a resource defined in a CICS bundle, that was created when you installed a platform or application.

- A TOPOLOGY.DEF.*context* security profile covers actions on the CICSPlex SM topology definitions for individual CICS regions that are part of a platform. *context* is the specific or generic name of the CICSplex that is covered by the security profile. Users with UPDATE access can modify the CSYSDEF for a CICS region that is part of a platform, with the exception of attribute values that are locked by the platform itself.
- CICS bundles created when you install a platform or application have a unique generated name beginning with the $ character. To provide security for actions on individual CICS bundles that were dynamically created in this way, you can set up a security profile specifying the BUNDLE resource type and the resource name $*. Users with UPDATE access for BUNDLE.$* can make available or unavailable, or enable or disable, BUNDLE resources created for platforms and applications, and users with READ access can inquire on those BUNDLE resources.

If you apply security measures to individual PROGRAM resources, for applications that are deployed on platforms, secure the programs that are declared as application entry points, but do not secure other programs in the applications. The security settings that you specify for a program that is part of an application deployed on a platform apply to both public and private programs, and do not take into account the version of the application. Programs that are declared as an application entry point must have a unique PROGRAM resource name in your environment. However, if you secure programs that run at a lower level in the application, programs with the same names might be running in different applications, which can lead to unforeseen consequences. In this situation, a user might have permission to access a program that is declared as an application entry point, but not have permission to access a program that runs at a lower level in the application, because the security settings from another instance of the program name are in effect. Consider the security measures that you apply to a program that is declared as an application entry point program, as applying to the whole application.

If you used CICS bundles in earlier CICS releases, check the security permissions that you gave to users for those bundles. Depending on the way in which you set up security for CICS bundles, users with authority to take actions on individual CICS bundles might now be able to act on resources that are dynamically created as part of the installation of a bundle. Ensure that the levels of authority for BUNDLE resources are still appropriate.

Table 27 on page 219 summarizes the security checks that apply to actions performed on a platform, an application, an individual CICS bundle that was dynamically created when you installed a platform or application, or a resource defined in a CICS bundle for a platform or application.

Table 27. Security checks for operations on platforms, applications, and generated CICS bundles

Operation	Platforms, including their CICS bundles	Applications, including their bundles	Dynamically-created CICS bundles	Resources defined in dynamically-created CICS bundles
Define	CLOUD.DEF profile (UPDATE, or READ to view definitions); also TOPOLOGY.DEF profile (UPDATE to modify individual CICS region CSYSDEF after platform install)	CLOUD.DEF profile (UPDATE, or READ to view definitions)	Cannot manage resource definitions individually	Cannot manage resource definitions individually
Install	CLOUD.PLATFORM profile (ALTER) and CLOUD.DEF profile (READ)	CLOUD.APPLICATION profile (ALTER) and CLOUD.DEF profile (READ)	Cannot install individually	Cannot install individually
Enable or disable	CLOUD.PLATFORM profile (UPDATE)	CLOUD.APPLICATION profile (UPDATE)	CICS command and resource security, and simulated CICS security checking in CICSPlex SM; use BUNDLE.$* profile	Cannot enable or disable individually
Make available or unavailable	Not applicable	CLOUD.APPLICATION profile (UPDATE)	CICS command and resource security, and simulated CICS security checking in CICSPlex SM; use BUNDLE.$* profile	Cannot make available or unavailable individually
Inquire	CLOUD.PLATFORM profile (READ); also allows viewing of management parts	CLOUD.APPLICATION profile (READ)	CICS command and resource security, and simulated CICS security checking in CICSPlex SM; use BUNDLE.$* profile	CICS command and resource security, and simulated CICS security checking in CICSPlex SM
Discard	CLOUD.PLATFORM profile (ALTER)	CLOUD.APPLICATION profile (ALTER)	Cannot discard individually	Cannot discard individually

For more information on setting up security for CICSPlex SM and creating security profiles, see Implementing CICSPlex SM security.

Chapter 7. Security for intercommunication

These topics tell you how to plan and implement security in an intersystem communication (ISC) environment using LU6.2 or LU6.1, in a multiregion operation (MRO) environment, and in an IP interconnectivity (IPIC) environment.

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 55.
- How the relationship between user security and transaction security determines which transactions a particular user is allowed to invoke. (See “Verifying CICS users” on page 53 and “Transaction security” on page 69.)
- How resource security determines which other resources a user is allowed to access. See “Resource security” on page 73.

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 11 and “RACF group profiles” on page 16 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 Distributed transaction processing overview.

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 224, "IPIC bind-time security" on page 261, or "Bind-time security with MRO" on page 268, 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" on page 223.

User security for intercommunication

In addition to the security profile that you set up for the link, you can further restrict each remote user's access to the transactions, commands, and resources in your system.

For ISC over SNA and MRO links, specify the ATTACHSEC parameter on the CONNECTION definition. For IPIC links, specify the USERAUTH parameter on the IPCONN resource definition.

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" on page 223.

For information on defining user security in your system, see “User security with LU6.2” on page 229, “IPIC user security” on page 264, or “User security with MRO” on page 272, 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 Transaction security, Security of resource definitions, and CICS command security.

Resources and commands are unsecured unless you explicitly request security protection in your transaction definitions.

Summary of intercommunication security levels

Table 28 on page 223 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 Intercommunication methods.

Table 28 on page 223 shows a summary of intercommunication security.

Table 28. 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		

Table 28. Summary of intersystem and interregion security (continued)					
Security level	**Security checks**	**LU type 6.1**	**LU type 6.2**	**IPIC**	**MRO**
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 RACF facilities.

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, 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 specified 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 29 on page 225 summarizes what happens.

Table 29. 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.
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 z/OS Communications Server session establishment.

To do this, take the following steps:

1. Ask your z/OS Communications Server 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 z/OS Communications Server 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 resource. For more information see CONNECTION resources.

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” on page 226. (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 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 resource definition, although you must also choose the appropriate system initialization parameters.

Figure 7 on page 227 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 7. Bind-time security

For APPC terminals defined as a TERMINAL-TYPETERM pair, the BINDSECURITY operand is on the TERMINAL definition.

Note: VTAM is the z/OS Communications Server (for SNA or IP)

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.

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 user ID has no authority.

When link security is in use, each session is given an authority defined by a *link user ID*. For LU6.2, all sessions in a connection can have the same link user ID, or different groups of sessions within the connection can have different link user IDs. 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 user ID matches the local region user ID:

- If the user IDs match, only one security check is made. This will either be against the default user (for ATTACHSEC=LOCAL) or against the user ID that is in the received FMH-5 attach request (ATTACHSEC=non-LOCAL).
- If the user IDs do not match, then for ATTACHSEC=LOCAL, resource checks are done only against the link user ID. For ATTACHSEC=non-LOCAL there are always two resource checks. One is against the link user ID, and the second is against the user ID 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 user ID had been revoked.

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 228. 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 230 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" on page 229 further describes these types of user checking. See "Specifying user security in link definitions" on page 230 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.

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.

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 identity 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” on page 230.

Specifying user security in link definitions

The level of user security you require for a remote system is specified in the ATTACHSEC operand in the CONNECTION definition, as shown in Figure 8 on page 230.

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 235. Figure 8 on page 230 shows an example of defining ATTACHSEC using CEDA.

```
CEDA DEFINE CONNECTION(name)
  GROUP(groupname)

  ATTACHSEC(LOCAL|IDENTIFY|VERIFY|PERSISTENT|MIXIDPE)
```

Figure 8. 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 233.

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.

Remote user sign-on status with LU6.2

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 one of the following events occurs:

- The period specified in the USRDELAY system initialization parameter system initialization parameter elapses after completion of the last transaction associated with the attach request for this user.
- CICS is notified of certain changes in the RACF profile of a signed-on remote user, or a signed-on user who is not directly using a physical terminal or console, through a type 71 RACF Event Notification (ENF). For details, see “Changing the RACF profile of a remote user” on page 14.
- The CICS region shuts down.

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 one of the following situations occurs:

- The transaction performs a syncpoint.
- The attach request ends.

- Sign off occurs because RACF notifies CICS of changes to a user profile, and an attached request associated with that signed-on user ID completes, for all operands of ATTACHSEC except LOCAL.
- Sign off occurs because RACF notifies CICS of changes to a user profile, a new attach request is made, and the value in the **USRDELAY** system initialization parameter has not expired. This sign off is followed by a sign on.

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 PVDELAY system initialization parameter . See the Improving the performance of a CICS system 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 30 on page 233 shows how CICS decides which userid to send.

Table 30. 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 “Remote user sign-on status with LU6.2” on page 231.

SNA profiles and attach-time security

Implementation of the LU6.2 attach-time security in CICSTS56.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 might cause upgrade 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 resource 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 an 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 Transaction security.

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 236.

CICS routing transaction, CRTE

You can use the CICS routing transaction, CRTE, with IPIC, LU6.2, or MRO links to run transactions that are 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 are on all systems.

Ensure that the terminal through which CRTE is started is defined on the remote system or is defined as shippable in the local system. The terminal operator needs RACF authority if the remote system is protected.

Security checking in the AOR for transactions that run under CRTE does not depend on what is specified on ATTACHSEC (for MRO and LU6.2 links) or USERAUTH (for IPIC links), nor does it depend on the user ID signed on in the TOR. Instead, security checking depends on whether the user signs on when 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 made against the link user ID 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 user ID of the signed-on user. For transactions attempting to access resources, security checking is made against the user ID of the signed-on user in the surrogate and the link user ID.

Resource and command security

Resource and command security in an intercommunication environment are handled in much the same way as in a single-system environment.

Resource and command security checking are performed only if the installed TRANSACTION definition specifies that they are required; for example, on the CEDA DEFINE TRANSACTION command, as shown in Figure 9 on page 235.

```
CEDA DEFINE TRANSACTION
  .
  RESSEC(YES)
  CMDSEC(YES)
  .
```

Figure 9. Specifying resource and command security for transactions

If a TRANSACTION definition specifies resource security checking, using RESSEC(YES), both the link and the user must also have sufficient authority for the resources that the attached transaction accesses.

If a TRANSACTION definition specifies command security checking, using CMDSEC(YES), both the link and the user must also have sufficient authority for the 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 Security of resource definitions and CICS command security.

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 228.

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 IPIC, LU6.2, or MRO links to run transactions that are 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 are on all systems.

Ensure that the terminal through which CRTE is started is defined on the remote system or is defined as shippable in the local system. The terminal operator needs RACF authority if the remote system is protected.

Security checking in the AOR for transactions that run under CRTE does not depend on what is specified on ATTACHSEC (for MRO and LU6.2 links) or USERAUTH (for IPIC links), nor does it depend on the user ID signed on in the TOR. Instead, security checking depends on whether the user signs on when 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 made against the link user ID 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 user ID of the signed-on user. For transactions attempting to access resources, security checking is made against the user ID of the signed-on user in the surrogate and the link user ID.

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 Security for CICS-supplied transactions.

If you include a remote resource in your resource definitions, you can arrange for security checking to be done locally, just as if the resource were a local one. Also, the system that owns the resource can be made to apply an independent check, if it is able to receive the user identifier. You can therefore choose to apply security restrictions on both sides, on either side, or not at all.

Note: Be aware that if you specify the SYSID option on a function-shipped request, security checking is done in the remote system but is **bypassed in the local system**. Figure 10 on page 237 summarizes what happens.

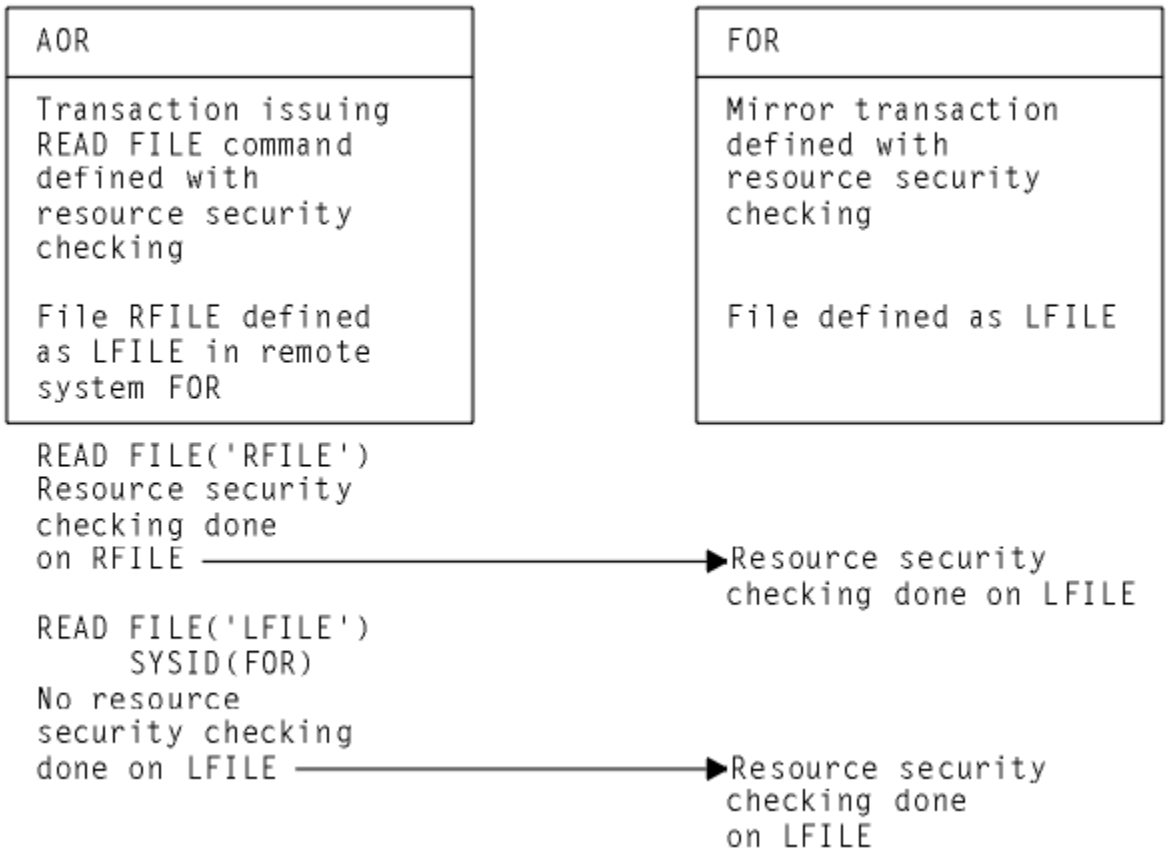


Figure 10. Security checking done with and without SYSID

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 API 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 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 238. 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 CICS command summary. For further information about DPL, refer to CICS distributed program link.

Security checking done in AOR with LU6.2

Security checking is different depending on how SECURITYNAME is specified in the AOR and TOR.

The link userid referred to in Table 31 on page 239 and Table 32 on page 239 is the one specified in the SECURITYNAME on the CONNECTION resource definition, or the USERID on the SESSION resource 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 31 on page 239 shows how checking is done when ATTACHSEC(LOCAL) is specified.

Table 31. 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 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 DFLTUSER
USERIDA	USERIDB	USERIDB	Check against USERIDB
USERIDA	USERIDB	USERIDC	Check against USERIDC

Table 32 on page 239 shows how checking is done when the ATTACHSEC parameter IDENTIFY (or PERSISTENT, or MIXIDPE) has been specified.

Table 32. 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

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

About this task

You specify link security on the CONNECTION or SESSIONS resource definitions. If you require more information about either of these resources, see CONNECTION resources and SESSIONS resources.

Procedure

- To specify that all sessions of a connection should have the same link userid, specify the SECURITYNAME attribute of the CONNECTION resource 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 resource 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 resource definition. Figure 11 on page 240 shows an example of doing this using CEDA.

```
CEDA DEFINE CONNECTION(name)
  GROUP(groupname)
  .
  ATTACHSEC(LOCAL)
```

Figure 11. 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 Transaction security.

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 12 on page 241.

```
CEDA DEFINE TRANSACTION
  .
  RESSEC(YES)
  CMDSEC(YES)
  .
```

Figure 12. 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 Security of resource definitions and Security of resource definitions.

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 Security for CICS-supplied transactions.

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 13 on page 242 summarizes what happens.

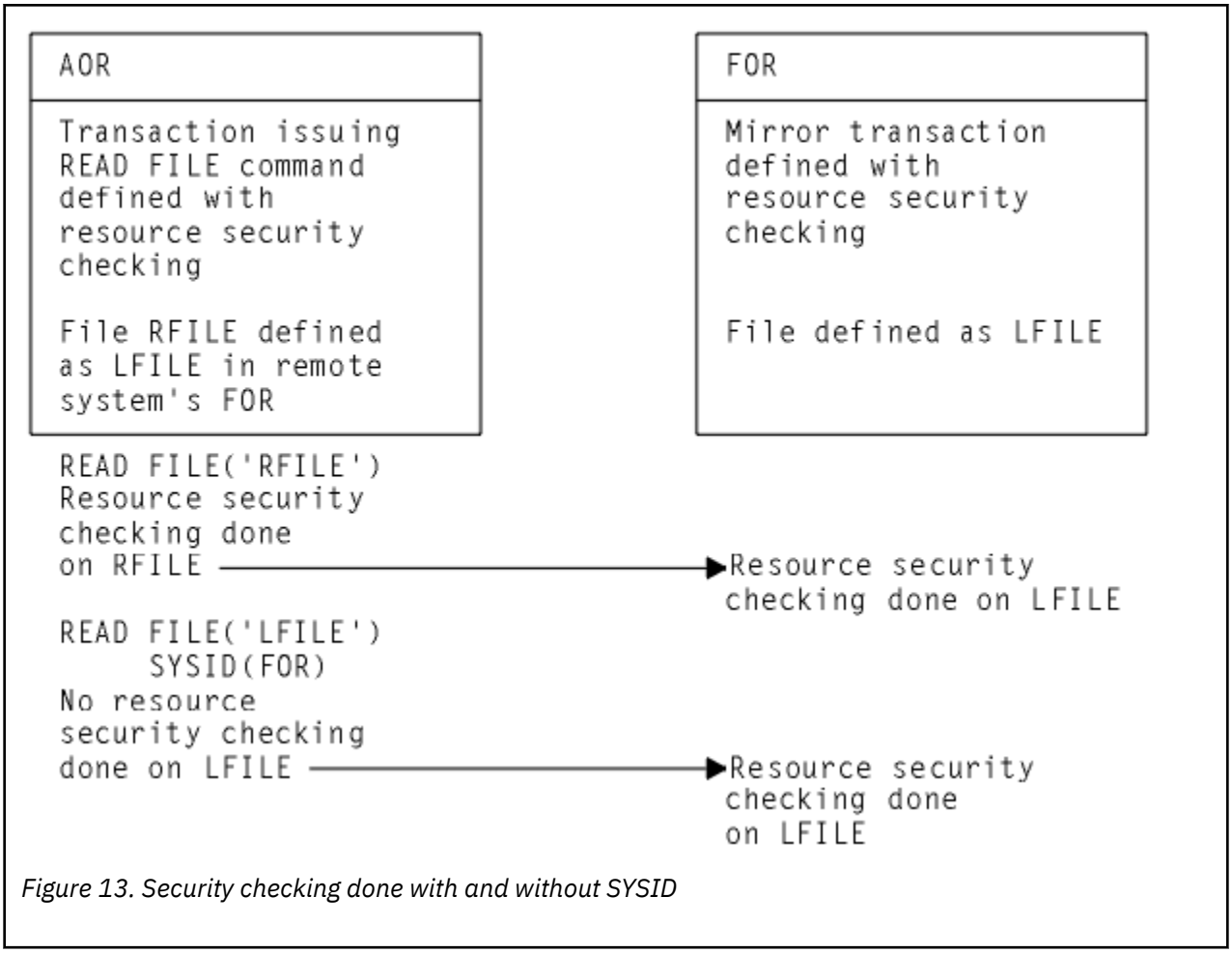


Figure 13. Security checking done with and without SYSID

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 33 on page 243 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 33 on page 243 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 33. 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

APPC password expiration management

APPC password expiration management (PEM) with CICS provides receive support for an APPC architected sign-on transaction.

Stabilization notice: Support for APPC PEM is stabilized. The PEM server does not support password phrases. To support authentication with password phrases when using CICS Transaction Gateway, you must migrate from APPC to IP interconnectivity (IPIC) and change your application code to use a current External Security Interface (ESI) API such as **`CICS_VerifyPassword`** and **`CICS_ChangePassword`** as described in the CICS Transaction Gateway for Multiplatforms product documentation.

Note: In the information about APPC PEM, *sign-on* is used in the sense defined in the APPC architecture, which is different from the meaning used elsewhere in CICS documentation.

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.

Sample program

You might find it useful to copy and modify an example program. For your guidance, a sample program is shipped in library CICSTS56.CICS.SDFHSAMP. The program is DFH$SNPW, the PEM sample program for Windows NT.

What you require to use APPC PEM

To use APPC PEM, you need a PEM client, a PEM server, and an external security manager.

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 information describes the PEM server provided by CICS Transaction Server for z/OS, Version 5 Release 6 . 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 14 on page 244 .

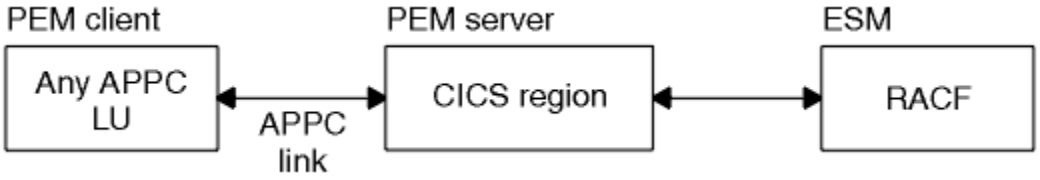


Figure 14. APPC PEM configuration

APPC PEM support for external security interface

Password expiration management provides support for 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 as follows:

CICS_VerifyPassWord
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
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.

Roles of PEM client and CICS PEM server

CICS Transaction Server for z/OS, Version 5 Release 6 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, which is an SNA service transaction program.

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 15 on page 246 .

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.

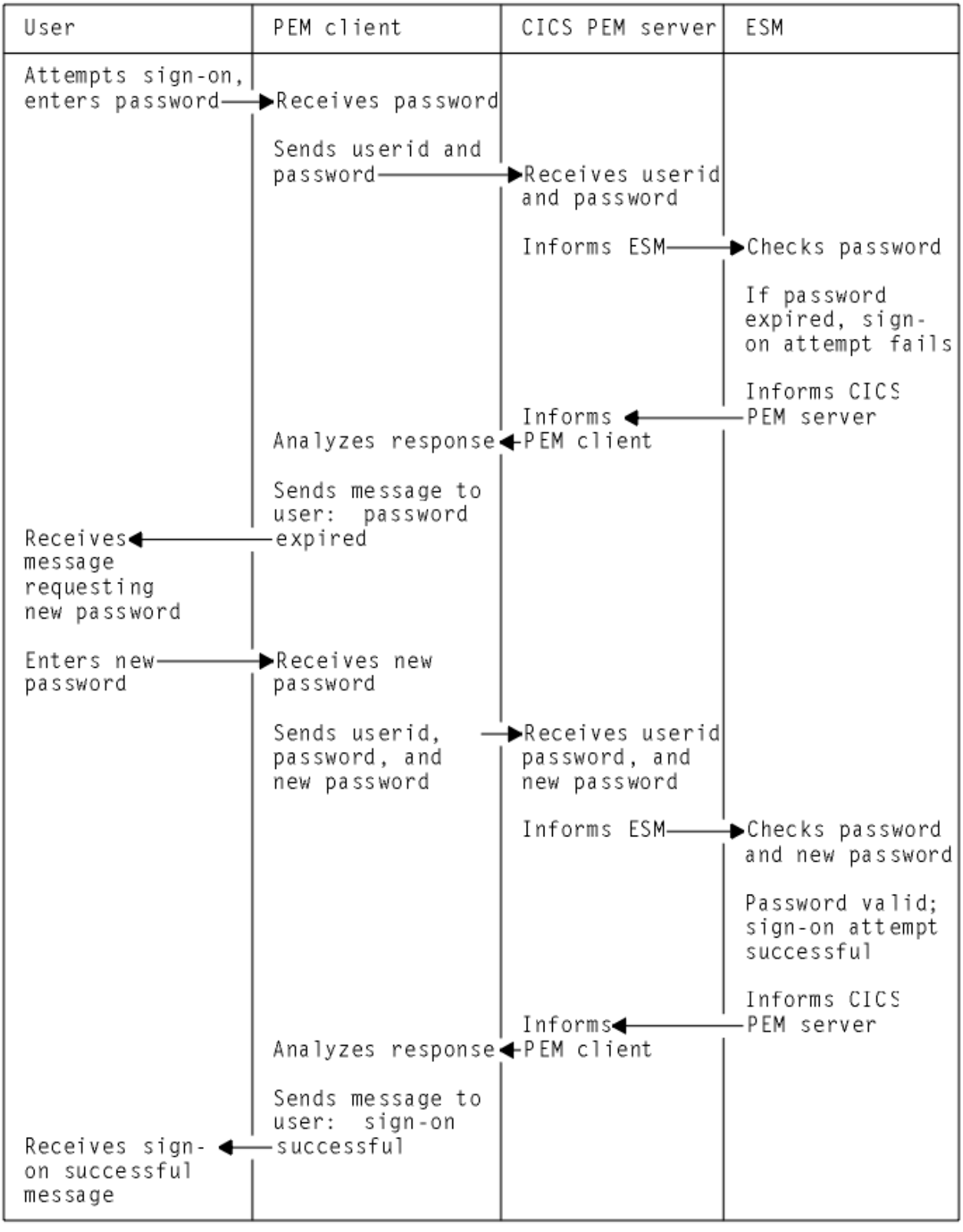


Figure 15. Example of signing on with APPC PEM

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.

   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 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 16 on page 248 through Figure 19 on page 251 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 16 on page 248 shows the expected flows between the PEM client and CICS PEM server during a successful sign-on when PV is not being used.

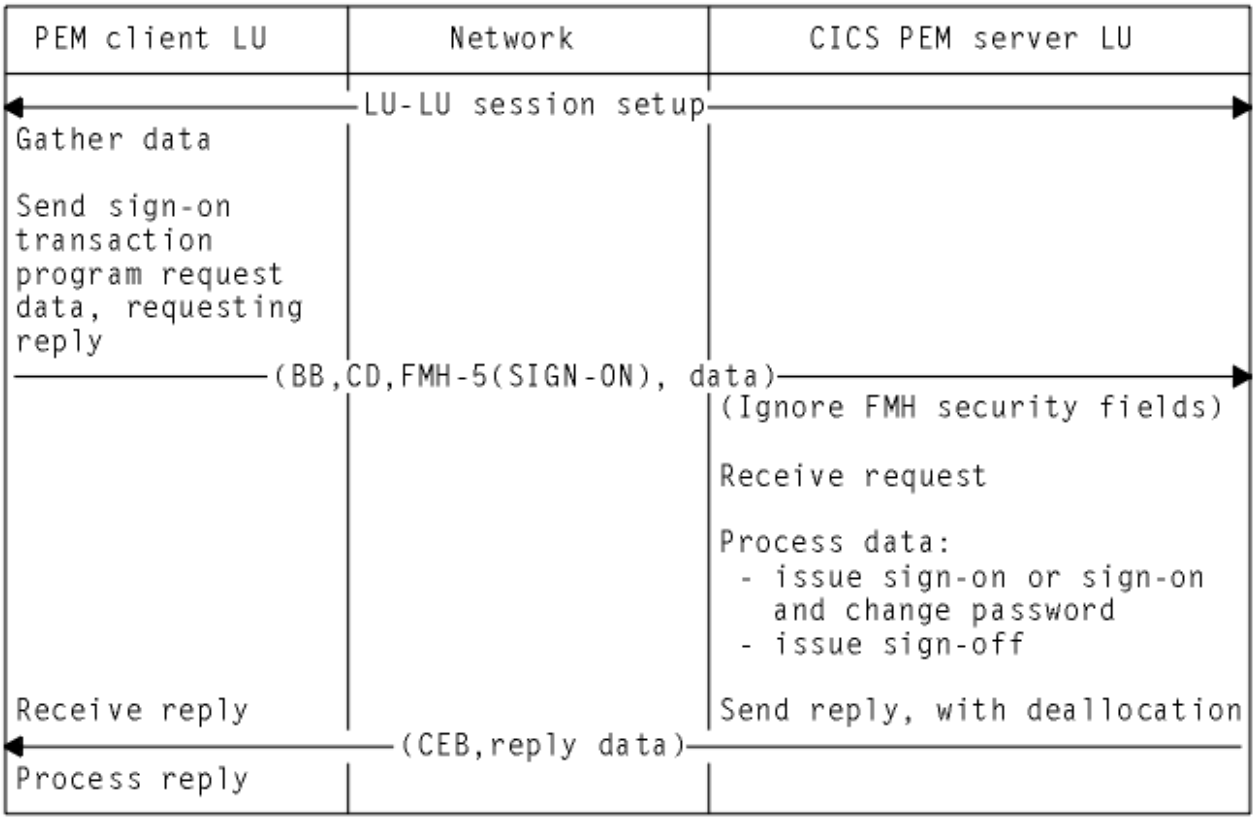


Figure 16. Expected flows for a successful sign-on with a non-PV connection

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 17 on page 249 shows the expected flows for an unsuccessful sign-on between a PEM client and CICS PEM server when PV is not being used.

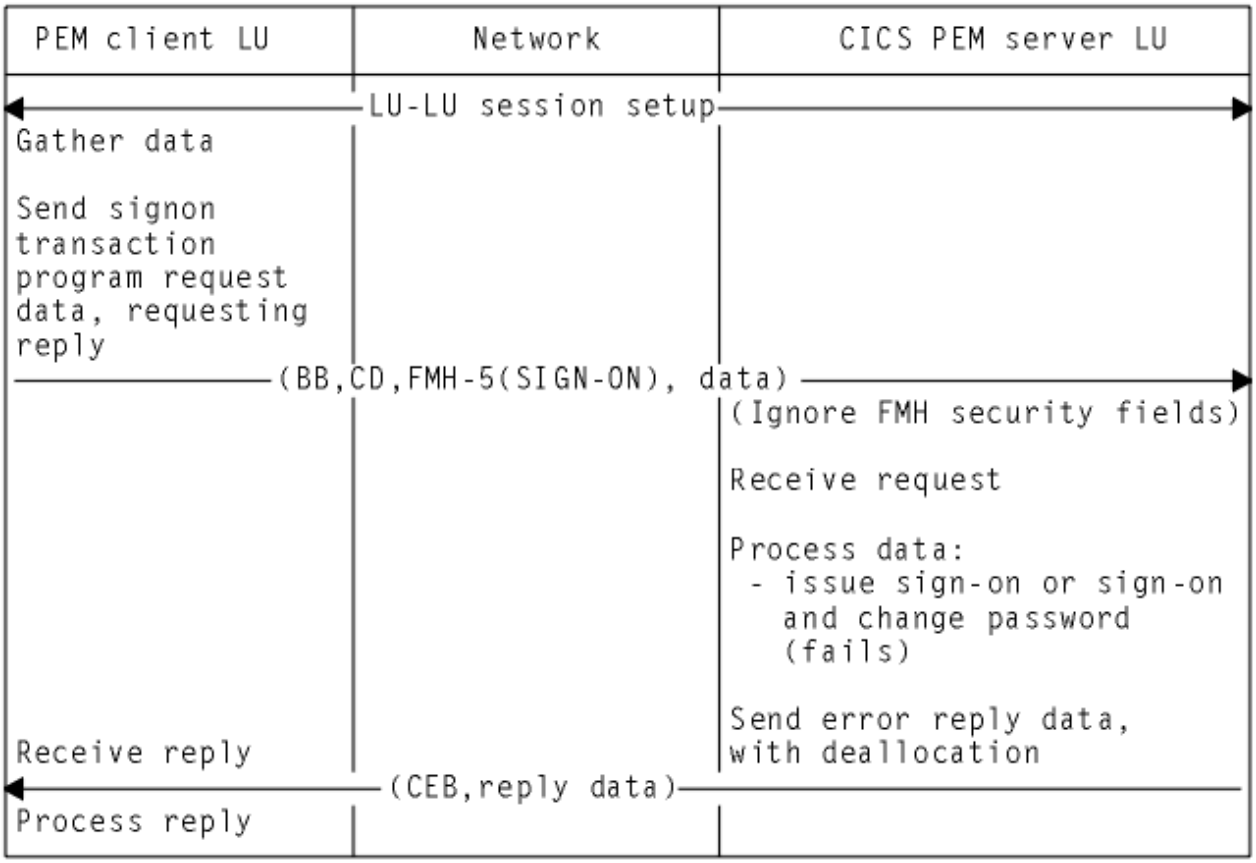


Figure 17. Unsuccessful sign-on—non-PV connection

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 18 on page 250 shows the expected flows between the PEM client and CICS PEM server during a successful sign-on on a PV connection.

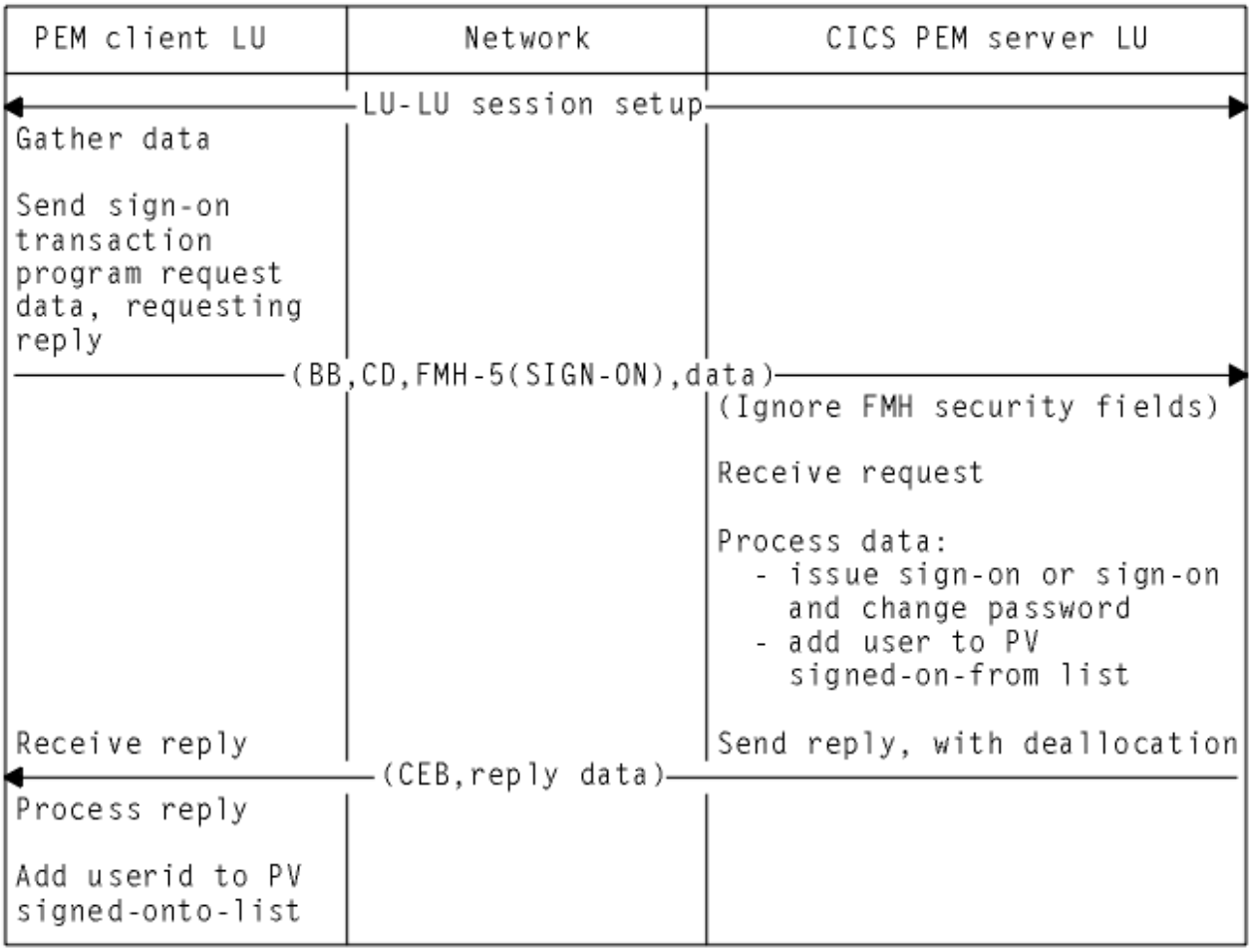


Figure 18. Successful sign-on—PV connection

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 19 on page 251 shows the expected flows between a PEM client and CICS PEM server during an unsuccessful sign-on on a PV connection.

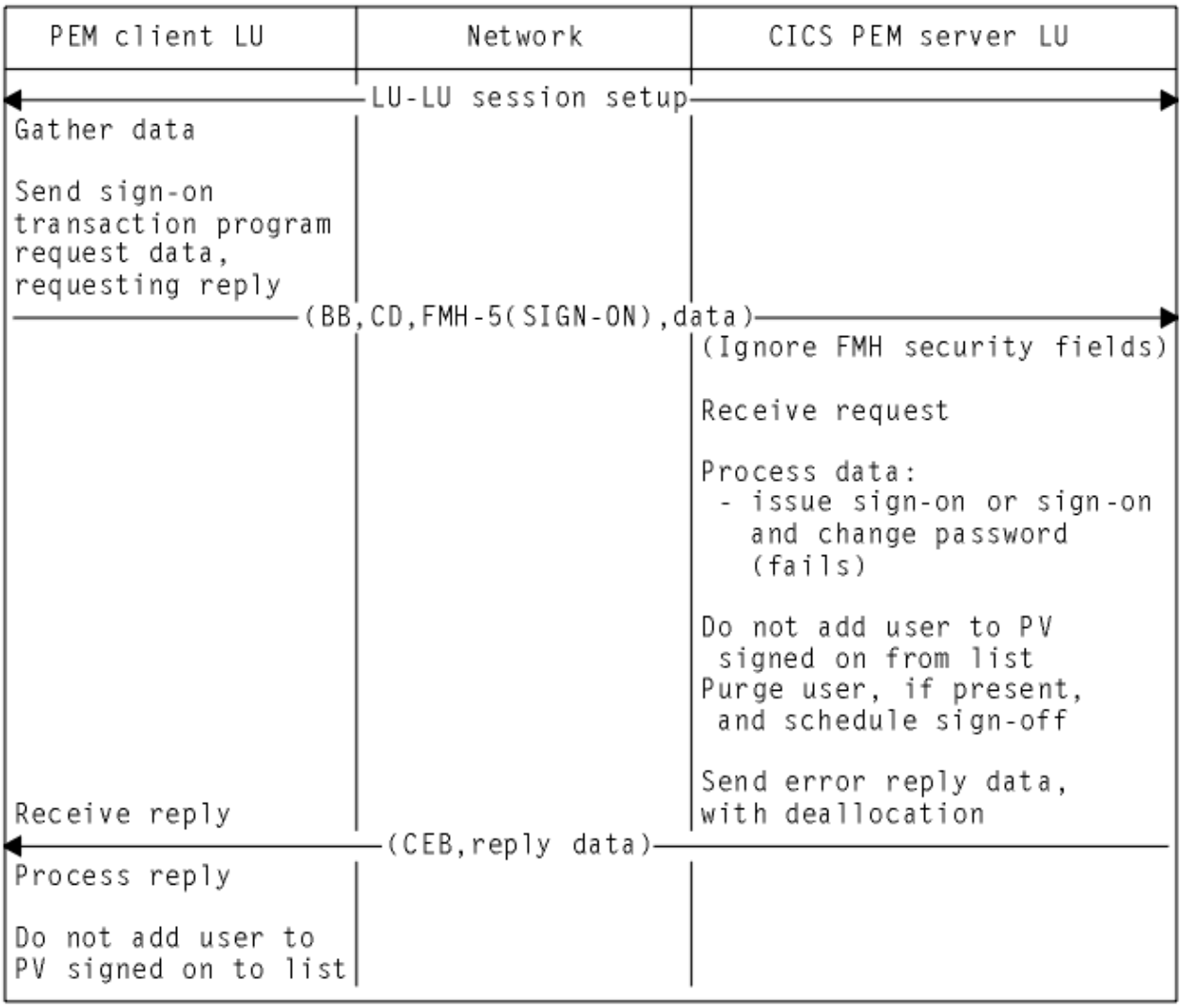


Figure 19. Unsuccessful sign-on—PV connection

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 example programs which is described in Introduction to APPC password expiration management, 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 253.
- 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 253.
- 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 253.
- 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 20 on page 253; if it does not, CICS rejects the data.

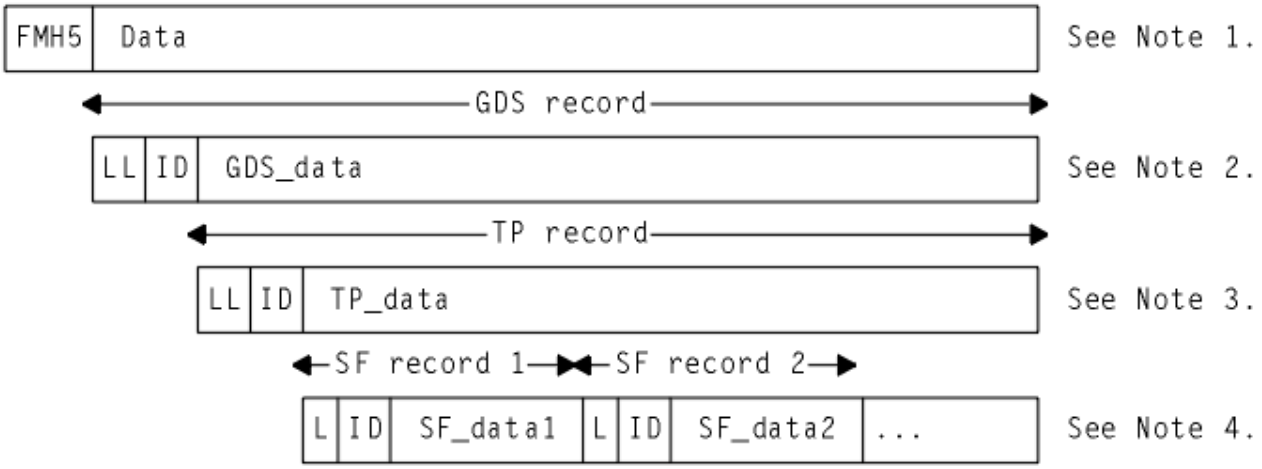


Figure 20. Format of user data

PEM client input and output data

To perform the CICS PEM server functions, the CICS PEM server takes input data from, and sends output data to, the PEM client sign-on transaction program.

The CICS PEM server functions are described in CICS PEM server processing.

- The PEM client sends data to the CICS PEM server, as described in Sign-on input data sent by PEM client.
- The CICS PEM server sends data to the PEM client, as described in Sign-on output data returned by CICS PEM server.

Ensure the data conforms to the standards described in Setting up the PEM client, and that its format is as described in Format of user data. See Examples of PEM client and CICS PEM server user data for an example 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. 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

The CICS PEM server needs input data from the PEM client sign-on transaction program.

See “Example: Sign-on with new password” on page 258 for an example of sign-on input data in GDS flows.

Table 34. 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.

Table 34. Sign-on request and data sent to CICS PEM server (continued)		
Length (bytes)	**Value**	**Meaning**
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

The CICS PEM server returns sign-on output data to the PEM client.

Table 35 on page 254 lists this data. See “Example: Response to correct sign-on data” on page 258 and “Example: Response to incorrect data format” on page 261 for examples of sign-on output data in GDS flows.

Table 35. 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.
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 37 on page 256.
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.

Table 35. Sign-on output data returned to PEM client (continued)			
Length (bytes)	**Value**	**Required or optional**	**Meaning**
2	X'0000' through X'0003', X'0005' through X'0007', X'000F'	Optional	sign-on request formatting error. See Table 38 on page 257.
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 36 on page 256 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 36 on page 256 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.
1	X'04'	Optional	Data ID of subfield for date and time password will expire.
8	See Table 36 on page 256 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

This topic lists the format of the date and time subfields that the CICS PEM server can return to the PEM client.

Table 36 on page 256 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 35 on page 254. See "Example: Response to correct sign-on data" on page 258 for an example of date and time subfields in a GDS flow.

Table 36. 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

This topic describes the sign-on completion status values that the CICS PEM server can return to the PEM client in the status completion subfield in the sign-on reply data.

Table 37 on page 256 describes the sign-on completion status values (referred to in Table 35 on page 254) that the CICS PEM server can return to the PEM client in the status completion subfield in the sign-on reply data. See “Example: Response to correct sign-on data” on page 258 for an example of sign-on completion status values in a GDS flow.

Table 37. 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 38 on page 257.

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

This topic lists the sign-on request formatting error values that the CICS PEM server can return to the PEM client.

Table 38 on page 257 lists the sign-on request formatting error values (referred to in Table 35 on page 254) that the CICS PEM server can return to the PEM client. Each is a 2-byte binary value. See "Example: Response to incorrect data format" on page 261 for an example of sign-on request formatting errors in a GDS flow.

Table 38. 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 by using GDS variables.

You can use the following examples to help you check the data being sent by your PEM client:

- "Example: Sign-on with new password" on page 258
- "Example: Response to correct sign-on data" on page 258
- "Example: Response to incorrect data format" on page 261

These examples are produced by the sample PEM client program shown in the library CICSTS56.CICS.SDFHSAMP. This program uses the following values:

partner_LU_alias

```
hostcics
```

LU_alias
ps2lua

mode_name
lu62ss

When you write your own PEM client program, use the values defined in your communications manager configuration.

Example: 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 34 on page 253:

0023
Length of entire GDS variable, including this 2-byte length value

1221
Data ID for sign-on

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

Example: Response to correct sign-on data

Figure 21 on page 259 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 21. 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 35 on page 254.

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 37 on page 256 for a list of sign-on completion status values.)

In Figure 21 on page 259, the last four lines of hexadecimal user data returned to the PEM client show the following optional values, as described in Table 35 on page 254. (Note that the formatting error subfields shown in Table 35 on page 254 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 36 on page 256:

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 36 on page 256:

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 36 on page 256:

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.)

Example: Response to incorrect data format

Figure 22 on page 261 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 22. 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 35 on page 254:

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 37 on page 256 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 35 on page 254.)

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 38 on page 257 for a description of other possible formatting errors).

Implementing IPIC security

These topics tell you how to implement security for IPIC connections.

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(*value*). The CIPHERS attribute can be specified in either of two ways:
    - A string of up to 56 hexadecimal digits that is interpreted as a list of up to 28 2-digit cipher suite codes.
    - The name of the SSL cipher suite specification file, which is a z/OS UNIX file in the `/security/ciphers` subdirectory of the directory that is specified by the **USSCONFIG** system initialization parameter. For more information, see Cipher suites and cipher suite specification files.

    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 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 resources definitions on both the local and remote regions specify:
  - PROTOCOL(IPIC).
  - SSL(CLIENTAUTH) or SSL(YES).
  - CIPHERS(*value*).
  - 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 resource must refer to a TCPIPSERVICE resource 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 the SECURITYNAME attribute is used to establish link security.

The default value is LINKAUTH(SECUSER)

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.

How the task user ID associated with the user request is determined

The user request will run under the task user ID that is shown in Table 39 on page 263, depending on the setting of the link user ID and the USERAUTH option. In some cases, there is a secondary user ID associate with the task. Security checks are also run against the secondary user ID.

In Table 39 on page 263:

- *link_user* is either the SECURITYNAME if LINKAUTH(SECUSER) is used, or the user ID associated with the certificate if LINKAUTH(CERTUSER) is used.
- *remote_user* is the user ID from the remote system in the message. For connections between CICS regions, this is the user ID of the remote CICS task.

Table 39. The task user ID associated with the user request

Link user ID	USERAUTH	Task user ID	Second user ID
link_user	LOCAL	*link_user*	
link_user	IDENTIFY/VERIFY	*remote_user*	*link_user*
link_user	DEFAULTUSER	Default user [1]	*link_user*
link_user = region user ID	LOCAL	Default user [1]	
link_user = region user ID	IDENTIFY/VERIFY	*remote_user*	

Table 39. The task user ID associated with the user request (continued)			
Link user ID	**USERAUTH**	**Task user ID**	**Second user ID**
link_user = region user ID	DEFAULTUSER	Default user [1]	

Note:

1. Use the default user only if the user task requires no authority.

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 of the received security context that flows from the requestor and under which the transaction operates. The received security context usually contains the user ID of the user running the client.

The security context might identify one of the following types of user:

- A user signed onto a terminal in a TOR
- A user ID of the transaction issuing the DPL request
- A client user of an ECI request

For IPIC connections, the USERAUTH attribute on the IPCONN resource definition 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 equivalent attribute for MRO and SNA connections is the ATTACHSEC attribute. For more information about the ATTACHSEC attribute and specifying user security with other intercommunications methods, see "Specifying user security in link definitions" on page 230.

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 ID. Enter IDENTIFY when the connecting system is trusted to pre-authenticate users, for example, if it is another CICS or CICS TG system.

SSL client authentication must be in use or the connecting system must be in the same sysplex.

VERIFY
Incoming attach requests must specify a user ID and 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.

Note: CICS uses password verification to verify a user ID during the processes described here. CICS enforces a full verification request once a day for each user ID that is used to log on to the CICS region. The full verification request using the RACROUTE REQUEST=VERIFY macro makes RACF record the date and time of last access for the user ID, and write user statistics.

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 four-character cipher suites that do not perform encryption, such as 003B, C001, C006, C00B, and C010, on the CIPHERS attribute of the TCPIPSERVICE definition. For a complete listing of cipher suite definitions supported by z/OS, see Cipher suite definitions in z/OS Cryptographic Services product documentation. To use any of these cipher suite definitions in CICS, you must set up an SSL cipher suite specification file as described in Creating an SSL cipher suite specification file.

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 Communications Server: IP Configuration Guide. For further information on the **NETACCESS** configuration parameter, see the z/OS Communications Server: IP Configuration Reference.

Remote user sign-on status with IPIC

If the IPCONN resource specifies USERAUTH(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 one of the following events occurs:

- The period specified in the system initialization parameter, USRDELAY system initialization parameter, elapses after completion of the last transaction associated with the attach request for this user.
- RACF notifies CICS with a RACF Event Notification (ENF) that the profile has been changed. For more information, see “Changing the RACF profile of a remote user” on page 14.

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 one of the following situations occurs:

- The transaction performs a syncpoint.
- The attach request ends.
- Sign off occurs because RACF notifies CICS of changes to a user profile, and an attached request associated with that signed-on user ID completes, for all operands of ATTACHSEC except LOCAL.
- Sign off occurs because RACF notifies CICS of changes to a user profile, a new attach request is made, and the value in the **USRDELAY** system initialization parameter has not expired. This sign off is followed by a sign on.

IPIC transaction, resource, and command security

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 "Transaction security" on page 69.

In an IPIC environment, two basic security requirements must be met before a transaction can be initiated:

- The link user ID must have sufficient authority to initiate the transaction (see "IPIC link security" on page 263).
- 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 23 on page 266.

```
CEDA DEFINE TRANSACTION
  .
  RESSEC(YES)
  CMDSEC(YES)
  .
```

Figure 23. Specifying resource and command security for transactions

If a TRANSACTION definition specifies resource security checking, using RESSEC(YES), 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 "Resource security" on page 73 and "CICS command security" on page 118.

CICS routing transaction, CRTE

You can use the CICS routing transaction, CRTE, with IPIC, LU6.2, or MRO links to run transactions that are 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 are on all systems.

Ensure that the terminal through which CRTE is started is defined on the remote system or is defined as shippable in the local system. The terminal operator needs RACF authority if the remote system is protected.

Security checking in the AOR for transactions that run under CRTE does not depend on what is specified on ATTACHSEC (for MRO and LU6.2 links) or USERAUTH (for IPIC links), nor does it depend on the user ID signed on in the TOR. Instead, security checking depends on whether the user signs on when 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 made against the link user ID 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 user ID of the signed-on user. For transactions attempting to access resources, security checking is made against the user ID of the signed-on user in the surrogate and the link user ID.

Security checking done in AOR with IPIC

This depends on how LINKAUTH, SECURITYNAME and USERAUTH are specified in the AOR.

The link user ID shown in the tables Table 40 on page 267, Table 41 on page 267, and Table 42 on page 268 is determined from the values of LINKAUTH and SECURITYNAME on the IPCONN definition. If LINKAUTH(SECUSER) is specified, the link user ID is determined from the SECURITYNAME attribute. If LINKAUTH(CERTUSER) is specified, the link user ID is determined from the external security manager such as RACF which associates a user ID with the certificate passed by the TOR during SSL handshake. LINKAUTH(SECUSER) is the default. The default value for SECURITYNAME is the default user ID.

If the link user ID is the same as the region user ID 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 the value specified with the USERAUTH attribute for the IPCONN in the AOR:

- If USERAUTH(LOCAL) is specified, security checking is done using the link user ID only.
- If USERAUTH(DEFAULTUSER) is specified, only the default user ID for the AOR is used.
- If USERAUTH(IDENTIFY) or USERAUTH(VERIFY) is specified, the link user ID is not used. Only the user ID received from the TOR is used to determine security.

USERAUTH(LOCAL) is the default.

Neither the region user ID for the TOR, nor the link user ID associated with the TOR's IPCONN definition for the AOR, is relevant to security checking in the AOR.

The following table shows how checking is done when USERAUTH(LOCAL) is specified.

Table 40. USERAUTH(LOCAL)

Region user ID for AOR	**Link user ID**	**Checking in AOR**
USERIDA	Not specified	Check against AOR DFLTUSER
USERIDA	USERIDA	Check against AOR DFLTUSER
USERIDA	USERIDB	Check against USERIDB

The following table shows how checking is done when USERAUTH(DEFAULTUSER) is specified.

Table 41. USERAUTH(DEFAULTUSER)

Region user ID for AOR	**Link user ID**	**Checking in AOR**
USERIDA	Not specified	Check against AOR DFLTUSER
USERIDA	USERIDA	Check against AOR DFLTUSER

Table 41. USERAUTH(DEFAULTUSER) (continued)		
Region user ID for AOR	**Link user ID**	**Checking in AOR**
USERIDA	USERIDB	Check against USERIDB and AOR DFLTUSER

The following table shows how checking is done when USERAUTH(IDENTIFY) or USERAUTH(VERIFY) is specified.

Table 42. USERAUTH(IDENTIFY) and USERAUTH(VERIFY)		
Region user ID for AOR	**Link user ID**	**Checking in AOR**
USERIDA	Not specified	Transmitted user ID and AOR DFLTUSER
USERIDA	USERIDA	Transmitted user ID only
USERIDA	USERIDB	Transmitted user ID and USERIDB

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 Multiregion operation.

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 in this document.

See Figure 24 on page 270 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 5 Release 6 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 24 on page 270.

You cannot specify to CICS whether or not you want connect security checking for MRO connections—CICS always issues the RACROUTE calls.

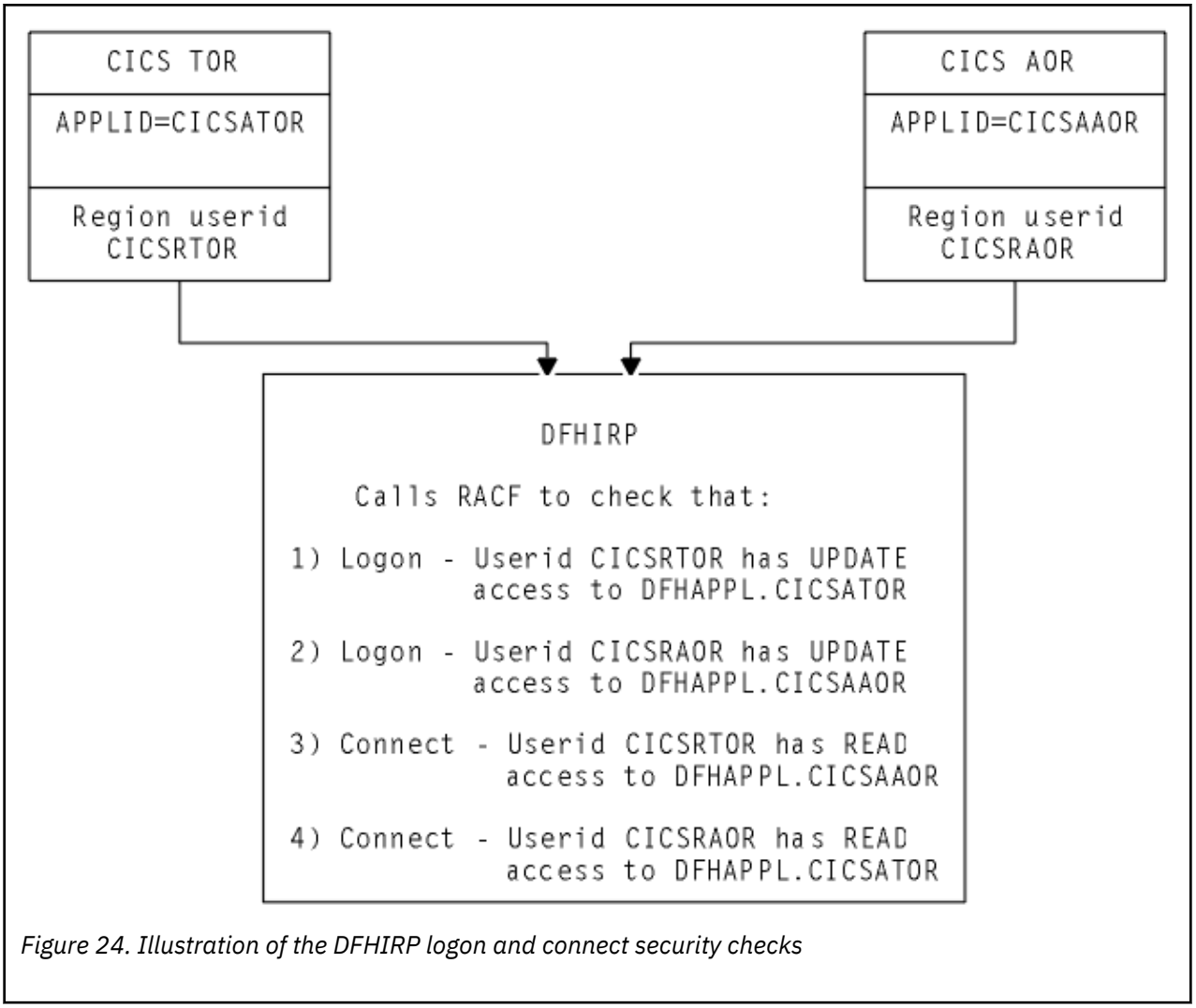


Figure 24. Illustration of the DFHIRP logon and connect security checks

The TOR and AOR shown in Figure 24 on page 270, 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 RACF facilities 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

About this task

For MRO connections, all sessions have the same link userid. If you require more information about the SESSIONS resources, see SESSIONS resources.

Procedure

- Specify the link userid in the USERID attribute of the SESSIONS resource definition. The SECURITYNAME field on the CONNECTION resource 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 271. 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” on page 272.

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 resource 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 275.

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.

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.

Remote user sign-on status with MRO

With the ATTACHSEC(IDENTIFY) parameter, 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 one of the following events occurs:

- The period specified in the USRDELAY system initialization parameter system initialization parameter elapses after completion of the last transaction associated with the attach request for this user.
- CICS is notified of certain changes in the RACF profile of a signed-on remote user, or a signed-on user who is not directly using a physical terminal or console, through a type 71 RACF Event Notification (ENF). For details, see “Changing the RACF profile of a remote user” on page 14.
- The CICS region shuts down.

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 one of the following situations occurs:

- The transaction performs a syncpoint.
- The attach request ends.
- Sign off occurs because RACF notifies CICS of changes to a user profile, and an attached request associated with that signed-on user ID completes, for all operands of ATTACHSEC except LOCAL.
- Sign off occurs because RACF notifies CICS of changes to a user profile, a new attach request is made, and the value in the **USRDELAY** system initialization parameter has not expired. This sign off is followed by a sign on.

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 43 on page 273 shows how CICS decides which userid to send.

Table 43. 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

Table 43. MRO attach-time user identifiers (continued)	
Characteristics of the local task	**User identifier sent by the TOR to the AOR**
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 in an intercommunication environment

In an intercommunication environment, the security requirements of a transaction are specified when the transaction is defined, as in a single-system environment.

See Transaction security for details.

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 in an intercommunication environment

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 25 on page 274.

```
CEDA DEFINE TRANSACTION
  .
  RESSEC(YES)
  CMDSEC(YES)
  .
```

Figure 25. 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 “Resource security” on page 73 and “CICS command security” on page 118.

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 Defining remote resources for transaction routing) 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 271.

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.

CICS routing transaction, CRTE

You can use the CICS routing transaction, CRTE, with IPIC, LU6.2, or MRO links to run transactions that are 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 are on all systems.

Ensure that the terminal through which CRTE is started is defined on the remote system or is defined as shippable in the local system. The terminal operator needs RACF authority if the remote system is protected.

Security checking in the AOR for transactions that run under CRTE does not depend on what is specified on ATTACHSEC (for MRO and LU6.2 links) or USERAUTH (for IPIC links), nor does it depend on the user ID signed on in the TOR. Instead, security checking depends on whether the user signs on when 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

made against the link user ID 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 user ID of the signed-on user. For transactions attempting to access resources, security checking is made against the user ID of the signed-on user in the surrogate and the link user ID.

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 Security for CICS-supplied transactions.

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 26 on page 276 summarizes what happens.

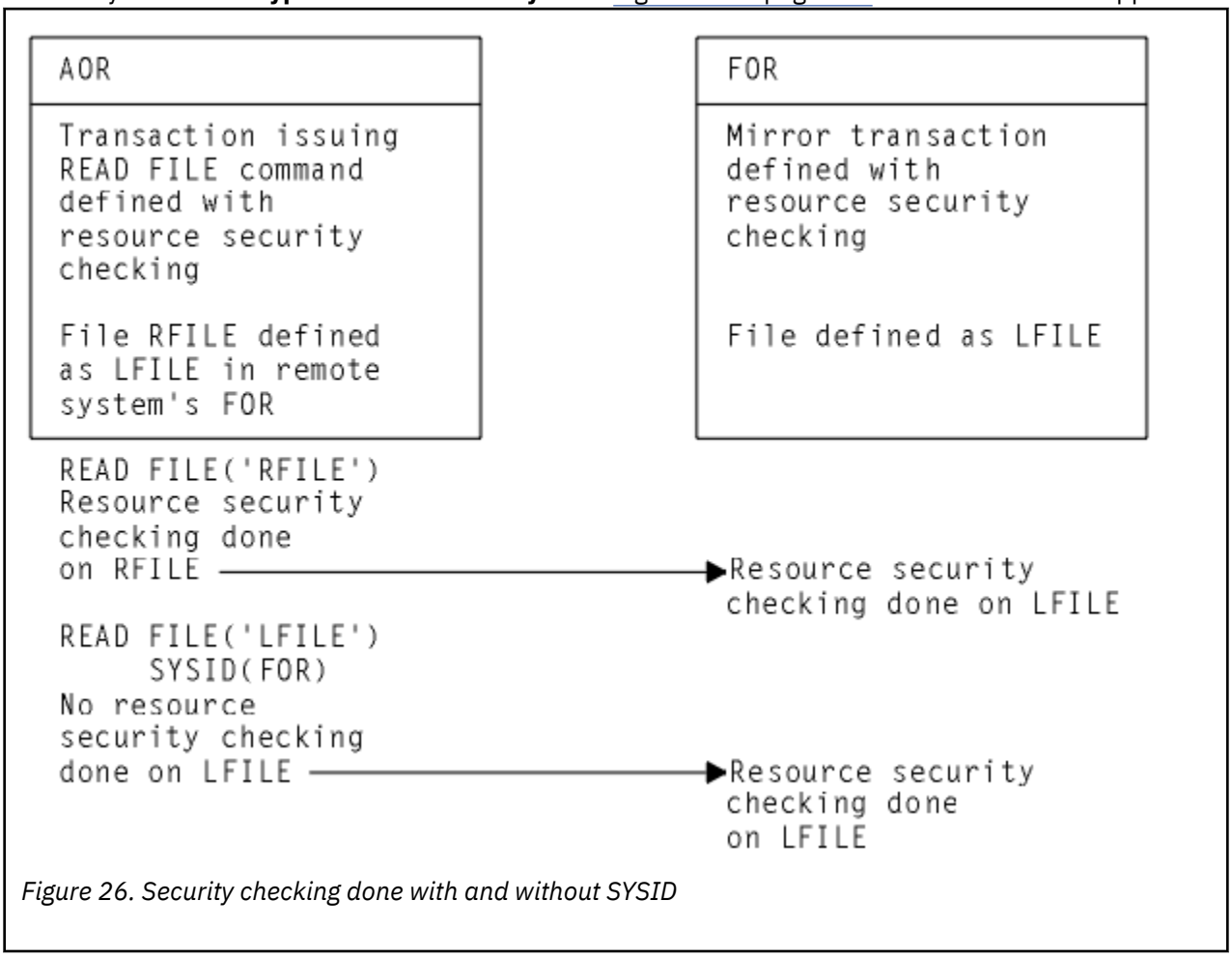


Figure 26. Security checking done with and without SYSID

For programming information on specifying the SYSID option, see READ.

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 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 user ID, which is validated if the MRO connection has been defined with ATTACHSEC(IDENTIFY).

To use the user ID parameter in the DFHXCIC call, the client program must have surrogate-user authority to the specified user ID. For more information, see "Surrogate user checking for EXCI calls" on page 116.

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 user IDs 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 277. If you define the mirror transaction with RESSEC(YES) in the server region, authorize these user IDs to access the server program that is being linked to by the mirror. If the server program accesses any CICS resources, authorize the same user IDs 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 user IDs 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 Exception conditions for LINK command. For further information about DPL, refer to Overview of DPL.

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 used is the one on the SESSIONS definition.

The region userid referred to in Table 44 on page 278 through Table 45 on page 278 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 44 on page 278 shows how checking is done in the AOR when ATTACHSEC(LOCAL) has been specified.

Table 44. Security checking done in AOR—ATTACHSEC(LOCAL) specified

Region userid for AOR	Userid in Session Definition in AOR	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 45 on page 278 shows how checking is done in the AOR when ATTACHSEC(IDENTIFY) has been specified.

Table 45. Security checking done in AOR—ATTACHSEC(IDENTIFY) specified

Region userid for AOR	Userid in session definition in AOR	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

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" on page 279 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 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 first 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 and file 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 user ID 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 (user IDs are 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 279). 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 user ID as a server can always bind to that server.
- It is impossible to control which user IDs can bind to a given APPLID without also controlling which user IDs can log on as servers for that APPLID.

SDT bind-time security uses different definitions from those employed by IPIC, 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 user IDs of the AOR and FOR are the same (undefined user IDs 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 user ID 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 272). 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 82.

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 279).

If the user ID is allowed to sign on to the FOR's application, the CONNECT request succeeds unless the AOR's user ID is not allowed to read the specified file. Otherwise, the CONNECT request is treated in the same way as when the AOR's user ID is undefined.

When file security is in force in an FOR, and the user ID of the AOR is *undefined*, a CONNECT request fails unless the FOR's default user ID (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 exists, SDT continues to allow access until something causes the connection to be broken. See Refreshing resource profiles in main storage.

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 user ID of the AOR. If you use IPIC instead of MRO for function shipping, ensure that the value of the IPCONN SECURITYNAME parameter or, if applicable, the AOR's certificate's user ID in the FOR is the same as the user ID of the AOR. Otherwise, the SDT CONNECT, and function shipping security checks are 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 this 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 279 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 further in “Authorizing a CICS region to a coupling facility data table ” on page 282.

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, the server region user ID requires ALTER access to a FACILITY class general resource profile called IXLSTR.*structure_name*.

- The server region user ID is the userid under which the job or started task is running.
- 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 coupling facility data table (CFDT) server starts up for a given coupling facility data table pool, CICS authorized cross-memory (AXM) services call RACF to establish that the server is authorized to act as a server for that pool.

To authorize a CFDT 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 the access of a CICS region to a coupling facility data table (CFDT) 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. For more information about CFDT server security parameters, see Security parameters.

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 following example shows the security parameters for a pool called PRODCFT1:

```
POOLNAME=PRODCFT1
SECURITY=YES
```

```
SECURITYCLASS=FCICSFCT
SECURITYPREFIX=NO
```

For example, if the pool is PRODCFT1, define the profile and the required PERMIT statement as follows:

```
RDEFINE FCICSFCT PRODCFT1 UACC(NONE)
PERMIT PRODCFT1 CLASS(FCICSFCT) ID(region_userid) ACCESS(UPDATE)
```

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 Coupling facility data table server parameters.

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 82 for details of CICS file security.

Chapter 8. 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.

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 288 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 292 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.

Related concepts

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.

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 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.

Related concepts

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.

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

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.

Related concepts

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.

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.

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

Identification is the process by which the identity of a user is established.

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 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.
- 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

Identification is the process by which the identity of a user is established. This is how a user's identity is established for the HTTP application protocol.

About this task

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 user ID 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 303 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.

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. 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. If authentication is required for the connection but the client does not provide an authenticated user ID, the request is rejected.
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.

Note: CICS uses password verification to verify a user ID during the processes described here. CICS enforces a full verification request once a day for each user ID that is used to log on to the CICS region. The full verification request using the RACROUTE REQUEST=VERIFY macro makes RACF record the date and time of last access for the user ID, and write user statistics.

The method used to identify the user is determined by the AUTHENTICATE and SSL attributes of the TCPIPSERVICE definition:

Table 46. 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, or ATTLSAWARE	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	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 and can be overridden by an analyzer program.

Table 46. How the user of an HTTP client is identified (continued)		
AUTHENTICATE	**SSL**	**How the user is identified**
CERTIFICATE	CLIENTAUTH, or ATTLSAWARE	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.
AUTOREGISTER	CLIENTAUTH, or ATTLSAWARE	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, or ATTLSAWARE	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 that are invalid, and that 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.

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

Identification is the process by which the identity of a user is established. This is how a user's identity is established for the IPIC protocol.

For the IPIC protocol you can use basic authentication to identify the user. Specify USERAUTH(VERIFY) or USERAUTH(IDENTIFY) in 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 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 a user.

You can use basic authentication with the HTTP, ECI and IPIC 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 a user.

You can use SSL client certificate authentication with the HTTP and IPIC application protocols.

CICS uses password verification to verify a user ID during the processes described here.

CICS enforces a full verification request at the first time each day that a user ID is used to log on to the CICS region or is verified through a **VERIFY PASSWORD** or **VERIFY PRASE** command. The full verification request records the date and time of last access for the user ID, and writes user statistics. A full verification is also made if an incorrect password or password phrase is entered, and in the next successful request. In other cases, the command uses a fastpath method to verify the password or password phrase. For details of the SAF interfaces used, see CICS security control points.

Authenticating HTTP users

You can use HTTP basic authentication or SSL client certificate authentication to authenticate HTTP users.

About this task

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, YES, CLIENTAUTH, or ATTLSAWARE	
HTTP with basic authentication	BASIC	NO, YES, CLIENTAUTH, or ATTLSAWARE	
HTTP with basic authentication	AUTOMATIC	NO, YES, CLIENTAUTH, or ATTLSAWARE	If SSL(CLIENTAUTH | ATTLSAWARE) is specified, and the client sends a certificate, then SSL client certificate authentication is used.

Authentication scheme	AUTHENTICATE	SSL	Notes
HTTP with SSL client certificate authentication	CERTIFICATE or AUTOREGISTER	CLIENTAUTH or ATTLSAWARE	If the client does not send a certificate, the connection is not established.
HTTP with SSL client certificate authentication	AUTOMATIC	CLIENTAUTH or ATTLSAWARE	If the client does not send a certificate, then basic authentication is used.

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

You can use basic authentication or SSL client certificate authentication to authenticate IPIC users.

For IPIC connections, you can use basic user authentication. Specify USERAUTH(VERIFY) in the IPCONN resource definition. Specify USERAUTH(LOCAL) if you do not want to authenticate the user.

Support for security protocols

CICS supports the Secure Sockets Layer and Transport Layer Security protocols.

Specifically, CICS supports TLS 1.0, TLS 1.1, and TLS 1.2. For more details of these protocols, see the relevant RFC:

TLS 1.0: RFC 2246
TLS 1.1: RFC 4346
TLS 1.2: RFC 5246

Note: The term SSL is used to refer to both the Secure Sockets Layer and Transport Layer Security protocols in the documentation, except where a specific point about either protocol is required.

The main features of these security protocols are:

Privacy
The data to be exchanged between the client and the server is encrypted. See "SSL encryption" on page 292 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 293 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.

Many different algorithms can be used for encrypting data, and for computing the message authentication code. Some algorithms provide high levels of security but require a large amount of computation for

encryption and decryption. Other algorithms are less secure but provide rapid encryption and decryption. The length of the key that is used for encryption affects the level of security; the longer the key, the more secure the data. SSL defines cipher suites to specify cryptographic algorithms that are used during an SSL connection.

SSL Encryption techniques

SSL uses two encryption techniques:

- 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 certificates; 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 312.

Cipher suites and cipher suite specification files

Many different algorithms can be used for encrypting data, and for computing the message authentication code. To allow users to select the level of security that suits their needs, and to enable communication with others who might have different needs, SSL defines *cipher suites*, or sets of ciphers. You can specify a list of cipher suites to be used during an SSL connection in the SSL cipher suite specification file.

Cipher suites

When an SSL connection is established, during the SSL handshake, the client and server exchange information about which TLS protocols and cipher suites they have in common. They then communicate using the protocol and common cipher suite that offers the highest level of security. If they do not have a protocol or cipher suite in common, then secure communication is not possible and CICS closes the connection.

The ciphers suites that are available depend on the value of the system initialization parameters **MINTLSLEVEL** and **NISTSP800131A**, and on what ciphers are supported by z/OS® System SSL. In addition, you can restrict the ciphers that are used by editing the list of cipher suites in the CIPHERS attribute on the appropriate resource definition, or by editing the SSL cipher suite specification file for the resource definition.

You can check which cipher suites are being selected for SSL inbound connections from each CICS region. The performance data field SOCIPHER (320) in the DFHSOCK group shows the code for the cipher suite that was used for each SSL inbound connection. Use this information to identify any cipher suites that are offered by the CICS region but are not being selected for SSL connections. You can also identify any less efficient or less secure cipher suites that are being selected for SSL connections. Then you can decide whether to eliminate such cipher suites.

To specify the level of encryption required:

For inbound HTTP
Use the CIPHERS attribute of the TCPIPSERVICE resource definition.

For outbound HTTP and web service requests
Use the CIPHERS attribute of the URIMAP resource definition.

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 codes.

The cipher suites that are supported by z/OS and CICS for each supported security protocol are described in Cipher Suite Definitions in z/OS Cryptographic Services System SSL Programming.

SSL cipher suite specification files

The SSL cipher suite specification file is an XML file that contains a list of cipher suites that can be used in an SSL connection.

The name of the file can be up to 28 characters in length including the extension, which must be `.xml`. The specified value must be a valid name for a UNIX file and can contain only the characters `A-Z a-z 0-9 # - . @ _`. It is case-sensitive.

The SSL cipher suite specification file must be in the *ussconfig*`/security/ciphers` directory, where *ussconfig* is the value of the SIT parameter **USSCONFIG**.

The CICS region must have permission to access z/OS UNIX, and it must have read and execute access to the directory that contains the file, and read access to the file itself.

Sample files are provided in the *usshome*`/security/ciphers` directory, where *usshome* is the value of the SIT parameter **USSHOME**. A schema file is also supplied, in the *usshome*`/schemas/security` directory. The file name is `ciphersfile.xsd`.

Structure of the SSL cipher suite specification file

Each cipher suite is specified as a **number** attribute of a **cipher** element. The cipher number is a four-character code. If you use a two-character code, pad it with leading zeros.

The sample files also contain a comment for each cipher, which contains a text string that describes the cipher suite. However, CICS does not validate this element nor take any action on it.

The following example shows the structure of the cipher file:

```
<?xml version="1.0"?>
<cipher_list xmlns="http://www.ibm.com/software/htp/cics/ciphers">
```

```
    <cipher number="000A">
        <!-- SSL_RSA_WITH_3DES_EDE_CBC_SHA -->
    </cipher>
    <cipher number="000D">
        <!-- SSL_DH_DSS_WITH_3DES_EDE_CBC_SHA -->
    </cipher>
    ...
</cipher_list>
```

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 for SSL sessions between clients and CICS. Reusing these session IDs allows CICS to perform partial handshakes with clients that it has previously authenticated. The SSL cache can be local to a CICS regions or shared between CICS regions on a sysplex. This is configured by the system initialization parameter **SSLCACHE**. For optimal performance, it is important that you select the correct option.

Local caching, SSLCACHE=CICS

In a local CICS region, by default, the SSL cache is stored in the enclave for the S8 TCBs. It is managed by z/OS System SSL as part of the SSL environment, which exists within this enclave.

When you issue the **PERFORM SSL REBUILD** command for the CICS region, a new cache is created. The new cache is populated by new SSL sessions that are established in the CICS region. The old cache is removed when the last connection using it is dropped.

If you use the **SSLCACHE=CICS** option and use port sharing to enable HTTP connection requests to the same host and port to resolve to different CICS regions, a full SSL handshake is required every time a connection request from a client resolves to a different region, and the benefits of the caching are lost.

Sysplex caching, SSLCACHE=SYSPLEX

Sharing SSL session IDs across different CICS regions on a sysplex is particularly useful when HTTP requests are being routed across a set of CICS regions by using TCP/IP connection workload balancing techniques, such as TCP/IP port sharing or Sysplex Distributor. You should use sysplex caching if you have multiple CICS socket-owning regions that accept SSL connections at the same IP address. If appropriate for your CICS systems, using sysplex caching can significantly reduce the number of full SSL handshakes.

To enable sysplex caching, activate the z/OS System SSL started task GSKSRVR and specify the system initialization parameter **SSLCACHE=SYSPLEX** for the CICS regions. For details of the SSL started task GSKSRVR and its configuration, see The SSL started task GSKSRVR.

To use the sysplex session cache, each system in the sysplex must be using the same external security manager, and a user ID on one system in the sysplex must represent the same user on all other systems in the sysplex.

The **PERFORM SSL REBUILD** command does not affect the sysplex cache.

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. All CICS processing for an SSL connection occurs on the S8 TCB. The web server HTTP attach task (CWXN by default) will stay on the S8 TCB until all the data have been sent or received. If the messages being sent or received are very large, the task could hit the runaway limit and be terminated. To avoid the task being abended, you might need to increase the transaction **RUNAWAY** value for the web server HTTP attach transaction (CWXN by default), the web server alias transactions, and any transactions that issue the web client API commands SEND, RECEIVE, and CONVERSE.

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.

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

Before you begin to configure CICS, decide which type of certificates to use in SSL handshakes.

About this task

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.

Procedure

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

- If the client certificate is not associated with a valid RACF userid, the signing certificates for any client certificates owned by clients with which you expect CICS to communicate using client authentication should be added to the keyring.

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.

   You can also specify the level of security for an SSL connection by using the **CIPHERS** attribute. You can specify this attribute either with a string that is interpreted as a list of cipher suite codes or with the name of a cipher suite specification file.

Example

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.

Setting up profiles in RACF

To build a RACF key ring in the RACF database that is suitable for use in a CICS region, you must grant access to the appropriate profiles in the FACILITY class.

About this task

You must grant this access only to users who administer CICS systems and not to general CICS users. The following profiles are available:

CONTROL

- IRR.DIGTCERT.GENCERT (to allow certificates to be signed by a CERTAUTH certificate)
- IRR.DIGTCERT.ADD on first execution (to allow a CERTAUTH certificate to be generated)
- IRR.DIGTCERT.CONNECT to connect CERTAUTH certificates for other users

UPDATE

- IRR.DIGTCERT.CONNECT (to connect CERTAUTH certificates to your keyring).
- IRR.DIGTCERT.* (to manage certificates for other users).

READ
IRR.DIGTCERT.* (to manage certificates for your own user ID).

IRR.DIGTCERT.* contains the wildcard asterisk, and is intended as a generic profile. To allow generic profiles to be created in the FACILITY class:

Procedure

1. Issue the command **SETROPTS GENERIC(FACILITY)**.
2. Issue the following command:

```
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. To permit a user ID or group *ringuser* to use the commands contained in DFH$RING 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)
```

   DFH$RING is the sample that is provided with CICS to help you set up a suitable key ring.

5. You must give the first user of DFH$RING, *certauser*, authority to create a certificate authority certificate:

```
  PERMIT IRR.DIGTCERT.ADD CLASS(FACILITY) ID(certauser) ACCESS(CONTROL)
```

   This certificate is then used to sign all the other certificates created by DFH$RING.

Results

If you have READ access to the IRR.DIGTCERT.ADD profile in the FACILITY class, you can add certificate information for your own user ID. If you have UPDATE access to the IRR.DIGTCERT.ADD profile in the FACILITY class, you can add certificate information for other user IDs . If you have RACF SPECIAL authority, you can execute RACDCERT ADD for any user ID.

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.

Before you begin

You must have authorization to use the **RACDCERT** command. This command installs and maintains digital certificates, key rings, and digital certificate mappings in RACF.

About this task

RACF supplies certificates for various certificate authorities, so you do not have to define them yourself. These certificates are listed in Supplied digital certificates in z/OS Security Server RACF Security Administrator's Guide.

Procedure

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.
6. Replace the self-signed certificate with your new CA-signed certificate:

```
RACDCERT ID(cics-region-userid) ADD('response.dataset') TRUST
```

What to do next

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 begin

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.

About this task

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$RING sample program, see Building a key ring with certificates using DFH$RING.

To create a key ring manually, follow these steps:

Procedure

Issue the following **RACDCERT** command:

```
RACDCERT ID(cics-region-userid) ADDRING(ringname)
```

The key ring must be associated with the CICS region user ID.

Results

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.

What to do next

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.

Before you begin

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.

About this task

DFH$RING is in library CICSTS56.CICS.SDFHSAMP. Edit the values in DFH$RING to create a suitable key ring and certificates:

Procedure

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     (2048                                        )"
   ```

   These values define appropriate fields in the distinguished names of the generated certificates. The country code for the *country* variable must be an ISO 3166-1 code. For a list of valid codes, see International Organization for Standardization Country Codes - ISO 3166. *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. A minimum of 2048 is recommended.

   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 'CICSTS56.CICS.SDFHSAMP(DFH$RING)' 'firstname lastname webservername  [ FORUSER(userid) ] '
```

   where *userid* is the CICS region user ID.

Results

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.

Example

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*-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*-2048-Certificate**
: This certificate can be used for CICS systems that require high-strength certificates. It can be used in CERTIFICATE attributes of TCPIPSERVICE, IPCONN, and URIMAP definitions, and EXEC CICS WEB OPEN commands.

***lastname*-Default-Certificate**
: This certificate is marked as the default certificate for the key ring and is the one that is used for all TCPIPSERVICE resources 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).

What to do next

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.

About this task

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.

Note: Multiple certificates with the same Distinguished Name on the same **KEYRING** are not supported.

Procedure

1. Create a certificate, specifying the CICS region user ID. Enter the **RACDCERT GENCERT** command as follows:

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 International Organization for Standardization Country Codes - ISO 3166. The value of *certifier* is the label of the signing certificate in the key ring.

```
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)
```

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 requests, specify the certificate in the TCPIPSERVICE resource.
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 make any updates or additions to the certificates in the key ring, issue the **PERFORM SSL REBUILD** command for the CICS region.

   The command rebuilds the SSL environment for the CICS region and refreshes the cache of certificates with the new information from the key ring.

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 web 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 288.

  Once a certificate has been registered in this way, it can be used for all inbound TCP/IP connections.
- You can use the RACDCERT command. If you do not want 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 want 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 data set 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 data set, see z/OS Security Server RACF Command Language Reference.

Using an existing certificate that is not owned by the CICS region user ID

You can share a single certificate between CICS systems by using the appropriate RACF facilities.

About this task

For any CICS resource that has the CERTIFICATE attribute and for Web Services Security, by default the certificate that is used must be owned by the CICS region user ID. If CICS needs to use a certificate that it does not own, for example a single certificate that is shared by multiple CICS systems where each system has a different region user ID, you can use the RACF Facility Class RDATALIB to allow multiple CICS systems to share a single certificate.

Procedure

1. Connect the certificate to its key ring with the PERSONAL usage option.
2. If the certificate is a USER certificate, grant to the CICS region user ID that you want to use the certificate UPDATE authority for the *ring_owner*.*ring_name*.LST resource in the RDATALIB class.
3. Activate the RDATALIB class by using the **RACLIST** command.

Results

CICS can use the certificate that is owned by the other user ID. For more information, see z/OS Security Server RACF Callable Services.

Configuring a RACF site certificate for use with CICS TS

If you want to enable SSL in your CICS Transaction Server for z/OS regions, but you do not want to define separate SSL Certificates for each region, you can use a *site certificate*.

Procedure

Build a keyring by following the instructions in “Building a key ring manually” on page 300 or “Building a key ring with certificates using DFH$RING” on page 301.

The key ring must be completely configured, that is, it must contain not only the certificate pointed to by your TCPIPSERVICE RDO definition, but also every certificate that was used to sign that certificate. These signing certificates must be in the KEYRING with USAGE=CERTAUTH.

The following figure shows an example of a completely configured key ring as listed by the RACF command: RACDCERT ID(*ring_owner*) LISTRING(*ring_name*)

```
Ring:  ring_name

 Certificate Label Name          Cert Owner  USAGE     DEFAULT
------------------------------ ----------- --------  -------
  Verisign Class 1 Primary CA     CERTAUTH    CERTAUTH  NO
```

```
IBM World Registry CA           CERTAUTH    CERTAUTH  NO
CICS-Sample-Certification       CERTAUTH    CERTAUTH  NO
Verisign Class 2 Primary CA     CERTAUTH    CERTAUTH  NO
SITECERT                        SITE        PERSONAL  YES
```

The certificate that you want to use as a site certificate (SITECERT) must be owned by SITE and have a usage of PERSONAL. This certificate is the one used by any TCPIPSERVICE definition that needs SSL encryption.

The key ring must be owned by the CICS region user ID. If multiple CICS regions use the same region user ID, they can share the same key ring. If they run under different region user IDs, then you must build separate key rings. However, you can use the same site certificate in each ring.

The site certificate must have a private key or the TCPIPSERVICE will either fail to install or will fail when an attempt is made to use it.

The CICS region user ID must have CONTROL access or greater to the profile IRR.DIGTCERT.GENCERT in the FACILITY class.

For more information, see the sections about RACF Callable Services Authorization and RACF Callable Services Usage Notes in z/OS Security Server RACF Callable Services.

Making 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.

Procedure

1. Enter the command `RACDCERT ID(userid) LIST` to find the label associated with the certificate.
2. Enter the command `RACDCERT ID (userid) ALTER(LABEL(label)) NOTRUST` to mark the certificate as untrusted.
3. If you amended the certificate while a running CICS region was using a key ring containing the certificate, issue the PERFORM SSL REBUILD command for the CICS region.

   The command rebuilds the SSL environment for the CICS region and refreshes the cache of certificates with the new information from the key ring.

   Note: The **PERFORM SSL REBUILD** command does not apply to SSL/TLS environments where CICS is using a TCPIPSERVICE that is defined with SSL(ATTLSAWARE), mandating AT-TLS secured client connections. If you want to refresh such SSL environments and cache, follow the instructions in Introduction to Application Transparent Transport Layer Security (AT-TLS).

Results

Clients are prevented from establishing CLIENTAUTH connections with this certificate.

System initialization parameters for SSL

Descriptions of system initialization parameters that relate to SSL.

The following system initialization parameters relate to SSL:

CRLPROFILE system initialization parameter
: 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 312.

KEYRING system initialization parameter
: 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 user ID. You can create an initial key ring with the DFH$RING exec in CICSTS56.CICS.SDFHSAMP.

MAXSSLTCBS system initialization parameter
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.

MINTLSLEVEL system initialization parameter
Specifies the minimum TLS protocol that CICS uses for secure TCP/IP connections.

SSLCACHE system initialization parameter
Specifies whether CICS should use a local cache of SSL sessions for 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. The cache contains session IDs that enable CICS to perform partial handshakes with clients that it has previously authenticated. A local cache is replaced when you issue the PERFORM SSL REBUILD command for the CICS region, but a sysplex cache is unaffected.

SSLDELAY system initialization parameter
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.

TCPIPSERVICE attributes for SSL

Descriptions of the attributes of the TCPIPSERVICE resource that relate to SSL.

About this task

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 protocol. The HTTP protocol supports the following authentication scheme:

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.

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

The CIPHERS attribute can be specified in either of two ways:

- A string of up to 56 hexadecimal digits that is interpreted as a list of up to 28 2-digit cipher suite codes.

- The name of the SSL cipher suite specification file, which is a z/OS UNIX file in the `security/ciphers` subdirectory of the directory that is specified by the **USSCONFIG** system initialization parameter. For example, if **USSCONFIG** is set to `/var/cicsts/dfhconfig` and **CIPHERS** is set to `strongciphers.xml`, the fully qualified file name is `/var/cicsts/dfhconfig/security/ciphers/strongciphers.xml`. For more information, see Creating an SSL cipher suite specification file.

When you use the CEDA transaction to define the resource, CICS automatically initializes the attribute with a default list of acceptable codes. For CICS to initialize the attribute, the KEYRING system initialization parameter must be specified in the CICS region where you are running CEDA. If KEYRING is not set, CICS does not initialize the attribute. The default list of 2-digit ciphers is 3538392F3233.

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.

When you use DFHCSDUP to define the resource, the CIPHERS attribute is not populated with a default value. If the resource is used to establish a secure connection and the CIPHERS attribute is empty, CICS uses the default list of 2-digit ciphers. An INQUIRE on the resource that is created in this manner, returns an empty string for the CIPHERS value.

Any unsupported ciphers are removed at run time. A list of the removed ciphers is reported in messages DFHSO0145 and DFHSO0146.

For more information, see Cipher suites and cipher suite specification files.

PORTNUMBER
Specifies the number of the port on which CICS is to listen for incoming client requests. The well known port for SSL services supported by CICS is 443, for HTTP 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.

ATTLSAWARE
CICS queries the client connection to determine whether AT-TLS is active. CICS retrieves a client certificate from TCP/IP if one was provided by the partner.

Note: If you specify SSL(ATTLSAWARE), you must also specify PROTOCL(HTTP).

Creating an SSL cipher suite specification file

You can create an SSL cipher suite specification file to specify a list of cipher suites to be used by SSL. If SSL is used for TCP/IP connections, you can specify the name of cipher suite specification file in the **CIPHERS** attribute for resources that define TCP/IP connections.

Procedure

1. Create an SSL cipher suite specification file either by editing a sample specification file or by creating your own one:
   - To modify a sample SSL cipher suite specification file, copy one of the sample files located in the *usshome*`/security/ciphers` directory to the *ussconfig*`/security/ciphers` directory, where

usshome
is the value of the SIT parameter **USSHOME**.

ussconfig
is the value of the SIT parameter **USSCONFIG**.

Note: The SSL cipher suite specification file must be in the *ussconfig*/security/ciphers directory.

- To create your own SSL cipher suite specification file, create an XML file in the *ussconfig*/security/ciphers directory and name the file according to the following rules:
  - The file name is up to 28 characters in length, including the .xml extension.
  - The file name must be a valid name for a UNIX file and contain only characters A-Z a-z 0-9 # - . @ _. It is case-sensitive.

2. Specify your list of cipher suites in the specification file as indicated in Cipher suites and cipher suite specification files.

   If you edit the sample file, you can remove unwanted cipher suites that do not meet your security requirements, or that are not supported by your hardware. You can also add cipher suites, but only those cipher suites that are supported by CICS and z/OS.
3. For the file to be effective in an SSL connection, ensure that the CICS region has permission to access z/OS UNIX, and that the region has read and execute access to the directory that contains the specification file, and read access to the file itself.

Results

You have created a cipher suite specification file. An SSL cipher suite file can be used by multiple resources. The first time when a resource that uses a specification file is installed, the file is read from zFS and parsed. Any errors are flagged during this parse. If the file is valid, the resource is installed and the cipher information is stored in a new control block that is associated with the file. When subsequent resources that use the same cipher file are installed, cached information in the control block is used.

What to do next

If you want to update the list of cipher suites in a cipher suite specification file, you can edit the file directly, but you must restart CICS for the updated list to take effect. The file is reread for any type of start, whether the **START** system initialization parameter is set to INITIAL, COLD, or AUTO.

To update the list of cipher suites for a resource without restarting CICS, you must use a new specification file:

1. Create a new cipher suite specification file. Ensure that the file name has not been loaded by this CICS system.
2. Update the existing resource definition to refer to the new file. For example, issue a **CREATE TCPIPSERVICE** command with CIPHERS(newciphers.xml) specified.
3. Reinstall the resource definition.

Customizing encryption negotiations

You can select the cipher suites that are used in the encryption negotiation process for SSL connections, to set a minimum level and a maximum level of encryption.

About this task

The CIPHERS attribute in the resource definitions TCPIPSERVICE, IPCONN, 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 codes for the cipher suites 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 negotiates with clients. You can also choose to remove cipher suites from the list. This option is useful if you want to ensure that only a very high level of encryption is used.

You can customize the list either by editing the list of codes directly or by setting the CIPHERS attribute to the name of an SSL cipher suite specification file, in which you have specified the cipher suites to be used. The SSL cipher suite specification file is a z/OS UNIX file in the `security/ciphers` subdirectory of the directory that is specified by the **USSCONFIG** system initialization parameter. For more information, see Creating an SSL cipher suite specification file

You can check which cipher suites are being selected for SSL inbound connections from each CICS region. The performance data field SOCIPHER (320) in the DFHSOCK group shows the code for the cipher suite that was used for each SSL inbound connection. Use this information to identify any cipher suites that are offered by the CICS region but are not being selected for SSL connections. You can also identify any less efficient or less secure cipher suites that are being selected for SSL connections. Then you can decide whether to eliminate such cipher suites.

For a list of cipher suites that are supported by CICS and z/OS, see Cipher suites and cipher suite specification files.

Procedure

1. Select the resource definition that you want to change.

   The CIPHERS attribute displays the default value. For CICS to display the default value, the **KEYRING** system initialization parameter must be specified in the CICS region where you are working with the resource definition.
2. Edit the attribute value to remove and reorder the cipher suites or to specify the name of the SSL cipher suite specification file.
   For example, you could specify `352F0A` or `strongciphers.xml`
3. Save the resource definition.

Example

Specifying `352F0A` means that CICS does not negotiate below 128-bit encryption for SSL connections that use this resource. Each of the 2-digit codes in the attribute, for example 35, 2F, and 0A, refers to cipher suites that have at least 128-bit encryption. CICS starts by trying to negotiate using the AES cipher suites 35 and 2F, because these cipher codes are first in the list. If the client does not have this level of encryption, CICS closes the connection.

Specifying `strongciphers.xml` means that CICS uses the ciphers listed in the cipher file located at *ussconfig*`/security/ciphers/strongciphers.xml`, where *ussconfig* is the value of the SIT parameter **USSCONFIG**.

Making your CICS TS system conformant to NIST SP800-131A

To make your system SP800-131A conformant, update various SIT parameters and resource attributes to use suitable cipher suites and certificates.

About this task

Conformance to the National Institute of Standards and Technology (NIST) SP800-131A security standard strengthens security by requiring the use of stronger cryptographic keys and more robust algorithms.

For full details of the NIST standards, see the NIST Computer Security Resource Center (nist.gov).

Procedure

To make your system conformant to NIST SP800-131A, complete the following steps:

1. Set the **NISTSP800131A** system initialization parameter to `NISTSP800131A=CHECK`.

2. Set the **MINTLSLEVEL** system initialization parameter to TLS12.
3. Set the **KEYRING** system initialization parameter to the name of a key ring that is populated with NIST SP800-131A conformant certificates.
4. Set the **USSCONFIG** system initialization parameter to the name and path of the root directory for CICS Transaction Server configuration files on z/OS UNIX.

   This directory must have a `/security/ciphers/` subdirectory that contains at least one SSL cipher suite specification file. For more information, see "Cipher suites and cipher suite specification files" on page 294.
5. Update any TCPIPSERVICE, IPCONN, or URIMAP definitions, setting the **CIPHERS** attribute to the name of an SSL cipher suite specification file that contains SP800-131A conformant cipher suites.

   The sample file `fipsciphers.xml` is suitable.
6. Update any TCPIPSERVICE, IPCONN, or URIMAP definitions, setting the **CERTIFICATE** attribute to the name of an SP800-131A conformant certificate label.

   Also, any outbound HTTP application that uses SSL must also use an SP800-131A conformant certificate on any **EXEC CICS WEB OPEN** command. If you use the key ring default certificate with any of these resources definitions or the **WEB OPEN** command, ensure that the key ring default certificate is SP800-131A conformant.
7. If you use the CICSPlex SM Web User Interface (WUI) to connect to CICS, set the **TCPIPSSLCIPHERS** WUI server initialization parameter to the name of an SSL cipher suite specification file that contains SP800-131A conformant cipher suites.

   The sample file `fipsciphers.xml` is suitable.
8. If you use the CICSPlex SM WUI to connect to CICS, set the **TCPIPSSLCERT** WUI server initialization parameter to the name of an SP800-131A conformant certificate label.

   If you use the key ring default certificate, ensure that it is SP800-131A conformant.

What to do next

Any clients that connect to CICS must be SP800-131A-conformant and must support TLS 1.2. To be conformant, they must be capable of using SP800-131A conformant cipher suites and, if they use certificates, SP800-131A conformant certificates.

Any partner CICS system must use `MINTLSLEVEL=TLS12` or `ENCRYPTION=ALL` to talk to an `MINTLSLEVEL=TLS12` system and it must be configured to use cipher suites and certificates that are SP800-131A conformant.

Note: If you set `NISTSP800131A=CHECK`, CICS takes the following actions:

- When a JVM server is started, CICS sets the Java properties to make Java NIST SP800-131A conformant.
- If you use SAML and sign outbound messages, CICS issues message DFHXS1300 to warn you to check that the certificates used are conformant.
- If you use WS-Security, CICS issues message DFHXS1301 because CICS support of WS-Security is not conformant with NIST SP800-131A.
- When a TCPIPSERVICE defined with SSL(ATTLSAWARE) is opened, CICS issues message DFHSO0148 to warn you that the AT-TLS policy used to secure this TCPIPSERVICE must be conformant with **NIST SP800-131A**. See AT-TLS policy configuration in z/OS Communications Server: IP Configuration Guide.

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 must authorize CICS to access them.

About this task

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. You can save these details in a specialized profile in the LDAPBIND RACF class. To set up the profile, follow these steps:

Procedure

1. The password that is used in the profile must be encrypted before it is stored in the RACF database. To encrypt the password, you must 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://EXAMPLE.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)
   ```

4. Specify the profile name in the system initialization parameter **CRLPROFILE** for each applicable CICS region.

Results

When you start a CICS region with the profile name specified in the **CRLPROFILE** system initialization parameter, the bind information for the LDAP server is cached in the SSL environment for the CICS region, which is managed by z/OS System SSL. When you issue the PERFORM SSL REBUILD command for the CICS region, the bind information for the LDAP server is refreshed from the external security manager.

What to do next

If the **CRLPROFILE** parameter is specified for a CICS region but is invalid, or if the specified profile contains invalid data, or if the LDAP server identified by the profile is unavailable when the CICS region starts, the CICS region disables its own access to the LDAP server. Messages DFHSO0128 and DFHSO0129 report this problem.

To restore access, you must fix the error and restart the CICS region. The PERFORM SSL REBUILD command cannot restore access to the LDAP server if the CICS region has disabled it. The refresh only takes place for an LDAP server that was available to the CICS region at the time when the command was issued.

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.

Before you begin

To use certificate revocation lists, you must install and configure an LDAP server. See z/OS Cryptographic Services PKI Services Guide and Reference. You also need to authorize CICS to access the LDAP server, as described in Configuring LDAP for CICS use.

About this task

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.

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.

Before you begin

If you need to install and configure an LDAP server, see z/OS Cryptographic Services PKI Services Guide and Reference.

About this task

Procedure

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.

Example

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"
```

What to do next

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” on page 313.

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.

Before you begin

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 312 for detailed instructions.

About this task

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.

Procedure

1. Specify the name of the LDAP server in the system initialization parameter **CRLPROFILE**.
2. Run the CCRL transaction
   - from a terminal. See “Running CCRL from a terminal” on page 313 .
   - using a command. See “Running CCRL from a START command” on page 314.

Running CCRL from a terminal

You can run the CICS-supplied transaction CCRL using a terminal to download certificate revocation lists (CRLs).

Before you begin

Read “Running the CCRL transaction” on page 313 to find out about the prerequisites before running this transaction from a terminal.

Procedure

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 312

Results

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.

What to do next

To set up regular updates, you can use a START command. See "Running CCRL from a START command" on page 314.

Running CCRL from a START command

You can schedule the CCRL transaction to run at regular intervals using a START command.

Before you begin

Read "Running the CCRL transaction" on page 313 to find out about the prerequisites before running this transaction from a terminal.

Procedure

- 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
        http://crl.verisign.com/ATTClass2Individual.crl')
        LENGTH(124) INTERVAL(960000)
  ```

  This example schedules the CCRL transaction to run in 96 hours.

Chapter 9. Security for data sources

Security for Db2

In the CICS Db2 environment, there are four main stages at which you can implement security checking.

The four stages are:

- When a CICS user signs on to a CICS region. CICS sign-on authenticates users by checking that they supply a valid user ID and password..
- When a CICS user tries to use or modify a CICS resource that is related to Db2. This could be a DB2CONN, DB2ENTRY or DB2TRAN resource definition; or a CICS transaction that accesses Db2 to obtain data; or a CICS transaction that issues commands to the CICS Db2 attachment facility or to Db2 itself. At this stage, you can use CICS security mechanisms, which are managed by RACF or an equivalent external security manager, to control the CICS user's access to the resource.
- When the CICS region connects to Db2, and when a transaction acquires a thread into Db2. Both the CICS region and the transaction must provide authorization IDs to Db2, and these authorization IDs are validated by RACF or an equivalent external security manager.
- When a CICS user tries to use a CICS transaction to execute or modify a Db2 resource. This could be a plan, or a Db2 command, or a resource that is needed to execute dynamic SQL. At this stage, you can use Db2 security checking, which is managed either by Db2 itself, or by RACF or an equivalent external security manager, to control the CICS user's access to the resource.

You can also use RACF, or an equivalent external security manager, to protect the components that make up CICS and Db2 from unauthorized access. You can apply this protection to Db2 databases, logs, bootstrap data sets (BSDSs), and libraries outside the scope of Db2, and to CICS data sets and libraries. You can use VSAM password protection as a partial replacement for the protection provided by RACF. For more information, see CICS system resource security.

Note: RACF is referred to here as the external security manager used by CICS. Except for the explicit RACF examples, the general discussion applies equally to any functionally equivalent non-IBM external security manager.

Figure 27 on page 316 shows the security mechanisms involved in a CICS Db2 environment.

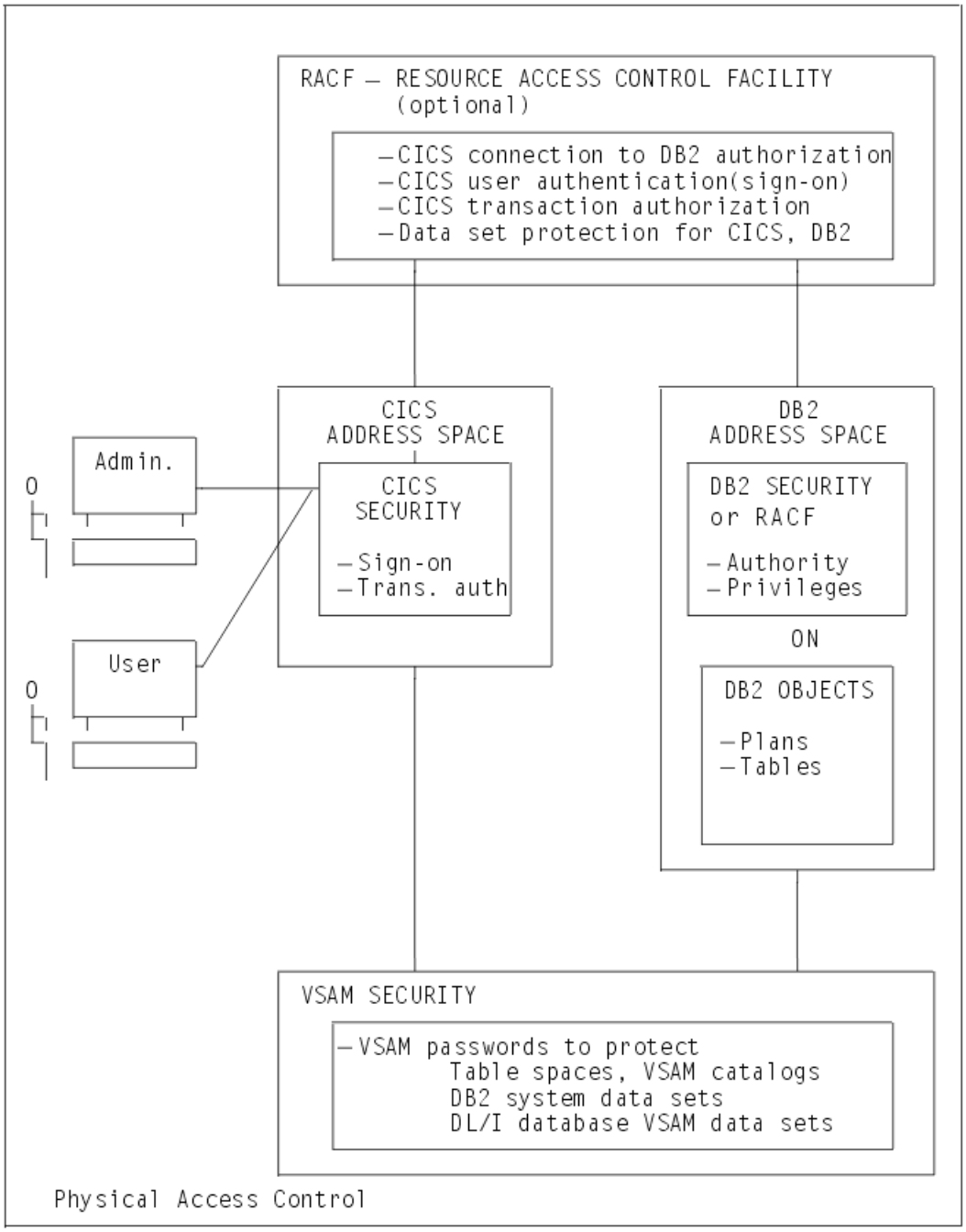


Figure 27. Overview of the CICS Db2 security mechanisms

Controlling access to Db2-related resources in CICS

You can control access to Db2 resources in your CICS region, and initiate security checking for the resources, by enabling an external security manager and the appropriate CICS security mechanism.

About this task

CICS users might want to perform the following activities involving Db2:

- Inquire on, modify, create or discard DB2CONN, DB2ENTRY, and DB2TRAN resource definitions.
- Use a transaction that accesses Db2 to obtain data, or issue CICS Db2 attachment facility commands or Db2 commands using the DSNC transaction.

Use RACF, or an equivalent external security manager to perform security checks in your CICS region. When a user tries to access protected resources, CICS calls the external security manager to perform security checking. RACF makes security checks using the CICS user's ID, which is authenticated when the user signs on to CICS. If a user does not sign on to CICS, they are given the default user ID, unless a user ID has been permanently associated with a terminal using preset security.

Enable the appropriate security mechanism for your CICS region: transaction-attach security, resource security, command security, or surrogate security.

For more information about CICS security, see CICS TS security.

Controlling users' access to DB2CONN, DB2TRAN, and DB2ENTRY resource definitions

You can control users' access to DB2CONN, DB2TRAN, and DB2ENTRY resource definitions by enabling different CICS security mechanisms.

About this task

The following security mechanisms can be used to control user access to Db2 resource definitions.

- Control users' ability to access particular resources by using the CICS **resource security** mechanism. Resource security is implemented at the transaction level. For example, you could prevent some users from modifying a particular DB2ENTRY definition. “Using resource security to control access to DB2ENTRY and DB2TRAN resource definitions” on page 317 tells you how to use this security mechanism.
- Control users' ability to issue particular SPI commands against Db2-related resources by using the CICS **command security** mechanism. Command security is also implemented at the transaction level. For example, you could permit only certain users to issue CREATE and DISCARD commands against DB2ENTRY resource definitions. “Using command security to control the issuing of SPI commands against DB2CONN, DB2ENTRY, and DB2TRAN resource definitions” on page 319 tells you how to use this security mechanism.
- Control users' ability to modify the authorization IDs that CICS provides to Db2, by using the CICS **surrogate security and AUTHTYPE security** mechanisms. The authorization IDs are used for Db2 security checking, and they are set by the AUTHID, COMAUTHID, AUTHTYPE and COMAUTHTYPE attributes on Db2-related resource definitions, and by the SIGNID attribute on the DB2CONN definition for the CICS region. CICS checks that the user who wants to modify the authorization ID, is permitted to act on behalf of the existing authorization ID that is specified in the resource definition. “Using surrogate security and AUTHTYPE security to control access to the authorization IDs that CICS provides to Db2” on page 320 tells you how to use these security mechanisms.

Using resource security to control access to DB2ENTRY and DB2TRAN resource definitions

The CICS resource security mechanism controls users' access to named CICS resources. For example, you can use it to protect certain resources, such as a particular DB2ENTRY definition, from being modified by particular users.

About this task

CICS command security can prevent users from performing certain actions on types of resources, such as "all DB2ENTRY definitions", but it cannot protect individual items within the resource type.

Because your CICS region only has one DB2CONN definition, you do not need to use resource security to protect it; you can control access to the DB2CONN definition using command security. Also, DB2TRAN definitions, for the purpose of resource security, are treated as extensions of the DB2ENTRY definition to which they refer, and are not defined for resource security in their own right. If you give a user permission to access a DB2ENTRY definition, you also give them permission to access the DB2TRAN definitions that refer to it. (In the case where a transaction changes the name of the DB2ENTRY with which a DB2TRAN

definition is associated, a double security check is performed, to verify the user's authority to modify both the old DB2ENTRY to which the definition referred, and the new DB2ENTRY to which it will refer.) For resource security, you therefore only need to define your DB2ENTRY definitions to RACF.

When resource security is enabled for a transaction, the external security manager checks that the user ID associated with the transaction is authorized to modify the resource that is involved. Security of resource definitions has more information about this process.

To protect your Db2-related resources using resource security, complete these steps.

Procedure

1. To enable RACF, or an equivalent external security manager, and make resource security available for a CICS region, specify SEC=YES as a system initialization parameter for the CICS region.
2. In RACF, create general resource classes to contain your Db2-related resources. You need a member class and a grouping class.

   Unlike the RACF default resource class names for CICS, there are no IBM-supplied default class names for DB2ENTRY resources. Create your own installation-defined class names by adding new class descriptors to the installation-defined part (module ICHRRCDE) of the RACF class descriptor table (CDT). For an example of how to do this, see the IBM-supplied sample job, RRCDTE, supplied in member DFH$RACF of CICSTS56.CICS.SDFHSAMP. This gives an example of a member class called XCICSDB2, and a grouping class called ZCICSDB2. This example uses the same naming convention as the default resource class names for CICS. Do not use existing CICS class names for Db2-related resource definitions; instead, create new class names using a similar naming convention.
3. Define profiles for your DB2ENTRY definitions in the resource classes that you have created.

   For example, to add a number of DB2ENTRY names to the XCICSDB2 resource class, use the RDEFINE command as follows:

   ```
   RDEFINE  XCICSDB2  (db2ent1, db2ent2, db2ent3.., db2entn) UACC(NONE)
                    NOTIFY(sys_admin_userid)
   ```

   Protecting DB2ENTRY resource definitions also protects access to associated DB2TRAN definitions, because a DB2TRAN is considered to be an extension to the DB2ENTRY to which it refers. You do not need to protect your DB2CONN definition using resource security.
4. To activate resource security for your Db2-related resources, specify XDB2=*name* as a system initialization parameter for the CICS region, where *name* is the general resource class name that you defined for your Db2-related resources.
5. Specify RESSEC=YES in the resource definition for any transactions involving Db2-related resources for which you want to enable resource security. Now, when a user tries to use one of these transactions to access one of the Db2-related resources that you have protected, RACF checks that the user ID is authorized to access that resource.
6. Give permission to your CICS users, or groups of users, to perform appropriate actions on each Db2-related resource that you have protected.

   Remember that if a user has permission to perform actions on a DB2ENTRY definition, they are automatically authorized to perform the same actions on DB2TRAN definitions associated with it.The access that users need to perform certain actions is as follows:

   INQUIRE command
   Requires READ authority

   SET command
   Requires UPDATE authority

   CREATE command
   Requires ALTER authority

   DISCARD command
   Requires ALTER authority

For example, you can use the PERMIT command to authorize a group of users to modify a protected DB2ENTRY, db2ent1 in class XCICSDB2, with UPDATE authority, as follows:

```
PERMIT db2ent1 CLASS(XCICSDB2) ID(group1) ACCESS(UPDATE)
```

Using command security to control the issuing of SPI commands against DB2CONN, DB2ENTRY, and DB2TRAN resource definitions

Use CICS command security mechanisms to protect DB2CONN, DB2ENTRY, and DB2TRAN resource definitions.

About this task

The CICS command security mechanism controls users' ability to issue particular SPI commands against types of Db2-related resource. For example, you can use it to control which users are allowed to issue CREATE and DISCARD commands against DB2ENTRY resource definitions. Unlike resource security, CICS command security cannot protect individual named resources; it is designed to protect types of resource. You can use command security to protect DB2CONN, DB2ENTRY, and DB2TRAN resource definitions.

When command security is enabled for a transaction, the external security manager checks that the user ID associated with the transaction is authorized to use that command to modify the type of resource that is involved. CICS command security has more information about this process.

If you have both resource security and command security enabled for a particular transaction, RACF performs two security checks against the user ID. For example, if a transaction involves the user issuing a DISCARD command against DB2ENTRY definition db2ent1, RACF checks:

1. That the user ID is authorized to issue the DISCARD command (ALTER authority) against the DB2ENTRY resource type.
2. That the user ID is authorized to access the DB2ENTRY definition db2ent1 with ALTER authority.

To protect your Db2-related resources using command security, complete these steps.

Procedure

1. To enable RACF, or an equivalent external security manager, for a CICS region, specify SEC=YES as a system initialization parameter for the region.
2. Add the Db2 resource names DB2CONN, DB2ENTRY, and DB2TRAN as resource identifiers in one of the IBM-supplied RACF resource classes for CICS commands, CCICSCMD or VCICSCMD.

   Alternatively, you can use a user-defined general resource class for your CICS commands. CICS resources subject to command security checking tells you more about this.

   For example, you can use the REDEFINE command to define a profile named CMDSAMP in the default class VCICSCMD, and use the ADDMEM operand to specify that the Db2 resource types are to be protected by this profile, as follows:

   ```
   RDEFINE  VCICSCMD CMDSAMP UACC(NONE)
                    NOTIFY(sys_admin_userid)
                    ADDMEM(DB2CONN, DB2ENTRY, DB2TRAN)
   ```

3. To make command security available for a CICS region:
   a) If you have used the IBM-supplied RACF resource classes CCICSCMD or VCICSCMD for CICS command profiles, specify XCMD=YES as a system initialization parameter for the region.

      Specifying YES means that CCICSCMD and VCICSCMD are used to build RACF's in-storage profiles.

   b) If you have used a user-defined general resource class for CICS commands, specify XCMD=*user_class* as a system initialization parameter for the region, where *user_class* is the name of the user-defined general resource class.
4. Specify CMDSEC=YES in the resource definition for any transactions involving Db2-related resources for which you want to enable command security. Now, when a user tries to use one of these

transactions to issue a command to modify one of the Db2-related resources that you have protected, RACF checks that the user ID is authorized to issue that command against that type of resource.

5. Give permission to your CICS users, or groups of users, to issue appropriate commands against each type of Db2-related resource. For command security, you need to give separate permissions relating to the DB2TRAN resource type, as well as to the DB2ENTRY resource type. You can also protect the DB2CONN resource type (that is, the CICS region's DB2CONN definition).

   The access that users need to issue certain commands is as follows:

   INQUIRE command
   Requires READ authority

   SET command
   Requires UPDATE authority

   CREATE command
   Requires ALTER authority

   DISCARD command
   Requires ALTER authority

   For example, if you have defined the Db2 resource types in the CMDSAMP profile as in the example in Step 2, you can use the PERMIT command to authorize a group of users to issue EXEC CICS INQUIRE commands against the Db2 resource types as follows:

```
PERMIT CMDSAMP CLASS(VCICSCMD) ID(operator_group) ACCESS(READ)
```

   Within a transaction, you can query whether a user ID has access to Db2 resource types by using the **EXEC CICS QUERY SECURITY RESTYPE(SPCOMMAND)** command, with the RESID parameter specifying DB2CONN, DB2ENTRY, or DB2TRAN.

Using surrogate security and AUTHTYPE security to control access to the authorization IDs that CICS provides to Db2

The CICS surrogate security and AUTHTYPE security mechanisms control users' ability to modify the authorization IDs that CICS provides to Db2.

About this task

Use surrogate security and AUTHTYPE security to ensure that only certain users are permitted to change the authorization IDs, which are used for security checking by Db2 own security checking. Surrogate security and AUTHTYPE security are set for the whole CICS region, and any transactions that involve changes to the authorization IDs are subject to them.

“Providing authorization IDs to Db2 for the CICS region and for CICS transactions” on page 323 explains how to select and change these authorization IDs. To summarize, the authorization IDs that CICS provides to Db2 are set by the AUTHID, COMAUTHID, AUTHTYPE, and COMAUTHTYPE attributes on Db2-related resource definitions, and by the SIGNID attribute on the DB2CONN definition for the CICS region. To change the authorization IDs, you first need authority to modify the DB2CONN and DB2ENTRY definitions, which might be protected by command security or resource security. Surrogate security provides an extra layer of protection, because it involves CICS acting on behalf of Db2 to check that the user that is modifying the authorization ID is permitted to act as a surrogate for the existing authorization ID that is specified in the resource definition.

True surrogate security provides security checking when a user attempts to change the SIGNID, AUTHID or COMAUTHID attributes on a DB2CONN or DB2ENTRY definition, all of which specify an authorization ID that is used when a process signs on to Db2. CICS uses the surrogate user facility of RACF to perform this checking. A surrogate user is one who has the authority to do work on behalf of another user, without knowing that other user's password. When a user attempts to change one of the SIGNID, AUTHID or COMAUTHID attributes, CICS calls RACF to check that the user is authorized as a surrogate of the authorization ID that is presently specified on the SIGNID, AUTHID or COMAUTHID attribute.

For the AUTHTYPE and COMAUTHTYPE attributes, which give a type of authorization ID to be used rather than specifying an exact authorization ID, CICS cannot use true surrogate security. Instead, it uses a mechanism called AUTHTYPE security. When a user attempts to change one of the AUTHTYPE or COMAUTHTYPE attributes, CICS calls RACF to check that the user is authorized through a profile that you have defined for the resource definition in the RACF FACILITY general resource class. Although AUTHTYPE security is not true surrogate security, it is enabled by the same system initialization parameter, and you will probably want to use it in addition to surrogate security, so the instructions in this topic tell you how to set up both types of security.

Note that when DB2CONN and DB2ENTRY resource definitions are installed as part of a cold or initial start of CICS, if surrogate security and AUTHTYPE security are enabled, RACF makes surrogate security and AUTHTYPE security checks for the CICS region user ID. If you install DB2CONN and DB2ENTRY resource definitions in this way, ensure that the CICS region user ID is defined as a surrogate user for any authorization IDs specified in the resource definitions, and also that it is authorized through the correct profiles in the RACF FACILITY general resource class.

To implement surrogate security and AUTHTYPE security to protect the authorization IDs that CICS provides to Db2, complete the following steps:

Procedure

1. To enable RACF, or an equivalent external security manager, for a CICS region, specify SEC=YES as a system initialization parameter for the region.
2. To activate surrogate security and AUTHTYPE security for a CICS region, specify XUSER=YES as a system initialization parameter for the region. This system initialization parameter enables both the security mechanisms. When the security mechanisms are enabled, CICS calls RACF to perform security checks whenever a transaction involves EXEC CICS SET, CREATE, and INSTALL commands that operate on the SIGNID, AUTHID, COMAUTHID, AUTHTYPE and COMAUTHTYPE attributes of DB2CONN and DB2ENTRY resource definitions. For the SIGNID, AUTHID and COMAUTHID attributes, RACF performs the surrogate security check, and for the AUTHTYPE or COMAUTHTYPE attributes, RACF performs the AUTHTYPE security check.
3. For the purpose of surrogate security, you need to define appropriate CICS users, or groups of users, as surrogates of any authorization IDs that are specified on the SIGNID, AUTHID or COMAUTHID attributes of your DB2CONN and DB2ENTRY definitions. To define a user ID as a surrogate of an authorization ID:
   a) Create a profile for the authorization ID in the RACF SURROGAT class, with a name of the form *authid*.DFHINSTL, with the authorization ID defined as the owner.
   For example, if you have specified DB2AUTH1 as an authorization ID on a SIGNID, AUTHID or COMAUTHID attribute, use the following command to create a profile:

   ```
   RDEFINE  SURROGAT  DB2AUTH1.DFHINSTL  UACC(NONE) OWNER(DB2AUTH1)
   ```

   b) Permit appropriate CICS users to act as a surrogate for the authorization ID, by giving them READ authority to the profile you have created.
   For example, to permit a user with the ID CICSUSR1 to act as a surrogate for the authorization ID DB2AUTH1, and therefore to install or modify any SIGNID, AUTHID or COMAUTHID attributes that specify DB2AUTH1 as the existing authorization ID, use the following command:

   ```
   PERMIT   DB2AUTH1.DFHINSTL  CLASS(SURROGAT) ID(CICSUSR1)  ACCESS(READ)
   ```

   Repeat this process for all the authorization IDs that you have specified on SIGNID, AUTHID or COMAUTHID attributes.
   c) If you might need to install DB2CONN and DB2ENTRY resource definitions containing SIGNID, AUTHID or COMAUTHID attributes as part of a cold or initial start of CICS, permit the CICS region user ID to act as a surrogate for any authorization IDs specified by those attributes.

   The defaults for DB2CONN and DB2ENTRY resource definitions do not involve the AUTHID and COMAUTHID attributes. The default SIGNID for an installed DB2CONN definition is the applid of the CICS region.

4. For the purpose of AUTHTYPE security, you need to create a profile for each of your DB2CONN or DB2ENTRY resource definitions in the RACF FACILITY general resource class, and give appropriate CICS users, or groups of users, READ authority to the profiles. (This process imitates the true surrogate security mechanism, but does not involve the use of a specific authorization ID; instead, it protects each resource definition.) To do this:

   a) Create a profile for the DB2CONN or DB2ENTRY resource definition in the RACF FACILITY general resource class, with a name of the form DFHDB2.AUTHTYPE.*authname*, where *authname* is the name of the DB2CONN or DB2ENTRY resource definition.
   For example, to define a profile for a DB2CONN resource definition named DB2CONN1, use the following command:

   ```
   RDEFINE  FACILITY  DFHDB2.AUTHTYPE.DB2CONN1   UACC(NONE)
   ```

   b) Give appropriate CICS users READ authority to the profile you have created.
   For example, to permit a user with the ID CICSUSR2 to install or modify the AUTHTYPE or COMAUTHTYPE attributes on a DB2CONN resource definition named DB2CONN1, use the following command:

   ```
   PERMIT  DFHDB2.AUTHTYPE.DB2CONN1 CLASS(FACILITY) ID(CICSUSR2) ACCESS(READ)
   ```

   Repeat this process for each of your DB2CONN and DB2ENTRY resource definitions. Also, if you might need to install DB2CONN and DB2ENTRY resource definitions containing AUTHTYPE or COMAUTHTYPE attributes as part of a cold or initial start of CICS, give READ authority to the CICS region user ID on the profiles for those resource definitions.

Controlling users' access to Db2-related CICS transactions

Use the CICS transaction-attach security mechanism to control users' access to: CICS transactions that access Db2 to obtain data, the DSNC transaction, and to any other transactions that issue CICS Db2 attachment facility commands and Db2 commands.

About this task

When transaction-attach security is enabled, RACF, or an equivalent external security manager, checks that the CICS user is authorized to run the transaction that they have requested.

To protect Db2-related transactions using transaction-attach security, follow the instructions in Transaction security. The process is the same for all CICS transactions; there are no special considerations for Db2-related transactions as far as the transaction-attach security mechanism is concerned. The instructions tell you how to:

- Set up appropriate system initialization parameters for the CICS region to activate transaction-attach security (see CICS parameters controlling transaction-attach security).
- Define transaction profiles to RACF for the transactions that you want to protect (see RACF profiles).

If you have defined transactions other than DSNC to issue CICS Db2 attachment facility commands and Db2 commands (for example, if you have created a separate transaction to run each command), remember to define these transactions to RACF as well.

You can now control which CICS users can use transactions that access Db2. Add the appropriate users or groups of users to the access list for the transaction profiles, with READ authority. RACF profiles has some recommendations about this.

For transactions that issue CICS Db2 attachment facility commands and Db2 commands, bear in mind that:

- CICS Db2 attachment facility commands operate on the connection between CICS and Db2, and they run entirely within CICS. Db2 commands operate in Db2 itself, and they are routed to Db2. You can distinguish Db2 commands from CICS Db2 attachment facility commands by the hyphen (**-**) character, which is entered with Db2 commands.

- If you have access to the DSNC transaction, CICS allows you to issue all of the CICS Db2 attachment facility commands and Db2 commands.
- If you have defined separate transactions to run individual CICS Db2 attachment facility commands and Db2 commands, you can give different CICS users authorization to subsets of these transaction codes, and therefore to subsets of the commands. For example, you could give some users authority to issue CICS Db2 attachment facility commands, but not Db2 commands. CICS-supplied transactions for CICS Db2 has the names of the separate transaction definitions that CICS supplies for the CICS Db2 attachment facility commands and the Db2 commands.

CICS Db2 attachment facility commands do not flow to Db2, so they are not subject to any further security checking. They are only protected by CICS transaction-attach security. However, Db2 commands, and CICS transactions that access Db2 to obtain data, are subject to further stages of security checking by the Db2 security mechanisms, as follows:

- When a transaction signs on to Db2, it must provide valid authorization IDs to Db2. The authorization IDs are checked by RACF or an equivalent external security manager.
- Because the transaction is issuing a Db2 command or accessing Db2 data, the authorization IDs that it has provided must have permission to perform these actions within Db2. In Db2, you can use GRANT statements to give the authorization IDs permission to perform actions.

In addition, the CICS region itself must be authorized to connect to the Db2 subsystem.

"Providing authorization IDs to Db2 for the CICS region and for CICS transactions" on page 323 tells you how to authorize the CICS region to connect to the Db2 subsystem, and how to provide valid authorization IDs for transactions.

"Authorizing users to access resources in Db2" on page 330 tells you how to grant permissions to the authorization IDs that the transactions have provided to Db2.

Providing authorization IDs to Db2 for the CICS region and for CICS transactions

CICS has two types of process that must provide Db2 with authorization IDs: the overall connection between a CICS region and Db2, and CICS transactions that acquire a thread into Db2.

About this task

For the purposes of security, Db2 uses the term "process" to represent all forms of access to data, either by users interacting directly with Db2, or by users interacting with Db2 by way of other programs, including CICS. A process that connects to or signs on to Db2 must provide one or more Db2 short identifiers, called authorization IDs, that can be used for security checking in the Db2 address space. Every process must provide a primary authorization ID, and it can optionally provide one or more secondary authorization IDs. Db2 privileges and authority can be granted to either primary or secondary authorization IDs. For example, users can create a table using their secondary authorization ID. The table is then owned by that secondary authorization ID. Any other user that provides Db2 with the same secondary authorization ID has associated privileges over the table. To take privileges away from a user, the administrator can disconnect the user from that authorization ID.

CICS has two types of process that need to provide Db2 with authorization IDs:

- The overall connection between a CICS region and Db2, which is created by the CICS Db2 attachment facility. This process has to go through Db2 connection processing to provide Db2 with authorization IDs.
- CICS transactions that acquire a thread into Db2. These could be, for example, a transaction that is retrieving data from a Db2 database, or the DSNC transaction that is issuing a Db2 command. For each CICS transaction, the actual process that Db2 sees is the thread TCB, which CICS uses to control a transaction's thread into Db2. These processes have to go through the Db2 sign-on processing to provide Db2 with authorization IDs.

During connection processing and sign-on processing, Db2 sets the primary and secondary authorization IDs for the process to use in the Db2 address space. By default, Db2 uses the authorization IDs that the process has provided. However, both connection processing and sign-on processing involve exit routines, and these exit routines allow you to influence the setting of the primary and secondary authorization IDs. Db2 has a default connection exit routine and a default sign-on exit routine. You can replace these with your own exit routines, and a sample connection exit routine and sign-on exit routine are supplied with Db2 to assist you with this.

Providing authorization IDs to Db2 for a CICS region

CICS provides a primary authorization ID and one or more secondary authorization IDs to Db2.

About this task

When the CICS Db2 attachment facility creates the overall connection between a CICS region and Db2, the process goes through the Db2 connection processing. The CICS region can provide:

- A primary authorization ID. The primary authorization ID becomes the CICS region's primary ID in Db2. For the connection between a CICS region and Db2, you cannot choose the primary authorization ID that is initially passed to the Db2 connection processing; it is the user ID for the CICS region. However, it is possible to change the primary ID that Db2 sets during connection processing, by writing your own connection exit routine. If RACF, or an equivalent external security manager, is active, the user ID for the CICS region must be defined to it. "Providing a primary authorization ID for a CICS region" on page 324 tells you about the possible primary authorization IDs for a CICS region.
- One or more secondary authorization IDs. You can use the name of a RACF group, or list of groups, as secondary authorization IDs for the CICS region. If you do this, you need to replace the default Db2 connection exit routine DSN3@ATH, which only passes primary authorization IDs to Db2. The sample Db2 connection exit routine DSN3SATH passes the names of RACF groups to Db2 as secondary authorization IDs. Alternatively, you can write your own connection exit routine that sets secondary IDs for the CICS region. "Providing secondary authorization IDs for a CICS region" on page 325 tells you how to set up secondary authorization IDs for a CICS region.

Providing a primary authorization ID for a CICS region

The primary authorization ID that is passed to Db2 depends whether CICS is running as a started task, a started job, or a job.

About this task

The connection type that CICS requests from Db2 is single address space subsystem (SASS). For the connection between a CICS region and Db2, you cannot choose the primary authorization ID that is initially passed to the Db2 connection processing. The ID that is passed to Db2 as the primary authorization ID for the CICS region is one of the following:

- The user ID taken from the RACF started procedures table, ICHRIN03, if CICS is running as a started task.
- The user parameter of the STDATA segment in a STARTED general resource class profile, if CICS is running as a started job.
- The user ID specified on the USER parameter of the JOB card, if CICS is running as a job.

The user ID that a CICS region might use must be defined to RACF, or your equivalent external security manager, if the external security manager is active. Define the user ID to RACF as a USER profile. It is not sufficient to define it as a RESOURCE profile.

Once you have defined the CICS region's user ID to RACF, permit it to access Db2, as follows:

1. Define a profile for the Db2 subsystem with the single address space subsystem (SASS) type of connection, in the RACF class DSNR. For example, the following RACF command creates a profile for SASS connections to Db2 subsystem DB2A in class DSNR:

```
RDEFINE DSNR (DB2A.SASS) OWNER(DB2OWNER)
```

2. Permit the user ID for the CICS region to access the Db2 subsystem. For example, the following RACF command permits a CICS region with a user ID of CICSHA11 to connect to Db2 subsystem DB2A:

```
PERMIT DB2A.SASS CLASS(DSNR) ID(CICSHA11) ACCESS(READ)
```

The Db2 connection exit routine takes the primary authorization ID (the user ID) provided by the CICS region, and sets it as the primary ID for the CICS region in Db2. The default Db2 connection exit routine DSN3@ATH, and the sample Db2 connection exit routine DSN3SATH, both behave in this way. It is possible to change the primary ID that Db2 sets, by writing your own connection exit routine. For more information about the sample connection exit routine and about writing exit routines, see Securing Db2 in Db2 for z/OS product documentation. However, you might find it more straightforward to provide secondary authorization IDs for the CICS region, and grant permissions to the CICS region based on these, rather than on the primary authorization ID.

Providing secondary authorization IDs for a CICS region

When a CICS region connects to Db2, it can provide one or more secondary authorization IDs to Db2, in addition to the primary authorization ID. You can use the name of the RACF group, or list of groups, to which the CICS region is connected, as secondary authorization IDs. This enables you to grant Db2 privileges and authority to RACF groups, and then connect multiple CICS regions to the same groups, instead of granting Db2 privileges to all the possible primary authorization IDs for each CICS region.

About this task

To provide the name of the RACF group, or list of groups, to which a CICS region is connected, to Db2 as secondary authorization IDs, complete the following steps.

Procedure

1. Specify SEC=YES as a system initialization parameter for the CICS region, to ensure that CICS uses RACF.
2. Connect the CICS region to the appropriate RACF group or list of groups.

   See RACF group profiles.
3. Replace the default Db2 connection exit routine DSN3@ATH.

   This exit routine is driven during connection processing. The default connection exit routine does not support secondary authorization IDs, so you need to replace it with the sample connection exit routine DSN3SATH, which is provided with Db2, or with your own routine. DSN3SATH is shipped in source form in the Db2 SDSNSAMP library, and you can use it as a basis for your own routine. DSN3SATH passes the names of the RACF groups to which the CICS region is connected, to Db2 as secondary authorization IDs. If the RACF list of groups option is active, DSN3SATH obtains all the group names to which the CICS region is connected, and uses these as secondary authorization IDs. If the RACF list of groups option is not active, DSN3SATH uses the name of the CICS region's current connected group as the only secondary authorization ID.

Results

When the CICS region connects to Db2, the sample connection exit routine sets the CICS region's primary authorization ID (the region's user ID) as the primary ID, and sets the names of the RACF groups to which the CICS region is connected, as secondary IDs.

What to do next

As an alternative to providing the names of RACF groups as secondary authorization IDs to Db2, you can write your own connection exit routine that sets secondary IDs for the CICS region. For information about writing connection exit routines, see Securing Db2 in Db2 for z/OS product documentation.

Providing authorization IDs to Db2 for CICS transactions

So that CICS can pass authorization IDs to Db2, CICS must be using RACF; SEC=YES must be specified in the SIT. This is because CICS needs to pass a RACF access control environment element (ACEE) to Db2.

About this task

When a CICS transaction's thread TCB signs on to Db2 and goes through the Db2 sign-on processing, it can provide:

- A primary authorization ID. For CICS transactions, you can choose the primary authorization ID. It can be the user ID or operator ID of the CICS user, or a terminal ID, or a transaction ID; or it can be an ID that you have specified. The ID that is used as the primary authorization ID is determined by attributes in the DB2ENTRY definition (for entry threads), or in the DB2CONN definition (for pool threads and command threads). "Providing a primary authorization ID for CICS transactions" on page 327 tells you how to choose the primary authorization ID for a CICS transaction.
- One or more secondary authorization IDs. You can use the name of a RACF group, or list of groups, as secondary authorization IDs. This has the advantage that you can grant Db2 privileges and authority to RACF groups, rather than to each individual CICS user. To use secondary authorization IDs, use the AUTHTYPE attribute in the DB2ENTRY definition (for entry threads), or the AUTHTYPE or COMAUTHTYPE attributes in the DB2CONN definition (for pool threads or command threads), to specify the GROUP option. You also need to replace the default Db2 sign-on exit routine DSN3@SGN, because the default routine does not pass secondary authorization IDs to Db2. When you specify the GROUP option, the primary authorization ID is automatically defined as the user ID of the CICS user associated with the transaction. "Providing secondary authorization IDs for CICS transactions" on page 329 tells you how to set up and use secondary authorization IDs.

A key consideration for choosing the authorization IDs that CICS transactions provide to Db2 is the security mechanism that you have chosen for security checking in the Db2 address space. This security checking covers access to Db2 commands, plans, and dynamic SQL. You can choose to have this security checking carried out by:

- Db2 internal security.
- RACF, or an equivalent external security manager.
- Partly Db2, and partly RACF.

If you are using RACF for some or all of the security checking in your Db2 address space, CICS transactions that sign on to Db2 **must** provide an authorization ID by one of the following methods:

- Specify AUTHTYPE(USERID) or COMAUTHTYPE(USERID) in the appropriate definition for the thread (DB2ENTRY or DB2CONN), to provide the user ID of the CICS user associated with the transaction to Db2 as the primary authorization ID.
- Specify AUTHTYPE(GROUP) or COMAUTHTYPE(GROUP) in the appropriate definition for the thread (DB2ENTRY or DB2CONN), to provide the user ID of the CICS user associated with the transaction to Db2 as the primary authorization ID, and the name of a RACF group or list of groups as the secondary authorization IDs.
- Specify AUTHTYPE(SIGN) in the appropriate definition for the thread (DB2ENTRY or DB2CONN), and specify the CICS region user ID in the SIGNID attribute of the DB2CONN, to provide the CICS region ID to Db2 as the primary authorization ID.

Note that if the RACF access control environment element (ACEE) in the CICS region is changed in a way that affects the CICS Db2 attachment facility, Db2 is not aware of the change until a sign-on occurs. You can use the CEMT or EXEC CICS SET DB2CONN SECURITY(REBUILD) command to cause the CICS Db2

attachment facility to issue a Db2 sign-on the next time a thread is reused, or when a thread is built on an already signed-on TCB. This ensures that Db2 is made aware of the security change.

Providing a primary authorization ID for CICS transactions

When a CICS transaction's thread TCB signs on to Db2, it must provide a primary authorization ID to Db2. The ID that a transaction uses as its primary authorization ID is determined by an attribute in the resource definition for the thread that the transaction uses to access Db2.

About this task

This means that all transactions that use the same type of thread (either the same type of entry thread, or pool threads, or command threads) must use the same type of primary authorization ID. In each CICS region, you need to set a primary authorization ID for:

- Each type of entry thread, using your DB2ENTRY definitions.
- The pool threads, using your DB2CONN definition.
- The command threads (used for the DSNC transaction), using your DB2CONN definition.

Before you start to set primary authorization IDs, ensure that you have authority to do so. As well as having authority to change your DB2CONN or DB2ENTRY definitions, if surrogate user checking is in force for the CICS region (that is, the system initialization parameter XUSER is set to YES), you need to obtain special authority to perform operations involving Db2 authorization IDs. These operations are modifying the AUTHID, COMAUTHID, AUTHTYPE, or COMAUTHTYPE attributes on a DB2ENTRY or DB2CONN definition, and modifying the SIGNID attribute on a DB2CONN definition. “Using surrogate security and AUTHTYPE security to control access to the authorization IDs that CICS provides to Db2” on page 320 tells you how to grant users authority to perform these operations.

There are two methods of setting the primary authorization ID for a particular type of thread:

1. Use the AUTHID attribute in the DB2ENTRY definition (for entry threads), or the AUTHID or COMAUTHID attribute in the DB2CONN definition (for pool threads or command threads), to specify a primary authorization ID. For example, you could define `AUTHID=test2`. In this case, the CICS Db2 attachment facility passes the characters TEST2 to Db2 as the primary authorization ID.

   Using AUTHID or COMAUTHID does not permit the use of secondary authorization IDs, and also is not compatible with the use of RACF, or an equivalent external security manager, for security checking in the Db2 address space.
2. Use the AUTHTYPE attribute in the DB2ENTRY definition (for entry threads), or the AUTHTYPE or COMAUTHTYPE attribute in the DB2CONN definition (for pool threads or command threads), to instruct CICS to use an existing ID that is relevant to the transaction as the primary authorization ID. This ID can be a CICS user ID, operator ID, terminal ID, or transaction ID; or it can be an ID that you have specified in the DB2CONN definition for the CICS region.

   Using AUTHTYPE or COMAUTHTYPE is compatible with the use of RACF (or an equivalent external security manager) for security checking in the Db2 address space, if you use the USERID or GROUP options, and with the use of secondary authorization IDs, if you use the GROUP option.

The two methods of determining the primary authorization ID are mutually exclusive; you cannot specify both AUTHID and AUTHTYPE, or COMAUTHID and COMAUTHTYPE, in the same resource definition.

Remember that all IDs that you select as primary authorization IDs must be defined to RACF, or your equivalent external security manager, if the security manager is active for the Db2 subsystem. For RACF, the primary authorization IDs must be defined as RACF USER profiles, not just as RESOURCE profiles (for example, as a terminal or transaction).

Follow the instructions in DB2CONN resources and DB2ENTRY resources to set up or modify DB2CONN and DB2ENTRY definitions. If you are using the AUTHTYPE or COMAUTHTYPE attributes to determine the primary authorization ID for a type of thread, use Table 47 on page 328 to identify the options that provide the required authorization ID and support the facilities you want. The key points to consider are:

- If you want to provide secondary authorization IDs to Db2 as well as a primary authorization ID, you need to select the GROUP option. When you specify the GROUP option, your primary authorization ID is automatically defined as your CICS user ID, but you can base your security checking on the secondary authorization IDs instead.
- If you are using RACF for security checking in the Db2 address space, you need to select either the GROUP option, or the USERID option. Only these options can pass the RACF access control environment element (ACEE) to Db2, which is required when RACF is used for security checking.
- Think about the performance and maintenance implications of your choice of authorization ID. Selecting authorization IDs for performance and maintenance outlines these. With the USERID, OPID, TERM, TX or GROUP options, sign-on processing occurs more frequently, and maintenance also takes more time, because you need to grant permissions to a greater number of authorization IDs. With the SIGN option, or using the AUTHID attribute instead of the AUTHTYPE attribute, sign-on processing is decreased, and maintenance is less complicated. However, using standard authorization IDs makes the Db2 security checking less granular.
- Think about the accounting implications of your choice of authorization ID. The authorization ID is used in each Db2 accounting record. From an accounting viewpoint, the most detailed information is obtained if using USERID, OPID, GROUP or TERM. However, depending on the ACCOUNTREC specification, it may not be possible to account at the individual user level in any case. For more information about accounting in a CICS Db2 environment, see Accounting and monitoring in a CICS Db2 environment in Monitoring.

Table 47 on page 328 shows the primary authorization IDs that the CICS Db2 attachment facility passes to Db2 when you select each option for the AUTHTYPE or COMAUTHTYPE attributes.

Table 47. Options available on the AUTHTYPE and COMAUTHTYPE attributes

Option	Primary authorization ID passed to Db2	Supports RACF checking for Db2?	Supports secondary auth IDs?
USERID	User ID associated with the CICS transaction, as defined to RACF and used in CICS sign-on	Yes	No
OPID	User's CICS operator ID, defined in the CICS segment of the RACF user profile	No	No
SIGN	An ID you specified in the SIGNID attribute of the DB2CONN definition for the CICS region. Defaults to the applid of the CICS region	Yes, when the SIGNID attribute of the DB2CONN resource matches the CICS region userid.	No
TERM	Terminal ID of the terminal associated with the transaction	No	No
TX	Transaction ID	No	No
GROUP	User's CICS RACF user ID used in CICS sign-on	Yes	Yes

If you are not planning to provide secondary authorization IDs for your CICS transactions, you do not need to replace the default Db2 sign-on exit routine DSN3@SGN. The default sign-on exit routine handles primary authorization IDs. However, the Db2 subsystem to which you are connecting might use a different sign-on exit routine for some other reason. If the Db2 subsystem uses the sample sign-on exit routine DSN3SSGN, you might need to make a change to DSN3SSGN, if **all** of the following conditions are true:

- You have chosen an AUTHID or AUTHTYPE option other than GROUP.
- RACF list of groups processing is active.
- You have transactions whose primary authorization ID is not defined to RACF.

If this is the case, see Securing Db2 in Db2 for z/OS product documentationfor the change you need to make to the sample sign-on exit routine.

Providing secondary authorization IDs for CICS transactions

When a thread TCB that belongs to a CICS transaction signs on to Db2, it can provide one or more secondary authorization IDs to Db2, in addition to the primary authorization ID.

About this task

You can provide Db2 with the name of a CICS user's RACF group, or list of groups, as secondary authorization IDs. This has the advantage that you can grant Db2 privileges and authority to RACF groups, rather than to each individual CICS user. CICS users can then be connected to, or removed from, RACF groups as required. RACF group profiles explains how users can be placed into RACF groups.

You can only provide secondary authorization IDs to Db2 for CICS transactions if you specify the GROUP option for the AUTHTYPE attribute in the DB2ENTRY definition (for entry threads), or the AUTHTYPE or COMAUTHTYPE attributes in the DB2CONN definition (for pool threads or command threads). If you specify any other option for AUTHTYPE or COMAUTHTYPE, the secondary authorization ID is set to blanks. When you specify the GROUP option, you cannot choose the primary authorization ID for the thread type; it is automatically defined as the user ID of the CICS user associated with the transaction. You should base your security checking on the secondary authorization IDs instead.

To provide the names of a user's RACF groups to Db2 as secondary authorization IDs, complete the following steps:

1. Specify SEC=YES as a system initialization parameter for the CICS region, to ensure that CICS uses RACF. If your CICS transaction profile names are defined with a prefix, also specify the system initialization parameter SECPRFX=YES or SECPRFX=*prefix*.
2. If the CICS region is using MRO:
   a. Ensure that each connected CICS region is also using RACF security (SEC=YES).
   b. Specify ATTACHSEC=IDENTIFY on the CONNECTION definition for the TOR, to ensure that sign-on information is propagated from the TOR to the AORs.
3. Replace the default Db2 sign-on exit routine DSN3@SGN. This sign-on exit routine is driven during sign-on processing. The default sign-on exit routine does not support secondary authorization IDs, so you need to replace it with the sample sign-on exit routine DSN3SSGN, which is provided with Db2, or with your own routine. DSN3SSGN is shipped in source form in the Db2 SDSNSAMP library, and you can use it as a basis for your own routine. For information about sign-on exits and about writing exit routines, see Securing Db2 in Db2 for z/OS product documentation. If the RACF list of groups option is not active, DSN3SSGN passes the name of the current connected group as the secondary authorization ID. If the RACF list of groups is active, DSN3SSGN obtains the names of all the groups to which the user is connected, and passes them to Db2 as secondary authorization IDs.
4. Use the AUTHTYPE attribute in the DB2ENTRY definition (for entry threads), or the AUTHTYPE or COMAUTHTYPE attribute in the DB2CONN definition (for pool threads or command threads), to specify the GROUP option as the authorization type for each type of thread for which you want to provide secondary authorization IDs. If surrogate user checking is in force for the CICS region (that is, the system initialization parameter XUSER is set to YES), you need to obtain special authority to perform operations involving Db2 authorization IDs. “Using surrogate security and AUTHTYPE security to control access to the authorization IDs that CICS provides to Db2” on page 320 tells you how to do this.

If you have successfully completed all the steps listed above, when a CICS transaction's thread TCB signs on to Db2 with the types of thread that you have defined with the GROUP option, the CICS user's user ID will be passed to Db2 as the primary authorization ID, and the user's RACF group or list of groups will be passed to Db2 as secondary authorization IDs. If you have not successfully completed all the steps, the CICS Db2 attachment facility will issue an authorization failed message.

As an alternative to providing the names of RACF groups as secondary authorization IDs to Db2, you can write your own sign-on exit routine that sets secondary IDs for CICS transactions. See Securing Db2 in Db2 for z/OS product documentation.

Authorizing users to access resources in Db2

Once the user has accessed the Db2 address space from a CICS transaction they might need permission to issue Db2 commands or execute a plan.

About this task

Access to the Db2 resources that a CICS user needs to issue Db2 commands or execute a plan is subject to security checking by the Db2 security mechanisms. You can choose to have this security checking carried out by:

- Db2 internal security.
- RACF, or an equivalent external security manager.
- Partly Db2, and partly RACF.

For more information about setting up RACF to perform security checking in the Db2 address space, see Securing Db2 in Db2 for z/OS product documentation.

If you are using RACF for some or all of the security checking in your Db2 address space, remember that CICS transactions that sign on to Db2 must provide an authorization ID. For more information, see “Providing authorization IDs to Db2 for CICS transactions” on page 326. CICS must also be using RACF (SEC=YES must be specified in the SIT). This is because when RACF is used for security checking in the Db2 address space, CICS needs to pass a RACF access control environment element (ACEE) to Db2. CICS can only produce an ACEE if it has RACF active, and only threads defined with the GROUP, SIGN or USERID option can pass the ACEE to Db2.

When the ACEE is passed to Db2, it is used by the Db2 exit DSNX@XAC, which determines whether RACF, or an equivalent non-IBM external security manager, or Db2 internal security is used for security checking. DSNX@XAC is driven when a transaction whose thread has signed on to Db2, issues API requests. You can modify DSNX@XAC. For more information, see Securing Db2 in Db2 for z/OS product documentation.

Db2, or the external security manager, performs security checking using the authorization IDs that the CICS transaction provided to Db2 when the thread that it was using signed on to Db2. The authorization IDs could be related to the individual CICS user (for example, the CICS user's user ID and the RACF groups to which the user is connected), or they could be related to the transaction (for example, the terminal ID or transaction ID), or they could be related to the whole CICS region. For more information, see “Providing authorization IDs to Db2 for the CICS region and for CICS transactions” on page 323.

Db2, or the external security manager, checks that you have given the authorization IDs permission to perform the relevant actions in Db2. You can give the authorization IDs this permission by using GRANT statements in Db2. For information about how to grant, and revoke, Db2 permissions for authorization IDs, see Securing Db2 in Db2 for z/OS product documentation.

Controlling users' access to Db2 commands

For CICS users, the first security checking related to Db2 commands is performed in the CICS address space, when the user tries to access a CICS transaction that issues Db2 commands. This could be DSNC, or a user-defined transaction that invokes DFHD2CM1 and runs an individual Db2 command.

About this task

“Controlling users' access to Db2-related CICS transactions” on page 322 describes how to control users' access to transactions that issues Db2 commands in the CICS address space.

When a user issues a Db2 command through a CICS transaction, they are also subject to Db2 security checking, which verifies that they are authorized to Db2 to issue the command. This security checking

uses the authorization IDs (primary or secondary) that the transaction has passed from CICS. “Providing authorization IDs to Db2 for the CICS region and for CICS transactions” on page 323 tells you how to choose these authorization IDs and provide them to Db2. For transactions that use DFHD2CM1 to issue Db2 commands, the authorization IDs are set by the COMAUTHID or COMAUTHTYPE attribute of the CICS region's DB2CONN definition. For other applications that issue Db2 commands, the authorization IDs are set by the AUTHID or AUTHTYPE attribute for the CICS region's resource definition for the type of thread used by the transaction (pool thread or entry thread). These attributes control the authorization ID, or type of authorization ID, that is passed to Db2 by a transaction that is using that type of thread.

Db2 commands are therefore subject to two security checks, one in the CICS address space and one in the Db2 address space. Figure 28 on page 331 illustrates the process.

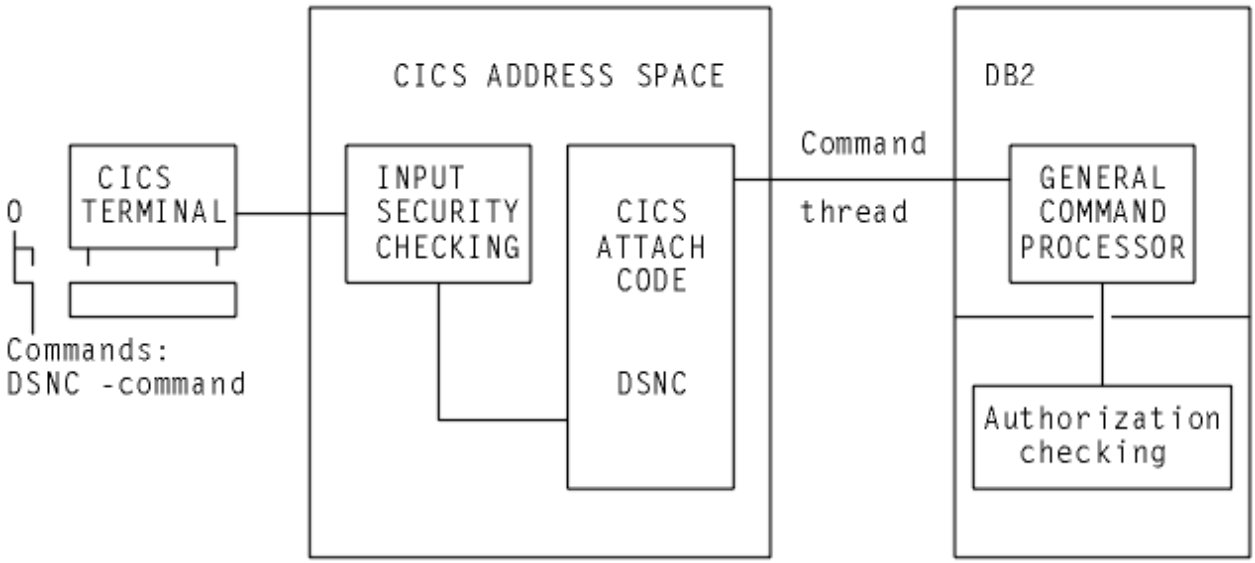


Figure 28. Security mechanisms for Db2 commands

In most cases, only a limited number of users are permitted to execute Db2 commands. A convenient solution can be to specify COMAUTHTYPE(USERID) on the DB2CONN definition, which resolves to the 8-byte CICS user ID as the authorization ID in Db2. Using this method, you can give different Db2 privileges explicitly to CICS user IDs. For example, you can use GRANT DISPLAY to give specific CICS user IDs permission to use only the -DIS command.

To authorize a user to issue a Db2 command, use a GRANT command to grant Db2 command privileges to the authorization ID that the transaction has passed from CICS. For information about how to grant, and revoke, Db2 permissions for authorization IDs, see Securing Db2 in Db2 for z/OS product documentation.

Controlling users' access to plans

A number of checks take place before a user can access plans in Db2. These checks start in the CICS address space before the request is passed to Db2 for further checks.

About this task

For Db2 commands, the first security check for users' access to plans takes place in the CICS address space, when CICS verifies that the user is permitted to access the transaction that will execute the plan. The second security check takes place in the Db2 address space, when Db2 verifies that the authorization ID provided by the transaction, is authorized to execute the plan. Figure 29 on page 332 illustrates this process.

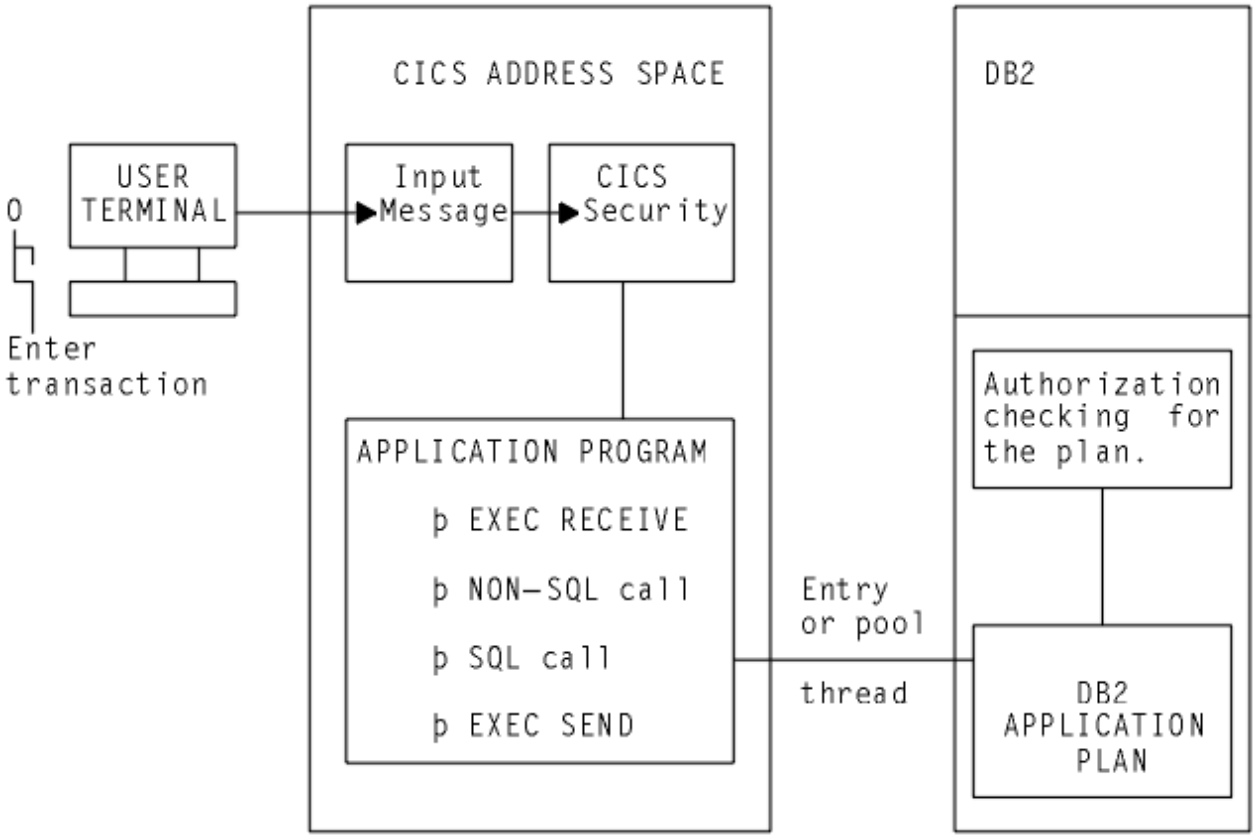


Figure 29. Security mechanisms for executing a plan

To authorize a user to execute a plan, use a GRANT command to grant Db2 command privileges to the authorization ID that the transaction has passed from CICS. For information about how to grant, and revoke, Db2 permissions for authorization IDs, see Securing Db2 in Db2 for z/OS product documentation.

If a plan includes dynamic SQL

When using static SQL, the binder of the plan must have the privileges needed to access the data, and the authorization ID passed from CICS to Db2 need only have the privileges to execute the plan.

About this task

However, if a plan includes the use of dynamic SQL, the authorization ID passed from CICS to Db2 must possess the privileges required to access all the Db2 resources involved, both the plan and the data. For example, if you specify AUTHTYPE(USERID), the CICS user ID must be granted Db2 privileges to the Db2 resources involved in the dynamic SQL. If this user ID is also a TSO user ID, it has access to the Db2 resources directly from SPUFI, QMF, and other utilities.

If you do not want to spend too much time granting Db2 privileges, where a transaction executes a plan that involves the use of dynamic SQL, consider using one of the following methods of supplying an authorization ID to Db2:

- Use the SIGN option on the AUTHTYPE attribute of the DB2ENTRY definition for the thread used by the transaction. This results in the transaction having the primary authorization ID that you specified in the SIGNID attribute of the DB2CONN definition for the CICS region. (This method is not suitable where RACF is used for security checking in the Db2 address space.)
- Use the AUTHID attribute of the DB2ENTRY definition for the thread used by the transaction, to specify a standard authorization ID. Use the same authorization ID for all the transactions that need to access dynamic SQL. (This method is not suitable where RACF is used for security checking in the Db2 address space.)

- Create a RACF group, and connect your CICS users to this RACF group. Use the GROUP attribute of the DB2ENTRY definition for the thread used by the transaction, so that the RACF group is one of the secondary IDs that is passed to Db2.

In each case, you can then grant Db2 privileges for Db2 resources involved in all dynamic SQL to a single ID, either the standard authorization ID from the DB2CONN definition or the AUTHID attribute, or the name of the RACF group. "Providing authorization IDs to Db2 for the CICS region and for CICS transactions" on page 323 tells you how to provide authorization IDs by all these methods.

Db2 multilevel security and row-level security

DB2® Version 8 introduced support for multilevel security. CICS does not provide specific support for multilevel security, but you can use CICS in a multilevel-secure environment provided that you take care with the configuration.

For more information about multilevel security, see the following publications:

- Securing Db2 in Db2 for z/OS product documentation
- z/OS Planning for Multilevel Security and the Common Criteria
- IBM Redbooks: Securing DB2 and Implementing MLS on z/OS

When multilevel security is implemented at row level (row-level security) for data in DB2 Version 8 or later, the RACF SECLABEL class is activated, and a set of security labels is defined for users and for the Db2 table rows. The RACF options SETR MLS and MLACTIVE are not required to be active. You can use Db2 row-level security without impact on the rest of the MVS system.

CICS is able to access Db2 rows secured in this way. For CICS, you need to ensure that the RACF user profile for a CICS user that needs access to the Db2 rows is defined in RACF to include a default SECLABEL. For details, see the z/OS Security Server RACF Security Administrator's Guide.

When a CICS user signs on to a CICS region with SEC=YES specified in the SIT, RACF associates the default SECLABEL with the RACF access control environment element (ACEE) for the user. The DB2ENTRY definition (or DB2CONN definition if the pool is being used) needs to specify AUTHTYPE=USERID or AUTHTYPE=GROUP, which ensures the ACEE is passed on to Db2 for further security checking. An individual CICS user can therefore only have one associated SECLABEL.

For non-terminal tasks or programs, such as PLT programs, if the PLTPIUSR system initialization parameter is not specified and the PLTPISEC=NONE system initialization parameter is specified, PLT programs are run under the CICS region userid. In this case, you need to define the CICS region userid with a default SECLABEL. If you need to define different SECLABELS for a transaction, you would need to run each transaction in a separate CICS region which has a different CICS region userid and associated SECLABEL.

Security for DBCTL

When you use CICS with DBCTL, several security facilities are available.

You can use one or more of the following optional security facilities:

- "PSB authorization checking by CICS" on page 334
- Resource access security checking by DBCTL
- DBCTL password security checking

For details about resource access security checking by DBCTL and DBCTL password security checking, see System administration in IMS product documentation.

Of the resources you can protect by using IMS security, you only need to be concerned about only with PSBs, databases, and commands.

PSB authorization checking by CICS

At PSB scheduling time, CICS invokes security checking to determine whether the terminal user is authorized to access the PSB. The actual check is carried out by an external security manager, which can be RACF or your own security program.

Although PSB scheduling requests are sent to DBCTL for processing, CICS does PSB authorization checking. For programming information about writing your own security program, see Invoking an external security manager.

Security for data sets: encryption

You can encrypt any data sets that you use with CICS for which z/OS data set encryption is supported. This includes user data sets that are accessed through CICS File Control APIs, queued sequential access method (QSAM) data sets used for CICS extrapartition transient data, basic sequential access method (BSAM) data sets used with CICS, and CICS system data sets that are appropriate candidates for encryption.

You can use data set encryption with any in-service release of CICS TS for z/OS.

Encrypted data sets must be storage management subsystem (SMS)-managed and extended format. To create an encrypted data set, you assign a key label to a data set when that new data set is allocated. The key label must point to an AES-256 bit encryption key in the integrated cryptographic service facility (ICSF) cryptographic key data set (CKDS) that will be used to encrypt or decrypt the data. The key label is not sensitive information, but the encryption key that it identifies is.

Encryption support for CICS user data sets

For user data sets defined to CICS, encryption support includes key-sequenced data sets (KSDS), entry-sequenced data sets (ESDS), relative record data sets (RRDS), and variable relative record data sets (VRRDS), accessed through base VSAM and VSAM Record Level Sharing (RLS). Encryption is also supported for the backing VSAM key-sequenced data sets that are used for shared data tables or coupling facility data tables.

Encryption support for CICS system data sets

You can encrypt any CICS system data sets for which encryption is supported, but some system data sets are good candidates for encryption, whereas others are not.

- Data sets with the *potential* to contain sensitive data are candidates for encryption. You must assess which of your data sets might contain sensitive data and whether to encrypt them.
- Data sets that do not, and are unlikely to, contain sensitive data are not candidates for encryption.

The following table lists the CICS system data sets for which encryption is possible, and whether they are candidates for encryption. If you want to use an "encrypt everything" approach, all the data set types that are listed in this table can be encrypted.

Table 48. Which system data sets are candidates for encryption?

CICS system data set	Candidate for encryption?	Special considerations
Temporary storage data set (DFHTEMP)	Yes, could contain sensitive data.	DFHTEMP supports extended format, but does **not** support extended addressing.
Intrapartition Transient Data (DFHINTRA)	Yes, could contain sensitive data.	DFHINTRA supports extended format, but does **not** support extended addressing.
Extrapartition Transient Data	Yes, could contain sensitive data.	Encryption is not supported for partitioned data sets (PDS).

Table 48. Which system data sets are candidates for encryption? (continued)

CICS system data set	Candidate for encryption?	Special considerations
Auxiliary Trace data sets (DFHAUXT and DFHBUXT)	Yes, can potentially include sensitive data in the diagnostics. Alternatively, use the CONFDATA system initialization parameter, which might provide sufficient protection.	If trace data is encrypted and you need to send it to IBM for diagnostics, use CICS trace formatting or another method to ensure that you send decrypted data.
CICS dump data sets (DFHDMPA and DFHDMPB)	Yes, can potentially include sensitive data in the diagnostics. Alternatively, use the CONFDATA system initialization parameter, which might provide sufficient protection.	If dump data is encrypted and you need to send it to IBM for diagnostics, use CICS dump formatting or another method to ensure that you send decrypted data. CICS dump data sets support extended format, but do **not** support extended addressing.
Doctemplate resources	Yes, can potentially contain sensitive data.	Provided that the data set used is of a type for which encryption is supported.
URIMAP resources used for static delivery	Yes, can potentially contain sensitive data.	Provided that the data set used is of a type for which encryption is supported.
BTS repository data sets and BTS local request queue (LRQ) data set	No, contain only control data.	None.
Global and Local catalog data sets (DFHGCD and DFHLCD)	No, contain only configuration data.	Consider only if you believe that CICS configuration data is sensitive information.
CICS system definition data set (DFHCSD)	No, contains only information about resource configuration.	Consider only if you believe that your resource definitions include any sensitive information.
CMAC messages data set (DFHCMACD)	No, contains only message details.	Consider only if you added your own messages that you believe contain sensitive data.
zFS files used for bundle definitions and web services	No, contain only configuration information.	Consider only if you believe that your bundle definitions include any sensitive information.

Related information

Encrypting data sets

Chapter 10. Security for external interfaces

Security for EXCI

CICS applies security checks in a number of ways against requests received from an MVS client program.

Using MRO logon and bind-time security

DFHIRP, the CICS interregion communication program, performs two security checks against users that want to either log on to IRP (specific connections only), or connect to a CICS region (also referred to as bind-time security).

About this task

Generic EXCI connections: The discussion about logon security checking in this section applies only to EXCI connections that are defined as SPECIFIC. The MRO logon security check is not performed for generic connections.

The MVS client program is treated just the same as another CICS region as far as MRO logon and connect (bind-time) security checking is concerned. This means that when the client program logs on to the interregion communication program, IRP performs logon and bind-time security checks against the user ID under which the client program is running. In the remainder of this information, this user ID is called the user ID of the batch region.

To enable your client program to log on to IRP successfully, and to connect to the target server region, first ensure that you define the user ID of the batch region in a user profile to RACF. After you define the user ID of the batch region to RACF, you can then give the batch region the appropriate logon and bind-time authorizations.

Procedure

- Logon authorization

  Authorize the user ID of the batch region to the DFHAPPL.*user_name* RACF FACILITY class profile, with UPDATE authority. The *user_name* part of the profile name is the user name defined on the INITIALIZE_USER command.

  Failure to authorize the user ID of the batch region to the DFHAPPL profile of the specific user ID logging on to IRP causes Allocate_Pipe processing to fail with RESPONSE(SYSTEM_ERROR) REASON(IRC_LOGON_FAILURE). The subreason field-1 for a logon security check failure returns decimal 204.

  See “Defining DFHAPPL FACILITY class profiles for an EXCI region” on page 338 for information about FACILITY class profiles for an EXCI client program.

- Bind-time authorization

  Authorize the user ID of the batch region to the DFHAPPL.*applid* RACF FACILITY class profile of the target CICS server region, with READ authority.

  Failure to authorize the user ID of the batch region to the DFHAPPL.*applid* profile of the CICS server region causes Open_Pipe processing to fail with RESPONSE(SYSTEM_ERROR) REASON(IRC_CONNECT_FAILURE). The subreason field-1 for a bind-time security check failure returns decimal 176.

  See Bind security for information about the MRO logon and bind-time security checks, and for examples of how to define the RACF DFHAPPL profiles.

Defining DFHAPPL FACILITY class profiles for an EXCI region

Define the *user_name* part of the DFHAPPL profile name as follows:

- For the EXCI CALL interface, the *user_name* must be the name you specify on the *user_name* parameter of the INITIALIZE_USER command.

  Define FACILITY class profiles, with appropriate authorizations, for each user name specified in a client program if the program has INITIALIZE_USER commands for more than one user name.

  For example, if the *user_name* defined on an INITIALIZE_USER command is DCEUSER1, define the DFHAPPL profile in the FACILITY class as follows:

  ```
  RDEFINE FACILITY (DFHAPPL.DCEUSER1) UACC(NONE)
  ```

  If the batch region's user ID is CLIENTA, authorize the batch region to log on to IRP as follows:

  ```
  PERMIT DFHAPPL.DCEUSER1  CLASS(FACILITY) ID(CLIENTA)
         ACCESS(UPDATE)
  ```

- For the EXEC CICS LINK command, the *user_name* is preset by the external CICS interface as DFHXCEIP. This does not require authorization for IRP logon, because the EXEC CICS LINK interface uses a generic connection to which the logon security check does not apply.

Link security

The target CICS server region performs link security checking against requests from the client program.

These security checks cover transaction attach security (when attaching the mirror transaction), and resource and command security checking within the server application program. The link user ID that CICS uses for these security checks is the batch region's user ID.

To ensure these link security checks do not cause security failures, you must ensure that the link user ID is authorized to the following resource profiles, as appropriate:

- The profile for the mirror transaction, either CSMI for the default, or the mirror transaction specified on the *transid* parameter. This is required for transaction attach security checking.
- The profiles for all the resources accessed by the CICS server application program—files, queues (transient data and temporary storage), programs, and so on. This is required for resource security checking.
- The CICS command profiles for the SPI commands issued by the CICS server application program—INQUIRE, SET, DISCARD and so on. This is required for command security checking.

See Bind security for information about MRO link security checking.

User security

The target CICS server region performs user security checking against the user ID passed on a DPL_ Request call. User security checking is performed only when connections specify ATTACHCSEC(IDENTIFY).

User security is performed in addition to any link security.

For user security, in addition to any authorizations you make for link security, you must also authorize the user ID specified on the DPL_Request call.

Note that there is no provision for specifying a user ID on the EXEC CICS LINK command. In this case, the external CICS interface passes the batch region's user ID. User security checking is therefore performed against the batch region's user ID if the connection definition specifies ATTACHSEC(IDENTIFY).

Note: If your connection resource definitions for the external CICS interface specify ATTACHSEC(IDENTIFY), your server programs will fail with an ATCY abend if you run them in an environment that does not have RACF, or an equivalent external security manager (ESM), installed and active.

If you want to run external CICS interface server programs without any security active, you must specify ATTACHSEC(LOCAL).

Surrogate user checking

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

EXCI client jobs are subject to surrogate user checking. You must authorize the batch region's user ID as a surrogate of the user ID specified on all DPL_Request calls. This means the batch region's user ID must have READ access to a profile named *execution-userid*.DFHEXCI in the SURROGAT general resource class (where *execution-userid* is the user ID specified on the DPL call). For example, the following commands define a surrogate profile for a DPL user ID, and grant READ access to the EXCI batch region:

```
RDEFINE  SURROGAT execution_userid.DFHEXCI UACC(NONE)
PERMIT execution_userid.DFHEXCI CLASS(SURROGAT) ID(batch_region_userid) ACCESS(READ)
```

If no user ID is specified on the DPL_Request call, no surrogate user check is performed because the user ID on the DPL_Request call defaults to the batch region's user ID. For this bypass of surrogate user checking to be successful, ensure that you have correctly omitted the user ID on the DPL_Request call. See the example of EXCI CALLs with null parameters in The EXCI CALL interface for the correct way to specify a null pointer when omitting an EXCI call parameter.

If the batch region's user ID and the CICS region user ID are different, link security checking is enforced. With link security, a non-authenticated user ID passed on a DPL_Request call cannot acquire more authority than that allowed by the link security check. It can acquire only the same, or less, authority than that allowed by the link security check.

For more information about CICS security, see CICS TS security.

Security for ONC RPC

Important: This information contains Product-sensitive Programming Interface and Associated Guidance Information.

Security is an important concern in the provision of ONC RPC support in the CICS environment, because CICS ONC RPC provides an Open Systems communications interface into CICS.

ONC RPC has its own security methods (called authentication in RPC) with dedicated fields in the ONC RPC call and reply message headers. There are three types of RPC authentication:

- **UNIX authentication**, which is used to transmit the client's UNIX user ID, group ID, and other identification information.
- **Data Encryption Standard (DES) authentication**, which is not available at ONC RPC Version 3.9, and so cannot be used with CICS ONC RPC.
- **Null authentication**, which offers no security checking.

This section describes how CICS ONC RPC interacts with the security facilities of ONC RPC and CICS.

Security in ONC RPC

ONC RPC has its own security methods (called authentication in RPC) with dedicated fields in the ONC RPC call and reply message headers.

There are three types of RPC authentication:

- **UNIX authentication**, which is used to transmit the client’s UNIX user ID, group ID, and other identification information.
- **Data Encryption Standard (DES) authentication**, which is not available at ONC RPC Version 3.9, and so cannot be used with CICS ONC RPC.

- **Null authentication**, which offers no security checking.

Security in CICS and its effect on CICS ONC RPC operations

During the operation of CICS ONC RPC, various CICS commands are used to make security checks with an external security manager (ESM).

The checks will always give positive results if SEC=NO is specified as a system initialization parameter. The checks will always give negative results if SEC=YES was specified, but the ESM abended while CICS was operating. The following discussion of the use made of CICS security commands assumes that SEC=YES is specified, and that the ESM is active.

- When a transaction whose user ID is *userid1* issues EXEC CICS START USERID(*userid2*), a surrogate-user check is made with the ESM to see that *userid1* is authorized to use *userid2*. The check is made only if XUSER=YES is specified as a system initialization parameter.

  This command is issued when the connection manager starts the server controller, and each time the server controller starts an alias transaction. In the first case, the user ID used is the one supplied to the connection manager as CRPM Userid on panel DFHRP02. In the second case, the user ID used is the one output from **Decode**.
- EXEC CICS VERIFY PASSWORD is issued by the alias before it links to the CICS program that services the client request. A check is made with the ESM that the user ID and password are an acceptable combination.
- EXEC CICS QUERY SECURITY is used by the alias to check that the user ID under which it is executing is authorized to use the CICS program. The check is made only if XPPT=YES is specified as a system initialization parameter.
- During the operation of the CICS program, security checks are made each time the program tries to access a protected resource. The check is made only if RESSEC(YES) is specified in the definition of the alias transaction, and the system initialization parameter controlling security checking for the resource type is set to YES.
- During the operation of the CICS program, security checks are made each time the program tries to use a command from the CICS SPI (system programming interface). The check is made only if CMDSEC(YES) is specified in the definition of the alias transaction, and if XCMD=YES is specified as a system initialization parameter.

Figure 30 on page 341 shows how CICS security interacts with the operation of CICS ONC RPC.

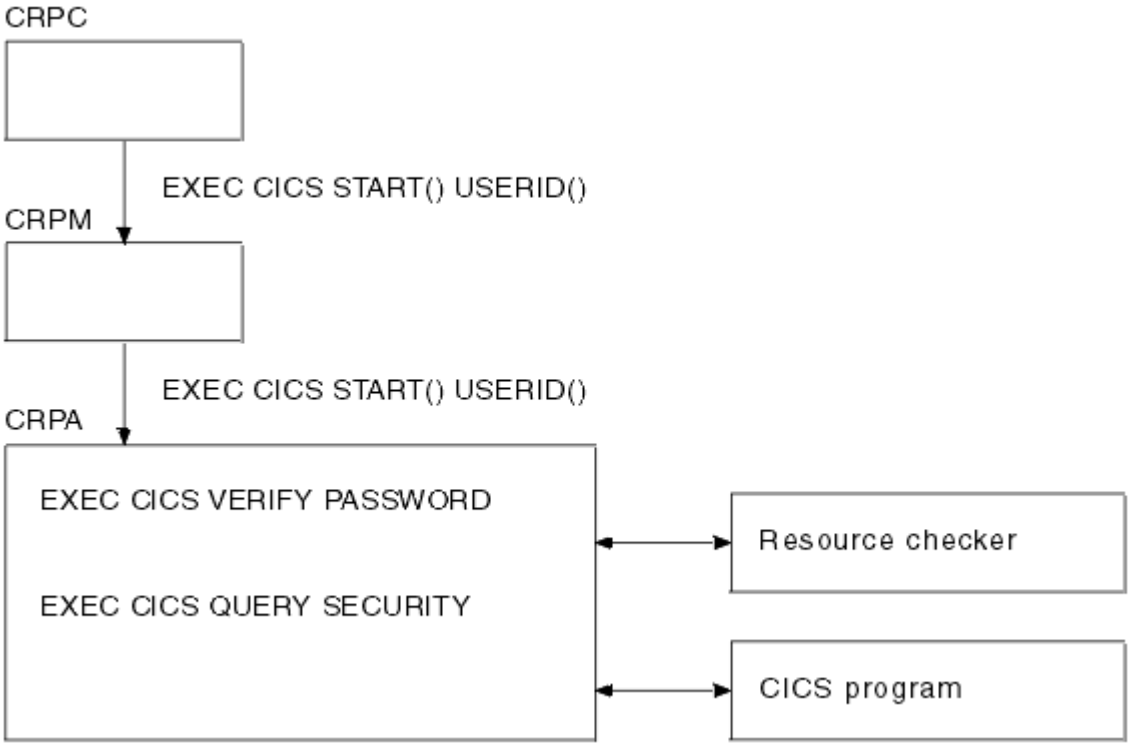


Figure 30. How CICS security interacts with CICS ONC RPC operations

The figure shows that the alias will link to the user-supplied resource checker program if one is configured, but the use of the resource checker program is not recommended. You should use the CICS security facilities, and make the appropriate definitions in the ESM.

RACF Secured Sign-on for ONC RPC clients

RACF Secured Sign-on support allows RPC clients to gain security access to CICS facilities by sending a PassTicket. This avoids the security hazard of a password being transmitted across the network in clear text.

For further information, see z/OS Security Server RACF System Programmer's Guide. This includes details of the algorithm that the RPC client must use to generate the PassTicket. This algorithm includes the DES algorithm.

PassTicket generation for ONC RPC clients

The algorithm that generates the PassTicket for an ONC RPC client is a function of the following items:

- The CICS user ID of the client.
- The CICS application ID of the CICS region running CICS ONC RPC.
- A secured sign-on application key, known to both sides.
- A time and date stamp.

To generate the PassTicket, the RPC client must:

- Know its CICS user ID, the server CICS application ID, and the application key.
- Synchronize its clock to within ten minutes of the server.
- Have access to the encryption algorithm on its machine. Only the DES algorithm may be used.

Writing the resource checker

Your resource checker program must be called DFHRPRSC. There can be only one resource checker in a CICS region.

The resource checker allows you to check the credentials of inbound client requests.

The resource checker can check the client address, passed as an input parameter, against a list of known clients for the host on which the request has been received. The password passed to the resource checker is blank.

Reference information for the resource checker

The resource checker is optionally invoked by the alias before it attempts to link to the CICS program that is to service the client request. It must say whether the client request is allowed to proceed.

The reference information for the resource checker is presented as follows:

- A summary table of parameters, showing which are for input only, and which for output only.
  - **Input** is for parameters that your resource checker may consult, but not change.
  - **Output** is for parameters that your resource checker must not consult, but may change.
- A description of the processing that the resource checker is expected to do.
- A list of parameters in alphabetical order, with a description of how CICS ONC RPC sets up the inputs, and what use it makes of the outputs.
- A list of the responses and reason codes that the resource checker can return, with a description of the action that CICS ONC RPC takes for each response and reason code.

The descriptions give the names of the program elements as they appear in C. In COBOL the names are all in uppercase, and the underscores are replaced by hyphens.

Summary of parameters

The format of the communication area containing the resource checker parameters is in the C header file DFHRPRDH, and the COBOL copybook DFHRPRDO. You will also need values defined in the C header file DFHRPUCH, or in the COBOL copybook DFHRPUCO.

Input	Output
res_check_alias_transid **res_check_cics_password_ptr** **res_check_cics_userid** **res_check_client_ip_address** **res_check_eyecatcher** **res_check_host_ip_address** **res_check_server_program_name**	**res_check_reason** **res_check_response**

Parameters

res_check_alias_transid
(Input only)

The 4-character name of the alias transaction that has linked to the resource checker.

res_check_cics_password_ptr
(Input only)

A pointer to the 8-character password passed from the requesting client or supplied by **Decode**. The value of this field is blank, and it is provided for compatibility with earlier versions of CICS ONC RPC.

res_check_cics_userid
(Input only)

The 8-character CICS user ID under which the alias is running.

res_check_client_ip_address
(Input only)

The fullword internet address of the client.

res_check_eyecatcher
(Input only)

A string of length 8. (Its value is defined in the header file DFHRPUCH and the copybook DFHRPUCO).

res_check_host_ip_address
(Input only)

The fullword internet address of the z/OS Communications Server host with which the server controller is in communication.

res_check_reason
(Output only)

The reason to be returned to the alias.

res_check response
(Output only)

The response to be returned to the alias.

res_check_server_program_name
(Input only)

The 8-character name of the CICS program that is to be invoked to perform the server function requested by the client.

Response and reason codes

You must return one of the following values in the **res_check_response** field.

URP_OK
The alias will continue to process the client request.

URP_EXCEPTION
The alias writes an exception trace entry (trace point 9F0E), and issues a message that depends on the reason code:

- URP_AUTH_BADCRED—message DFHRP0130

  An **svcerr_auth** call with a why-value of AUTH_BADCRED is used to send a reply to the client.
- URP_AUTH_TOOWEAK—message DFHRP0184

  An **svcerr_auth** call with a why-value of AUTH_TOOWEAK is used to send a reply to the client.
- Any other value—message DFHRP0185

  An **svcerr_systemerr** call is used to send a reply to the client.

URP_INVALID
The alias writes an exception trace entry (trace point 9F0E), and issues a message (DFHRP0186).

An **svcerr_systemerr** call is used to send a reply to the client.

URP_DISASTER
The alias writes an exception trace entry (trace point 9F0E), and issues a message (DFHRP0187).

An **svcerr_systemerr** call is used to send a reply to the client.

If you return any other value in **res_check_response**, the alias writes an exception trace entry (trace point 9F0E), and issues a message (DFHRP0188). An **svcerr_systemerr** call is used to send a reply to the client.

You can supply a 32-bit reason code in conjunction with the response value to provide further information in error cases. CICS ONC RPC does not take any action on the reason code returned by the resource checker, except as indicated previously under URP_EXCEPTION. The reason code is output in any trace or messages that result from the resource checker, and you may use it as a debugging aid.

See Numeric values of response and reason codes for the numeric values of the response and CICS-defined reason codes in trace output.

Chapter 11. Security for Java applications

You can secure Java applications to ensure that only authorized users can deploy and install applications, and access those applications from the web or through CICS. You can also use a Java security manager to protect the Java application from performing potentially unsafe actions.

You can add security at different points in the Java application lifecycle:

- Implement security checking for defining and installing Java application resources. Java applications are packaged in CICS bundles, so you must ensure that users who are allowed to install applications in the JVM server can install this type of resource.
- Implement security checking for application users to ensure that only authorized users can access an application.
- Implement security checking for CICS Java tasks that are started using the CICSExecutorService. All such CICS tasks run under the CJSA transaction and the default user ID.
- Implement security restrictions on the Java API by using a Java security manager.

Java applications can run in an OSGi framework or a Liberty server. Liberty is designed to host web applications and includes an OSGi framework. The security configuration for a Liberty server is different, because Liberty has its own security model.

To configure security for OSGi applications, use CICS resource security to authorize which users can manage the lifecycle of the JVMSERVER and the Java applications. Use CICS transaction security to determine who can access the application.

Configuring security for OSGi applications

Use CICS resource security to authorize which users can manage the lifecycle of the JVMSERVER and the Java applications. Use CICS transaction security to determine who can access the application.

Procedure

- Authorize application developers and system administrators to create, view, update, and remove JVMSERVER and BUNDLE resources as appropriate. The JVMSERVER resource controls the availability of the JVM server. The BUNDLE resource is a unit of deployment for the Java application and controls the availability of the application.
- Authorize users to run the application by ensuring the relevant user ID is allowed to attach the transaction under which the application will run.

Results

You have successfully configured security for Java applications that run in an OSGi framework.

Configuring security for a Liberty JVM server

You can use the CICS Liberty security feature to authenticate users and authorize access to web applications through Java Platform, Enterprise Edition roles (Java EE roles), providing integration with CICS transaction and resource security. You can also use CICS resource security to authorize the appropriate users to manage the lifecycle of both the JVMSERVER resource and Java web applications that are deployed in a CICS BUNDLE resource. In this topic, authentication verifies the identity of a given user, typically by requiring the user to enter a username and password. Authorization then grants access control permissions based on the identity of the authenticated user.

Before you begin

1. Ensure that the CICS region is configured to use SAF security and is defined with SEC=YES as a system initialization parameter. If CICS security is turned off (SEC=NO), you can still use Liberty security by manually configuring the `server.xml` file as described in "6" on page 347.
2. Authorize application developers and system administrators to create, view, update, and remove JVMSERVER and BUNDLE resources to deploy web applications into a Liberty JVM server.

The JVMSERVER resource controls the availability of the JVM server, and the BUNDLE resource is a unit of deployment for the Java applications and controls the availability of the applications. The default behavior of the CICS TS security feature, `cicsts:security-1.0`, is to use the SAF registry. If you use an LDAP registry, a SAF registry is not created. For more information, see Configuring security for a Liberty JVM server by using distributed identity mapping. The basic user registry (which is also used by `quickStartSecurity`) is only suitable for simple security testing. Be aware that if you configure and run with basic user registry and you need to switch to `cicsts:security-1.0`, you need to delete the session tokens.

About this task

This task explains how to configure security for a Liberty JVM server and integrate Liberty security with CICS security. For information about how to configure security for Link to Liberty, see Linking to Java applications in a Liberty JVM server by using the @CICSProgram annotation. For guidance on configuring security for the JCICSX remoting server, see "Configuring security for remote JCICSX API development" on page 367.

The default transaction ID for running web requests is CJSA. However, you can configure CICS to run web requests under a different transaction ID by using a URIMAP of type JVMSERVER. Typically, you might specify a URIMAP to match the generic context root (URI) of a web application to scope the transaction ID to the set of servlets that make up the application. Or you might choose to run each individual servlet under a different transaction with a more precise URI.

Calls to the JCICSX Liberty JVM server are run under transaction CJXA.

The default user ID for running web requests is the CICS default user ID. If a URIMAP is available and contains a static user ID, it is used in preference to the default user ID. If the web request contains a user ID in its security header, it takes precedence over all other mechanisms.

Tasks starting from Liberty that are not classified as web requests run under the CJSU transaction by default. Although there is no URIMAP style mechanism for these types of tasks, you can override the default transaction ID by using the JVM profile property of `com.ibm.cics.jvmserver.unclassified.tranid` and the default user ID by using the JVM profile property `com.ibm.cics.jvmserver.unclassified.userid`.

Note: The user ID requires permission to attach the specified transaction. For more information, see Transaction security.

Procedure

1. Configure the Liberty angel process to provide authentication and authorization services to the Liberty JVM server, see The Liberty server angel process.

   Tip: If you have a named angel process, you need to configure your Liberty JVM server to connect to it by adding the following line to your JVM profile.

   ```
   -Dcom.ibm.ws.zos.core.angelName=<named_angel>
   ```

2. Optional: Enforce the requirement to connect to the Liberty angel process when the Liberty JVM server is being enabled by adding the following line to your JVM profile:

   ```
   -Dcom.ibm.ws.zos.core.angelRequired=true
   ```

   This option prevents the Liberty JVM server from starting if the angel process is unavailable.

It instructs CICS to call the Liberty angel check API to verify whether an angel process is available for Liberty JVM server startup.

If the angel process is unavailable, CICS reacts as follows:

- If the Liberty JVM server is being enabled through the CEMT transaction, a message is issued, and the Liberty JVM server is disabled.
- If the Liberty JVM server is being enabled by the **SET JVMSERVER** SPI command or by using the CMCI through the CICS Explorer, a message is issued, and the Liberty JVM server is disabled.
- If the Liberty JVM server is being enabled by the CICS CREATE SPI, by BAS, or from GRPLIST, a message is issued, and CICS will wait 30 seconds before retrying the Liberty angel check API call. If the angel process is unavailable on the fifth attempt, a WTOR message is issued, giving the operator the option to continue waiting or to disable the JVMSERVER resource.

3. Add the `cicsts:security-1.0` feature to the `featureManager` list in the `server.xml`,

```
<featureManager>
   ...
   <feature>cicsts:security-1.0</feature>
</featureManager>
...
```

4. Add the System Authorization Facility (SAF) registry to `server.xml` by using the following example:

```
<safRegistry id="saf" enableFailover="false"/>
```

5. Save the changes to `server.xml`.
6. Optional: Alternatively, if you are autoconfiguring the Liberty JVM server and the **SEC** system initialization parameter is set to YES in the CICS region, the Liberty JVM server is dynamically configured to support Liberty JVM security when the JVM server is restarted. For more information, see Configuring a Liberty JVM server.

   If the **SEC** system initialization parameter is set to NO, you can still use Liberty security for authentication or SSL support. If CICS security is turned off, and you want to use a Liberty security, you must configure the `server.xml` file manually:

   a. Add the `appSecurity-2.0` feature to the `featuremanager` list.
   b. Add a user registry to authenticate users. Liberty security supports SAF, LDAP, and basic user registries. For more information, see Configuring a user registry in Liberty.
   c. Add security-role definitions to authorize access to application resources, see “Authorizing users to run applications in a Liberty JVM server” on page 354.

Results

The web container is automatically configured to use the z/OS Security feature of Liberty. A SAF registry is used for authentication, and Java EE roles are respected for authorization. Authorization constraints and security roles govern who can access the application. These are usually defined in the deployment descriptor (`web.xml`) of the application, but might also be defined as security annotations in the source-code. Typically, users and groups are mapped to roles by the applications *<application-bnd>* element in `server.xml`. Alternatively, if the*<safAuthorization>* element is configured in `server.xml`, the mappings are held in SAF (as EJBROLEs in RACF).

What to do next

Note: You can also delegate authentication to another identity by configuring the RunAs specification for Liberty, see Configuring RunAs authentication in Liberty.

- Configure Liberty application security authentication rules; see “Authenticating users in a Liberty JVM server” on page 352.
- Define authorization rules for web applications; see “Authorizing users to run applications in a Liberty JVM server” on page 354 and “Authorization using SAF role mapping” on page 358.
- Modify the Liberty authentication cache.

For more information about using Secure Sockets Layer (SSL), see “Configuring SSL (TLS) for a Liberty JVM server using a Java keystore” on page 372.

The Liberty angel process

The Liberty angel process is a started task that allows Liberty servers to use z/OS authorized services. It's long-lived and can be shared among your multiple Liberty servers. When you include the `cicsts:security-1.0` feature, the CICS Liberty JVM server uses the angel process to call z/OS authorized services such as System Authorization Facility (SAF).

Named angels

A Liberty server can only connect to one angel process at server startup. However, all Liberty servers that are running on a z/OS image can share a single angel process. This is regardless of the level of code that the servers are running or whether they are running in a CICS JVM server. To achieve this, you need to use named angels.

If an angel process is not given a name, it becomes the default angel process. You can have only one default angel process. If you try to create another, it fails to start.

Optionally, you can name an angel process. Named angels allow multiple uniquely named angel processes to run on a single z/OS system, in addition to the default unnamed angel process.

A named angel has the same function as the default angel process, but it can be used for a selected group of Liberty servers. This provides the ability to isolate servers from one another, so that they can run different service levels or be managed independently.

For more information about named angel processes, see Named angel.

Angel version interoperability

All Liberty servers that are running on a z/OS image can share a single angel process, regardless of the level of Liberty code that the servers are using. It's recommended that the angel process be upgraded before the Liberty servers that use its services, because it provides back-level support for earlier versions of Liberty servers. This ensures support is available for all authorized services potentially required by the Liberty servers.

Important: Install the latest version of the angel process, regardless of which product it is bundled with. The latest version might be bundled with other IBM software, and might supersede the version that is bundled with CICS.

You can identify the version of Liberty for the angel process and the Liberty JVM server that's running in CICS as shown in “Examples of identifying Liberty versions” on page 351.

Running the angel process started task

1. Locate the JCL procedure for the **started** task in the USSHOME directory, for example: `/usr/lpp/cicsts56/wlp/templates/zos/procs/bbgzangl.jcl`
2. Modify and copy the JCL procedure to a JES procedure library. You can set **ROOT** to the value of USSHOME/`wlp`, for example: ROOT=`/usr/lpp/cicsts56/wlp`
3. Start the angel process. In the following examples, `[.identifier]` indicates an optional identifier that can be up to 8 characters.
   a. To start the angel process without naming it, use the following command:

      ```
      START BBGZANGL[.identifier]
      ```

   b. To start the angel process as a named angel process, code the NAME parameter on the operator START command. For example:

      ```
      START BBGZANGL[.identifier],NAME=<named_angel>
      ```

The angel process name is 1 - 54 characters inclusive, and must use only the following characters: `A-Z 0-9 ! # $ + - / : < > = ? @ [ ] ^ _ ` { } | ~`

Note: A Liberty server can use its own named angel process. One benefit of this isolation is that the angel process can be serviced without affecting any other Liberty server instances on the LPAR. The angel process must be running before the Liberty JVM server starts.

4. Start the Liberty JVM server. By default, the server connects to the unnamed angel process if one is available. To connect to a specific angel process, set the `com.ibm.ws.zos.core.angelName` property, for example:

```
-Dcom.ibm.ws.zos.core.angelName=named_angel
```

5. You can specify that CICS checks for the presence of a running angel process before enabling, by setting the `com.ibm.ws.zos.core.angelRequired` property to true. For example:

```
-Dcom.ibm.ws.zos.core.angelRequired=true
```

The server fails if the angel process is not available during startup. Use of this property allows a quicker and cleaner failure.

Interacting with the angel process started task

In the following examples, `[.identifier]` indicates an optional identifier that can be up to eight characters.

- Display the Liberty JVM servers that are connected to the angel process use the following console command:

```
MODIFY BBGZANGL[.identifier],DISPLAY,SERVERS,PID
```

A list of job names and process identifiers (PID) are displayed:

```
15.48.45 STC82204  CWWKB0067I ANGEL DISPLAY OF ACTIVE SERVERS
15.48.45 STC82204  CWWKB0080I ACTIVE SERVER ASID 5c JOBNAME IYK3ZNA1 PID 83953428
15.48.45 STC82204  CWWKB0080I ACTIVE SERVER ASID 5c JOBNAME IYK3ZNA1 PID 33621002
```

Each Liberty JVM server runs under a unique PID, and is returned by the CICS command `INQUIRE JVMSERVER`.

- Stop the angel process.

```
STOP BBGZANGL[.identifier]
```

Note: The Liberty JVM server must be stopped before restarting or applying maintenance to the angel process.

SAF profiles used by the angel process

This section describes the SAF profiles to which access is required for CICS processing. For information on the full set of SAF profiles defined by Liberty, refer to Enabling z/OS authorized services on Liberty for z/OS.

- The Liberty JVM server runs under the authority of the CICS region user ID. This user ID must be able to connect to the angel process to use authorized services. The user ID that the angel process runs under needs access to the SAF STARTED profile, for example:

```
RDEFINE STARTED BBGZANGL.* UACC(NONE) STDATA(USER(WLPUSER))
SETROPTS RACLIST(STARTED) REFRESH
```

- For the Liberty JVM server to connect to an angel process, create a profile for the angel (**BBG.ANGEL**, or **BBG.ANGEL.<namedAngelName>** if you are using a named angel process) in the **SERVER** class. Give the CICS region user ID (*cics_region_user*) authority to access it, for example, in RACF:

```
RDEFINE SERVER BBG.ANGEL UACC(NONE)
PERMIT BBG.ANGEL CLASS(SERVER) ACCESS(READ) ID(cics_region_user)
```

- For a Liberty server to use the z/OS authorized services, create a **SERVER** profile for the authorized module **BBGZSAFM** and give the CICS region user ID (*cics_region_user*) to the profile:

```
RDEFINE SERVER BBG.AUTHMOD.BBGZSAFM UACC(NONE)
PERMIT BBG.AUTHMOD.BBGZSAFM CLASS(SERVER) ACCESS(READ) ID(cics_region_user)
```

- Give the Liberty JVM server, under the authority of the CICS region user ID (*cics_region_user*), access to the SAF user registry and SAF authorization services (**SAFCRED**) in the **SERVER** class. For example, in RACF:

```
RDEFINE SERVER BBG.AUTHMOD.BBGZSAFM.SAFCRED UACC(NONE)
PERMIT BBG.AUTHMOD.BBGZSAFM.SAFCRED CLASS(SERVER) ACCESS(READ) ID(cics_region_user)
```

- Create a **SERVER** profile for the **IFAUSAGE** services (**PRODMGR**) and allow the CICS region user ID access to it. This allows the Liberty JVM server to register and unregister from **IFAUSAGE** when the CICS JVM server is enabled and disabled:

```
RDEFINE SERVER BBG.AUTHMOD.BBGZSAFM.PRODMGR UACC(NONE)
PERMIT BBG.AUTHMOD.BBGZSAFM.PRODMGR CLASS(SERVER) ACCESS(READ) ID(cics_region_user)
```

- Refresh the **SERVER** resource:

```
SETROPTS RACLIST(SERVER) REFRESH
```

The following table summarizes the SAF security profiles that are used by a Liberty server running in a CICS JVM server.

Table 49. SAF profile table for CICS Liberty security

Class	Profile	Required for	CICS region user ID 1	Unauthenticated user ID 2	Authenticated user ID 3
SERVER	BBG.ANGEL	Angel process registration at Liberty server startup	READ		
SERVER	BBG.ANGEL.<namedAngelName>	Angel process registration at Liberty server startup	READ		
SERVER	BBG.AUTHMOD.BBGZSAFM	Angel process registration at Liberty server startup	READ		
SERVER	BBG.AUTHMOD.BBGZSAFM.SAFCRED	Angel process registration at Liberty server startup	READ		
SERVER	BBG.AUTHMOD.BBGZSAFM.PRODMGR	Angel process registration at Liberty server startup	READ		

Table 49. SAF profile table for CICS Liberty security (continued)					
Class	**Profile**	**Required for**	**CICS region user ID 1**	**Unauthenticated user ID 2**	**Authenticated user ID 3**
SERVER	BBG.SECPFX.BBGZDFLT 4	Authentication or authorization	READ		
APPL	BBGZDFLT 4	Authentication or authorization		READ	READ
EJBROLE	BBGZDFLT.<resource>.<role> 5	Authentication or authorization			READ

1. User ID that is associated with the CICS job or started task.
2. User ID used for unauthenticated requests in Liberty. The value is controlled by using the `unauthenticatedUser` attribute of the `<safCredentials>` element. This value defaults to `WSGUEST`.
3. User ID authenticated by the Liberty server.
4. BBGZDFLT is the default value for the security profile prefix that is set by using the `profilePrefix` attribute of the `<safCredentials>` element, for example: `<safCredentials profilePrefix="BBGZDFLT"/>`.
5. EJBROLE profiles are required if the `<safAuthorization>` element is configured. The default pattern for the profile is controlled by the SAF role mapper element, which defaults to `<safRoleMapper profilePattern="%profilePrefix%.%resource%.%role%"/>`.

For more information, see Process types on z/OS.

Examples of identifying Liberty versions

Example: Identifying the angel Liberty version from the started task system log

If the Liberty angel process is running Liberty 18.0.0.2 or above, the started task system log contains a message that indicates the Liberty version:

```
CWWKB0079I THE ANGEL BUILD LEVEL IS 18.0.0.2 20180619-0654 2018.7.0.0 20180619-0654
```

Example: Identifying the version of a Liberty JVM server running in CICS from message DFHSJ1405

The version of a Liberty running in a CICS JVM server is available in the following message:

```
DFHSJ1405I 08/22/2018 17:04:39 IYK3ZDRI JVMSERVER EYUCMCIJ is running WebSphere Application Server
                          Version 18.0.0.2 Liberty - (18.0.0.2-cl180220180619-0403) process ID
67174497.
```

Example: Identifying both Liberty versions by running scripts

Suppose that the angel JCL specifies the following ROOT parameter:

```
// SET ROOT='/usr/lpp/zosmf/wlp'
```

To find out what the version of Liberty for the angel process is, run the following script:

```
/usr/lpp/zosmf/wlp/bin/productInfo version --verbose
```

For a Liberty JVM server running in CICS, run the following script:

```
/usr/lpp/cicsts56/wlp/bin/productInfo version --verbose
```

```
WebSphereApplicationServer.properties:
        com.ibm.websphere.productId=com.ibm.websphere.appserver
        com.ibm.websphere.productOwner=IBM
        com.ibm.websphere.productVersion=16.0.0.3
        com.ibm.websphere.productName=WebSphere Application Server
        com.ibm.websphere.productInstallType=Archive
        com.ibm.websphere.productEdition=zOS
        com.ibm.websphere.productLicenseType=IPLA

WebSphereApplicationServerZOS.properties:
        com.ibm.websphere.productId=com.ibm.websphere.appserver.zos
        com.ibm.websphere.productOwner=IBM CORP
        com.ibm.websphere.productVersion=16.0.0.3             <== Liberty Version
        com.ibm.websphere.productName=WAS FOR Z/OS
        com.ibm.websphere.productPID=5655-WAS
        com.ibm.websphere.productQualifier=WAS Z/OS
        com.ibm.websphere.productReplaces=com.ibm.websphere.appserver
        com.ibm.websphere.productEdition=
        com.ibm.websphere.gssp=true

zOSMF.properties:
        com.ibm.websphere.productId=com.ibm.zoszmf
        com.ibm.websphere.productOwner=IBM
        com.ibm.websphere.productVersion=2.2.0
        com.ibm.websphere.productName=z/OSMF
        com.ibm.websphere.productPID=5650-ZOS
        com.ibm.websphere.productQualifier=z/OSMF
        com.ibm.websphere.productReplaces=com.ibm.websphere.appserver.zos
        com.ibm.websphere.productEdition=N/A
```

Figure 31. Example output

Authenticating users in a Liberty JVM server

Although you can configure CICS security for all web applications that run in a Liberty JVM server, the web application will only authenticate users if it includes a security constraint. The security constraint is defined by an application developer in the deployment descriptor (web.xml) of the Dynamic Web Project or OSGi Application Project. The security constraint defines what is to be protected (URL) and by which roles.

A `<login-config>` element defines the way a user gains access to web container and the method used for authentication. The supported methods are either HTTP basic authentication, form based authentication or SSL client authentication. For further details on how to define application security for CICS see SSL security for Explorer connections in the CICS Explorer product documentation. Here is an example of those elements in web.xml:

```
<!-- Secure the application -->
<security-constraint>
    <display-name>com.ibm.cics.server.examples.wlp.tsq.web_SecurityConstraint</display-name>
        <web-resource-name>com.ibm.cics.server.examples.wlp.tsq.web</web-resource-name>
        <description>Protection area for com.ibm.cics.server.examples.wlp.tsq.web</description>
        <url-pattern>/*</url-pattern>
    </web-resource-collection>
    <auth-constraint>
        <description>Only SuperUser can access this application</description>
        <role-name>SuperUser</role-name>
    </auth-constraint>
    <user-data-constraint>
        <!-- Force the use of SSL -->
        <transport-guarantee>CONFIDENTIAL</transport-guarantee>
    </user-data-constraint>
</security-constraint>

<!-- Declare the roles referenced in this deployment descriptor -->
<security-role>
    <description>The SuperUser role</description>
    <role-name>SuperUser</role-name>
</security-role>

<!--Determine the authentication method -->
<login-config>
        <auth-method>BASIC</auth-method>
```

```
</login-config>
```

Note: If you use RequestDispatcher.forward() methods to forward requests from one servlet to another, the security check occurs only on the first servlet that is requested from the client.

Tasks that are authenticated in CICS using Liberty security can use the user ID derived from any of the Liberty application security mechanisms to authorize transaction and resource security checks in CICS. The CICS user ID is determined according to the following criteria:

1. Liberty application security authentication.

   Integration with the SAF user registry is part of the CICS Liberty security feature, unless distributed identity mapping is used. Any of the application security mechanisms supported by Liberty are supported in CICS, this includes HTTP basic authentication, form login, SSL client certificate authentication, identity assertion using a custom login module, JACC, JASPIC, or a Trust Association Interceptor (TAI). All SAF user IDs authenticated by Liberty must be granted read access to the Liberty JVM server APPL class profile. The name of this is determined by the profilePrefix setting in the safCredentials element in the Liberty server configuration file server.xml.

   ```
   <safCredentials profilePrefix="BBGZDFLT"/>
   ```

   The APPL class is also used by CICS terminal users to control access to specific CICS regions and your Liberty JVM server can use the same profile as the CICS APPLID depending upon your security requirements. If you do not specify this element, then the default profilePrefix of BBGZDFLT is used.

   You must define the APPLID and users must have access to the it. To configure and activate the BBGZDFLT profile in the APPL class:

   ```
   RDEFINE APPL BBGZDFLT UACC(NONE)
   SETROPTS CLASSACT(APPL)
   ```

   The users must be given read access to the BBGZDFLT profile in the APPL class in order to authenticate. To allow user AUSER to authenticate against the BBGZDFLT APPL class profile:

   ```
   PERMIT BBGZDFLT CLASS(APPL) ACCESS(READ) ID(AUSER)
   ```

   The Liberty SAF unauthenticated user id must be given read access to the APPL class profile. The SAF unauthenticated user id can be specified in the safCredentials element in the Liberty server configuration file server.xml.

   ```
   <safCredentials unauthenticatedUser="WSGUEST"/>
   ```

   If you do not specify the element, then the default unauthenticatedUser is WSGUEST. To allow the SAF unauthenticated user id WSGUEST read access to the BBGZDFLT profile in the APPL class:

   ```
   PERMIT BBGZDFLT CLASS(APPL) ACCESS(READ) ID(WSGUEST)
   ```

   The WLP z/OS System Security Access Domain (WZSSAD) refers to the permissions granted to the Liberty server. These permissions control which System Authorization Facility (SAF) application domains and resource profiles the server is permitted to query when authenticating and authorizing users. The CICS region user ID must be granted permission within the WZSSAD domain to make authentication calls. To grant permission to authenticate, the CICS region ID must be granted READ access to the BBG.SECPFX.<APPL> profile in the SERVER class:

   ```
   RDEFINE SERVER BBG.SECPFX.BBGZDFLT UACC(NONE)
   PERMIT BBG.SECPFX.BBGZDFLT CLASS(SERVER) ACCESS(READ) ID(cics_region_user)
   ```

   For more details refer to Accessing z/OS security resources using WZSSAD.
2. If an unauthenticated subject is supplied from Liberty, then the USERID defined in the URIMAP will be used.
3. If no USERID is defined in the URIMAP the request will run under the CICS default user ID.

Note:

Due to the way that security processing for Liberty transactions is deferred during CICS transaction attach processing, the user ID used in the CICS Monitoring Facility (CMF) records, the z/OS Workload Manager (WLM) classification, and the task association data and the UEPUSID global user exits field for the XAPADMGR exit, will be determined as follows; the user ID in the HTTP security header, or if there isn't one, the user ID taken from matching URIMAP. If neither exist, the CICS default user ID will be used.

Be aware that Liberty caches authenticated user IDs and, unlike CICS, does not check for an expired user ID within the cache period. You can configure the cache timeout by using the standard Liberty configuration process. Please see Configuring the authentication cache in Liberty.

Authorizing users to run applications in a Liberty JVM server

You can use Enterprise Java application security roles to authorize access to Enterprise Java applications. Additionally, in a Liberty JVM server you can further restrict access to transactions (run as part of the application) by using CICS transaction and resource security.

About this task

Your application is secured by providing an authorization constraint, the `<auth_constraint>` element, in the deployment descriptor (`web.xml`). If present, this ensures that access to your application is achieved only by a user that is a member of an authorized role. User or group membership to an Enterprise Java role is determined in one of two ways:

- Use an `<application-bnd>` element in the `<application>` element of your `server.xml` to describe the user/group to role mappings directly in XML.
- Use `<safAuthorization>` in your `server.xml` to allow users/groups role membership to be mapped by SAF (typically using EJBROLEs).

For more information, see Authorization using SAF role mapping.

Using CICS security allows you to re-use existing security procedures but requires that individual web applications are accessed from different URIMAPs. Using role-based security allows you to use existing standard Enterprise Java security definitions from another Enterprise Java application server. For more information, see “Authenticating users in a Liberty JVM server” on page 352.

If you want to use CICS transaction and resource authorization exclusively, or prefer to use finer-grained annotation-based role checking in code, you can defer the authorization decision to those components by using the special subject ALL_AUTHENTICATED_USERS role, as shown in the following example. If you deploy a Liberty application in a CICS bundle, CICS automatically configures this for you.

Note: Access checks are performed for the declarative security annotations and CICS transaction and resource security only after the configured constraints (`web.xml`) are verified

```
<application id="com.ibm.cics.server.examples.wlp.tsq.app"
    name="com.ibm.cics.server.examples.wlp.tsq.app" type="eba"
    location="${server.output.dir}/installedApps/com.ibm.cics.server.examples.wlp.tsq.app.eba">
    <application-bnd>
        <security-role name="cicsAllAuthenticated">
            <special-subject type="ALL_AUTHENTICATED_USERS"/>
        </security-role>
    </application-bnd>
</application>
```

Using this special subject, and giving the cicsAllAuthenticated role access to all URLs in your web applications deployment descriptor (`web.xml`), allows access to the web application using any authenticated user ID and authorization to the transaction must be controlled using CICS transaction security. If you deploy your application directly to the dropins directory, it is not configured to use CICS security as dropins does not support security.

If you are using safAuthorization then the `<application-bnd>` no longer acts as the source of user ID to role mapping. Instead, EJBROLEs in SAF determine which SAF users are in which roles (EJBROLEs). With safAuthorization the `<application-bnd>` is ignored. To achieve the same effect and allow all

authenticated users to be authorized to run your application, the <auth-constraint> in web.xml must use the special role **, for example:

```
<auth-constraint>
        <description>special role for all authenticated users</description>
        <role-name>**</role-name>
</auth-constraint>
```

- The special role name ** is a shorthand for any authenticated user independent of role.
- The special role name * is a shorthand for all role names defined in the deployment descriptor.

When the special role name ** appears in an authorization constraint, it indicates that any authenticated user, independent of role, is authorized to perform the constrained requests. Special roles do not need an additional <security-role> declaration in web.xml.

To use CICS transaction or resource security you should follow the following steps:

Procedure

1. Define a URIMAP of type JVMSERVER for each web application. Typically, you might specify a URIMAP to match the generic context root (URI) of a web application to scope the transaction ID to the set of servlets that make up the application. Or you may choose to run each individual servlet under a different transaction with a more precise URI.
2. Authorize all users of the web application to use the transaction specified in the URIMAP using CICS transaction or resource security profiles.

Authorizing applications by using OAuth 2.0

OAuth 2.0 is an open standard for delegated authorization. The OAuth authorization framework enables a user to grant a third-party application access to information that is stored with another HTTP service without sharing their access permissions or the full extent of their data.

WebSphere Liberty supports OAuth 2.0, and can be used as an OAuth service provider endpoint and an OAuth protected resource enforcement endpoint. Liberty supports persistent OAuth 2.0 services. See Configuring persistent OAuth 2.0 services. Clients can be defined locally with the localStore and client elements. The following procedure uses local clients to enable OAuth 2.0 authorization.

Before you begin

SAF security is a common use-case in CICS, and this procedure uses SAF in the examples.

Ensure that the CICS region is configured to use SAF security and is defined with SEC=YES as a system initialization parameter.

Optionally, you can grant an administrator user access to the SAF EJBROLE BBGZDFLT.com.ibm.ws.security.oauth20.clientManager. The security role clientManager controls access to the management interfaces, allowing local clients to be queried, and persistent local clients to be created. The administrator user controls OAuth 2.0 local clients.

Configure the Liberty angel process to provide authentication and authorization services to the Liberty JVM server. See The Liberty server angel process.

For more information about OAuth, see oauth-2.0.

About this task

The following procedure covers how to:

- Create an OAuth 2.0 service provided in a Liberty JVM server.
- Create a locally configured client.
- Use this local client to grant an OAuth 2.0 token to a relying party application, also known as a third-party web application.
- Use this token to access protected resources in an application.

Restriction: Db2 JDBC type 2 connectivity is not supported for persistent OAuth 2.0 services.

Procedure

1. Configure an OAuth 2.0 service provider.
   a) Add the `oauth-2.0` and the `cicsts:security-1.0` features to the `featureManager` element in `server.xml`.

```
<featureManager>
    ...
    <feature>oauth-2.0</feature>
    <feature>cicsts:security-1.0</feature>
</featureManager>
...
```

   b) Configure an OAuth 2.0 provider in `server.xml`.

```
<oauthProvider id="myProvider">
</oauthProvider>
```

2. Configure a local client for the relying party application. Local clients define the details of the relying party application, including the name, secret password, and redirect URI of the application.
   a) Define a meaningful local client name and create a secret password that is used by the server for authorization. The local client application listens on a URI, and the server supplies authorization codes.
   b) Configure an OAuth 2.0 local client in the `oauthProvider` element of `server.xml`, supplying the local client ID, secret password, and the redirect URI.

```
<oauthProvider id="myProvider">
    <localStore>
        <client id="myClient" redirect="https://client.example.ibm.com/webApp/redirect"
secret="mySecret" />
    </localStore>
</oauthProvider>
```

   Important:

   Although it is not shown in this example, it is important to encode passwords and limit access to `server.xml` configuration. Passwords can be encoded by using the Liberty `securityUtility`, found in USS_HOME/`wlp/bin/securityUtility`. For more information, see securityUtility command.

   Note: More than one local client can be configured in the `localStore` element.

3. When the relying party application requires access to protected resources on the server, the user must authorize access to these resources first.
   a) The relying party application requires the user to authenticate with the server, and select the type of access for the relying party application by linking or redirecting the user to the authorization endpoint:

```
https://hostname:port/oauth2/endpoint/provider_name/authorize
```

   or

```
https://hostname:port/oauth2/declarativeEndpoint/provider_name/authorize
```

   Additional parameters are required in the query parameters of the URL. For the local client that was configured in step 2, the following GET request is required (all one line):

```
https://zos.example.ibm.com/oauth2/endpoint/myProvider/authorize?response_type=code
   &client_id=myClient&client_secret=mySecret&redirect_uri=https://client.example.ibm.com/webApp/redirect
```

After the user selects the access for the relying party application, they are redirected back to the relying party application using the redirect URI:

```
https://client.example.ibm.com/webApp/redirect?code=access_code
```

The relying party application must store this access code to request an OAuth token.

Note: For local clients, the user must exist in a user register in the Liberty JVM server. For more information about authenticating users in Liberty JVM servers, see Authenticating users in a Liberty JVM server.

b) The relying party application requests an OAuth 2.0 token by sending a POST request to the server:

```
https://hostname:port/oauth2/endpoint/provider_name/token
```

The relying party application sends the authorization code that is received from the authorization endpoint, the local client ID, and the secret password in the POST data (`grant_type` is all one line):

```
POST https://zos.example.ibm.com/oauth2/endpoint/myProvider/token HTTP/1.1
Content-Type: application/www-form-urlencoded

grant_type=authorization_code&code=code&client_id=myClient
   &client_secret=mySecret&redirect_url=https://client.example.ibm.com/webApp/redirect
```

This returns a JSON document that contains the token.

4. Use the token to access protected resources.

a) Add the token to the `Authorization` header on the HTTP request.

```
Authorization: Bearer <token>
```

Results

Users are able to authorize third-party applications to access their protected resources in a Liberty JVM server through OAuth 2.0 authorizations flows. The Liberty JVM server can configure the provider of these tokens and create locally configured clients.

Several methods to grant tokens are available. For more information, see OAuth 2.0 service invocation.

Configuring persistent OAuth 2.0 services

WebSphere Liberty supports persisting OAuth 2.0 local clients and tokens to a database. With persistent OAuth 2.0, an authorized local client can continue to access OAuth 2.0 services after a restart.

Before you begin

SAF security is a common use-case in CICS, and this procedure uses SAF in the examples.

- Gain the necessary access to create tables and read/write to these tables in a database and configure it in the Liberty `server.xml`.
- Grant access to the SAF EJBROLE `BBGZDFLT.com.ibm.ws.security.oauth20.clientManager` to an administrator user to control OAuth 2.0 local clients.
- Create an OAuth 2.0 provider in the Liberty `server.xml`. For more information, see Authorization using OAuth 2.0.

About this task

The following steps create a persistent OAuth 2.0 local client. This local client is used to grant OAuth 2.0 tokens.

Restriction: Db2 JDBC type 2 connectivity is not supported for persistent OAuth 2.0 services.

Procedure

1. Create the necessary tables using IBM Db2 for persistent OAuth services as a guide.
2. Create a persistent local client by sending a POST request to the URL:

```
https://hostname:port/oauth2/endpoint/provider_name/registration
```

Use the JSON document which is described in the first table in Configuring an OpenID Connect Provider to accept client registration requests; for example:

```
{
    "client_id": "client_id",
    "client_secret": "client_secret",
    "grant_types": [ "authorization_code", "refresh_token" ],
    "redirect_uris": [ "https://client.example.ibm.com/webApp/redirect" ]
}
```

Results

A persistent OAuth 2.0 local client is created. When this local client is used to produce tokens, the tokens are persisted to the database. If the server restarts, the persistent local client and tokens remain valid.

Authorization using SAF role mapping

Mapping Java EE roles to users and groups can be achieved in different ways. In distributed systems, a basic registry or LDAP registry would typically be used in conjunction with an application specific <application-bnd> element, to map users from those registries into *roles*. The deployment descriptor of the application determines which roles can access which parts of the application.

About this task

On z/OS, there is an additional registry type, the System Authorization Facility (SAF) registry. A Liberty JVM server implicitly uses this type for authentication when the cicsts:security-1.0 feature is installed unless configured to use LDAP. You can choose to make use of SAF authorization. When using SAF authorization, user to role mappings are used to map roles to EJBROLE resource profiles using the SAF role mapper. The server queries SAF to determine if the user has the required READ access to the EJBROLE resource profile.

In a Liberty JVM server, if you want to use Java EE roles without SAF authorization, you cannot use CICS bundles to install your applications. This is because a CICS bundle installed application automatically creates an <application-bnd> element and uses the ALL_AUTHENTICATED_USERS special-subject, which prevents you from defining the element yourself. Instead, you must create an <application> element in server.xml directly and configure the <application-bnd> with the roles and users you require.

If, however, you choose to use Java EE roles and SAF authorization, you can continue to use CICS bundles to lifecycle your web applications. The <application-bnd> is ignored by Liberty in favor of using the role mappings determined by the SAF registry. Role mappings are determined by virtue of a user belonging to an EJB role.

Tip: When SAF authorization is enabled, EJB roles in RACF are used for role mapping instead of the roles in server.xml. Therefore, special subjects such as ALL_AUTHENTICATED_USERS and EVERYONE, or users can not be defined in server.xml in this case.

Tip: It is advisable to create or update your EJB roles before starting the CICS region. Liberty issues a RACROUTE REQUEST=LIST with GLOBAL=NO in order to support a minimum version of z/OS. The address space will not see updates until it is restarted (or started).

Procedure

1. Add the <safAuthorization id="saf"/> element to your server.xml. If you are using the cicsts:distributedIdentity-1.0 feature, this is defined for you.

2. Optional: You can add `racRouteLog="ASIS"` to the element in the previous step.

   This allows you to see the RACF EJBROLE logging from Liberty.
3. Create the EJB roles in RACF, with reference to the prefix scheme described.
4. Add users to those EJB roles.

   By default, if SAF authorization is used, the application uses the pattern `<profile_prefix>.<resource>.<role>` to determine if a user is in a role. The `profile_prefix` defaults to BBGZDFLT but can be modified using the `<safCredentials>` element. For example, you can set it to the `APPL_ID` of a region. If you want multiple regions to share identical security configuration, you can set `<profile_prefix>` to the same value for those regions. For more information, see Accessing z/OS security resources using WZSSAD.

   The role mapping preferences can be modified using the `<safRoleMapper>` element in the `server.xml`, for example:

   ```
   <safRoleMapper profilePattern="myprofile.%resource%.%role%" toUpperCase="true"/>
   ```

   Users can then be authorized to a particular EJB role using the following RACF commands, where WEBUSER is the authenticated user ID.

   ```
   RDEFINE EJBROLE BBGZDFLT.MYAPP.ROLE UACC(NONE)
   PERMIT BBGZDFLT.MYAPP.ROLE CLASS(EJBROLE) ACCESS(READ) ID(WEBUSER)
   ```
5. Optional: If you are deploying the CICS servlet examples and want to use the Java EE role security with SAF authorization, create a SAF EJBROLE for each servlet that you have deployed. For example, if you use the default APPL class of BBGZDFLT, define the following EJBROLE security definitions using RACF commands:

```
RDEFINE EJBROLE BBGZDFLT.com.ibm.cics.server.examples.wlp.hello.war.cicsAllAuthenticated UACC(NONE)
RDEFINE EJBROLE BBGZDFLT.com.ibm.cics.server.examples.wlp.tsq.app.cicsAllAuthenticated UACC(NONE)
RDEFINE EJBROLE BBGZDFLT.com.ibm.cics.server.examples.wlp.jdbc.app.cicsAllAuthenticated UACC(NONE)
SETROPTS RACLIST(EJBROLE) REFRESH
```

   Give read access to the defined roles for each web user ID that requires authorization:

```
PERMIT BBGZDFLT.com.ibm.cics.server.examples.wlp.hello.war.cicsAllAuthenticated
       CLASS(EJBROLE) ID(user) ACCESS(READ)
PERMIT BBGZDFLT.com.ibm.cics.server.examples.wlp.tsq.app.cicsAllAuthenticated
       CLASS(EJBROLE) ID(user) ACCESS(READ)
PERMIT BBGZDFLT.com.ibm.cics.server.examples.wlp.jdbc.app.cicsAllAuthenticated
       CLASS(EJBROLE) ID(user) ACCESS(READ)
SETROPTS RACLIST(EJBROLE) REFRESH
```

Results

You can authorize access to web applications using CICS Security, Java EE role security, or both by defining the roles and the users in the roles.

Configuring security for a Liberty JVM server with the Enterprise Java security API

Java EE 8 introduces a portable, flexible, and standardized security model with the Java EE security API 1.0. A Liberty JVM server can be configured to respect the new security configuration through the inclusion of the Liberty appSecurity-3.0 feature.

The Java EE security API 1.0 specification covers three principles:

1. Authentication mechanism: provided by the HttpAuthenticationMechanism interface for the servlet container
2. Identity store: an attempt to standardize the JAAS LoginModule
3. Security context: an access point for programmatic security

Authentication mechanism

An authentication mechanism is a way that is used to obtain a username and password from the user to be processed later by the Java Security API. There are two standard options for authentication, both take advantage of the annotations that are introduced by the Java EE security 1.0 API.

HTTP basic authentication

Basic authentication displays the browser's native login dialog before the user can access the protected resource.

```
@BasicAuthenticationMechanismDefinition(realmName="user-realm")
@WebServlet("/home") @DeclareRoles({"user"})
@ServletSecurity(@HttpConstraint(rolesAllowed = "user"))
public class HomeServlet extends HttpServlet {
    ...
}
```

Form-based authentication

You can use form-based authentication to replace the browser’s built-in dialog with your own custom HTML form. You can create an application config class with annotations as follows:

```
@FormAuthenticationMechanismDefinition(
    loginToContinue = @LoginToContinue(
        loginPage = "/login",
        errorPage = "/error"
    )
)
@ApplicationScoped
public class ApplicationConfig {
    ...
}
```

Identity store

A component acts as a DAO (Data Access Object) for accessing user information, including their usernames, passwords, and associated roles. A number of identity store types are introduced by the Java EE security API 1.0, including:

Database identity store

A database identity store is used to retrieve user information from a relation database.

```
@DatabaseIdentityStoreDefinition(
dataSourceLookup = "jdbc/sec",
    callerQuery = "#{'select password from USR where USERNAME = ?'}",
    groupsQuery = "#{'select ugroup from USR where USERNAME = ?'}",
    hashAlgorithm = Pbkdf2PasswordHash.class,
    priorityExpression = "#{100}",
    hashAlgorithmParameters = {
        "Pbkdf2PasswordHash.Iterations=3072",
        "Pbkdf2PasswordHash.Algorithm=PBKDF2WithHmacSHA512",
        "Pbkdf2PasswordHash.SaltSizeBytes=64"
    }
)
```

Lightweight Directory Access Protocol (LDAP) identity store

LDAP is a common way of organizing a user's access to different systems across a single organization. LDAP realizes the idea of Single-Sign On, where a user has a single username and password, and then uses it across all different systems that are used to perform the everyday business of a specific organization.

```
@WebServlet("/home")
@ServletSecurity(@HttpConstraint(rolesAllowed = "user"))
@LdapIdentityStoreDefinition(
    url = "ldap://localhost:33389/",
    callerBaseDn = "ou=user,dc=jsr375,dc=net",
    groupSearchBase = "ou=group,dc=jsr375,dc=net"
)
public class HomeServlet extends HttpServlet{
```

```
    ...
}
```

URL: The URL of the LDAP server to use for authentication.

callerBaseDn: Base distinguished name for callers in the LDAP store.

groupSearchBase: Search base for looking up groups.

Custom identity store

In addition to the built-in identity stores found in Java EE security API 1.0, a user can implement their own identity store and control exactly where to obtain user information. This can be achieved by creating a custom identity store class, then creating an HTTP authentication mechanism associated with this custom identity store.

Security context

The security context object is used to programmatically check a user's authority to access a specific resource. This is useful when you need to perform custom behavior. In this example, the user is forwarded to another page only if they have access to it:

```
@WebServlet("/home")
public class HomeServlet extends HttpServlet {
    @Inject
    private SecurityContext securityContext;
    @Override
    protected void doGet(HttpServletRequest req, HttpServletResponse resp)
        throws ServletException, IOException {
        if (securityContext.hasAccessToWebResource("/anotherServlet", "GET")) {
            req.getRequestDispatcher("/anotherServlet").forward(req, res);
        } else {
            req.getRequestDispatcher("/logout").forward(req, res);
        }
    }
}
```

For more information about the Java EE 8 security API, see Java EE Security API in the Liberty documentation.

Authenticating by using a database identity store

You can use the @DatabaseIdentityStoreDefinition interface to retrieve user credentials from a database for authentication.

About this task

Follow these steps to authenticate by using a database identity store.

Procedure

1. Add the `appSecurity-3.0` feature to `server.xml` before you start the server.
2. Ensure that CDI annotation file scanning is enabled. CICS disables it by default in `server.xml`.

   You can ensure CDI annotation file scanning is enabled by checking the following line is not present in `server.xml`: `<cdi12 enableImplicitBeanArchives="false"/>`.
3. Create a table in the database and set up `server.xml`.
   For example, to create a Db2 table using SQL:

```
CREATE TABLE PXX.USR (
    USERNAME      VARCHAR ( 256 ) NOT NULL,
    PASSWORD      VARCHAR ( 256 ) NOT NULL,
    UGROUP        VARCHAR ( 256 ) NOT NULL
) IN SECU.TSSE;
CREATE UNIQUE INDEX INDXUSRS ON PXX.USR (USERNAME);
```

The password in the database must be encrypted. An example of inserting an encrypted password into a database can be found here: Database Setup

a) Add the jdbc-4.2 feature in `server.xml`:

```
<feature>jdbc-4.2</feature>
```

b) Set `jndiName` in `server.xml`, for example:

```
<dataSource id="DefaultDataSource" jndiName="jdbc/sec">
    <jdbcDriver libraryRef=<xxx>"/>
    ...
</dataSource>
```

4. Determine whether to use SAF for the CICS task userid.

a) If you do not want to push the database identity onto the CICS task, you can remove the default safRegistry setting in `server.xml`. This makes the CICS task run under the default CICS userid.

b) If you want CICS tasks to run under specific SAF users mapped from your database identity store, you need to take the following steps:

i) Configure SAF in `server.xml` by setting the following SAF elements.

```
<safCredentials mapDistributedIdentities="true" profilePrefix=lt;xxx>"/;
<safAuthorization id="saf"/>
<safRoleMapperprofilePattern=“<xxx>.%resource%.%role%" toUpperCase="false;
```

ii) Issue the RACMAP command. The general RACMAP command of mapping a distributed userid to a SAF userid is in the format of:

```
RACMAP ID(userid)
MAP
WITHLABEL('label-name')
USERDIDFILTER(NAME('distributed-identity-user-name'))
REGISTRY(NAME('distributed-identity-registry-name'))
```

Use `defaultRealm` in `REGISTRY(NAME(<nnn>))`, and use `<username_in_DBIS>` in `USERDIDFILTER(NAME(<nnn>))`, for example:

```
RACMAP ID(JATM12) MAP WITHLABEL('authorisedUser:JATM12')
USERDIDFILTER(NAME('authorisedUser')) REGISTRY(NAME('defaultRealm'))
```

Note: If you deploy the application in a CICS bundle, the security role `"cicsAllAuthenticated"` is automatically set in the `installedApps.xml` as follows:

```
<application ...>
    <application-bnd>
        <security-role name="cicsAllAuthenticated">
            <special-subject type="ALL_AUTHENTICATED_USERS"/>
        </security-role>
    </application-bnd>
</application>
```

The security role `"cicsAllAuthenticated"` takes precedence over the group name that is stored in the database identity store and an HTTP 403 error occurs. There are two options you take:

i) Deploy your database identity store application with a direct `<application>` element in `server.xml`.

ii) Deploy within a CICS bundle, but use safAuthorization to bypass the CICS-generated `<application-bnd>` which overrides the group information stored in the Custom Identity Store.

Results

You have successfully configured the database identity store.

Authenticating by using a custom identity store

You can use a custom identity store to implement your own identity store and control exactly where to obtain user information.

About this task

Follow these steps to authenticate by using a custom identity store.

Procedure

1. Add the `appSecurity-3.0` feature to `server.xml` before you start the server.
2. Ensure that CDI annotation file scanning is enabled. CICS disables it by default in `server.xml`.

   You can ensure that CDI annotation file scanning is enabled by checking the following line is not present in `server.xml`: `<cdi12 enableImplicitBeanArchives="false"/>`.
3. Create Java classes to process the custom identity store logic and build them into a WAR file.
   a) Create a custom identity store object, by creating a class that implements the IdentityStore interface, as shown in the following example:

```
@ApplicationScoped
public class MyIdentityStore implements IdentityStore {
    public CredentialValidationResult validate(UsernamePasswordCredential userCredential)
{
        if (userCredential.compareTo("authorisedUser", "tomtom")) {
            return new CredentialValidationResult("authorisedUser",
                new HashSet<String>(asList("user")));
        }
        return INVALID_RESULT;
    }
}
```

   b) Create an HTTP authentication mechanism associated with this identity store, which is used with the identity store class that is created in the previous step:

```
@ApplicationScoped
public class MyAuthMechanism implements HttpAuthenticationMechanism {

    @Inject
    private IdentityStoreHandler idStoreHandler;

    public AuthenticationStatus validateRequest(HttpServletRequest req,
        HttpServletResponse res, HttpMessageContext context) {
        CredentialValidationResult result = idStoreHandler.validate(
            new UsernamePasswordCredential(
                req.getParameter("name"),
                req.getParameter("password")));
        if (result.getStatus() == CredentialValidationResult.Status.VALID) {
            return context.notifyContainerAboutLogin(result);
        } else {
            return context.responseUnauthorized();
        }
    }
}
```

   c) Create a servlet.

```
@WebServlet("/home")
@ServletSecurity(@HttpConstraint(rolesAllowed = "user"))
public class Servlet extends HttpServlet {...}
```

4. Determine whether to use SAF for the CICS task userid.
   a) If you do not want to push the custom identity onto the CICS task, you can remove the default safRegistry setting in `server.xml`. This makes the CICS task run under the default CICS userid.
   b) If you want CICS tasks to run under specific SAF users mapped from your custom identity store, you need to take the following steps:

i) Configure SAF in `server.xml` by setting the following SAF elements.

```
<safCredentials mapDistributedIdentities="true" profilePrefix="<xxx>"/>
<safAuthorization id="saf"/>
<safRoleMapperprofilePattern="<xxx>.%resource%.%role%" toUpperCase="false"/>
```

ii) Issue the RACMAP command. The general RACMAP command of mapping a distributed userid to a SAF userid is in the format of:

```
RACMAP ID(userid)
MAP
WITHLABEL('label-name')
USERDIDFILTER(NAME('distributed-identity-user-name'))
REGISTRY(NAME('distributed-identity-registry-name'))
```

Use "`defaultRealm`" in `REGISTRY(NAME('<nnn>'))`, and use "`<username_in_CIS>`" in `USERDIDFILTER(NAME('<nnn>'))`, for example:

```
RACMAP ID(JATM12) MAP WITHLABEL('authorisedUser:JATM12')
USERDIDFILTER(NAME('authorisedUser')) REGISTRY(NAME('defaultRealm'))
```

Note: If you deploy the application within a CICS bundle, the security role "`cicsAllAuthenticated`" is automatically set in `installedApps.xml` as follows:

```
<application ...>
    <application-bnd>
        <security-role name="cicsAllAuthenticated">
            <special-subject type="ALL_AUTHENTICATED_USERS"/>
        </security-role>
    </application-bnd>
</application>
```

It takes precedence over the group name that is stored in the custom identity store and an HTTP 403 error occurs. There are two options you can take:

i) Deploy your custom identity store application with a direct `<application>` element in `server.xml`.

ii) Deploy within a CICS bundle, but use safAuthorization to bypass the CICS-generated `<application-bnd>` which overrides the group information stored in the custom identity store.

Results

You have successfully configured the custom identity store.

Configuring security for a Liberty JVM server by using an LDAP registry

Liberty uses a user registry to authenticate a user and retrieve information about users and groups to perform security-related operations, including authentication and authorization. Default CICS Liberty security uses the SAF registry. However, many transactions that run on CICS are initiated by users who authenticate their identities on distributed application servers, so CICS also supports the use of a Lightweight Directory Access Protocol (LDAP) registry in Liberty. To use LDAP, it is necessary to manually configure the `server.xml`.

Before you begin

- Ensure that the CICS region is configured to use SAF security and is defined with SEC=YES as a system initialization parameter.
- Authorize application developers and system administrators to create, view, update, and remove JVMSERVER and BUNDLE resources to deploy web applications into a Liberty JVM server. The JVMSERVER resource controls the availability of the JVM server, and the BUNDLE resource is a unit of deployment for the Java applications and controls the availability of the applications.

About this task

This task explains how to configure LDAP security for a Liberty JVM server, and integrate Liberty security with CICS security. Distributed identity mapping can be used to associate a SAF user ID with a distributed identity. You can use the CICS distributed identity mapping feature to set up distributed identity mapping. A user can then log on to a CICS web application with their distributed identity, as authenticated by an LDAP server. Filters that are defined in the z/OS security product (RACMAP) determine the mapping of this identity to a SAF user ID. This SAF user ID can then be used to authorize access to web applications through JEE application role security, providing integration with CICS transaction and resource security. You can map a SAF user ID to one or more distributed identities.

The default transaction ID for running any web request is CJSA. You can configure CICS to run web requests under a different transaction ID by using a URIMAP of type JVMSERVER. You can specify a URIMAP to match the generic context root (URI) of a web application to scope the transaction ID to the set of servlets that make up the application. Or you can choose to run each individual servlet under a different transaction with a more precise URI.

There are three scenarios for this task:

- Scenario 1 – Distributed identity mapping with SAF authorization
- Scenario 2 – Distributed identity mapping without SAF authorization
- Scenario 3 – LDAP for authentication and authorization

Procedure

1. **Distributed identity mapping with SAF authorization**

   You can use the CICS distributed identity mapping feature, `cicsts:distributedIdentity-1.0` to enable LDAP distributed identities to be mapped to SAF user IDs. When used with the CICS security feature `cicsts:security-1.0`, Liberty LDAP security is used for authentication and JEE application role security from EJB role mappings are respected for authorization. CICS transactions run under the mapped SAF user ID providing integration with CICS transaction and resource security.

   a. Configure the WebSphere Liberty angel process to provide authentication and authorization services to the Liberty JVM server, for more information see The Liberty server angel process.

   b. Add the `cicsts:security-1.0` and the `cicsts:distributedIdentity-1.0` feature to the featureManager list in the `server.xml`.

   ```
   <featureManager>
      ...
      <feature>cicsts:security-1.0</feature>
      <feature>cicsts:distributedIdentity-1.0</feature>
   </featureManager>
   ...
   ```

   c. Configure Liberty to use LDAP authentication by defining the LDAP server in the `server.xml`, for example:

   ```
   <ldapRegistry id="ldap"
       host="host.domain.com" port="389"
       ldapType="IBM Tivoli Directory Server"
       baseDN="ou=users,dc=domain,dc=com"
       ignoreCase="true">
   </ldapRegistry>
   ```

   Full details on configuring LDAP user registries with Liberty are available in Configuring LDAP user registries in Liberty.

   d. Remove the safRegistry element, if present. Save the changes to the `server.xml`.

   e. Make the necessary RACF definitions, including setting up the RACMAPs to map distributed identities to SAF user IDs as which are described in Configuring LDAP user registries in Liberty and providing access for these user IDs to the appropriate EJBROLEs as described in “Authorization using SAF role mapping” on page 358. CICS configures SAF authorization and the mapDistributedIdentities attributes in the safCredentials configuration element for you.

When the `cicsts:distributedIdentity-1.0` feature is used with the `cicsts:security-1.0` feature, Liberty LDAP security is used for authentication, and JEE application role security from EJB role mappings are respected for authorization. CICS transactions run under the RACMAP mapped user ID providing integration with CICS transaction and resource security.

What to do next

Back to top

2. **Distributed identity mapping without SAF authorization**

   It is possible to allow CICS transactions to run under a RACMAP mapped user ID while respecting the roles configured in the application's <application-bnd> element. This might be useful when migrating work from distributed Liberty to CICS Liberty. Be aware that if CICS bundles are used, a user-defined <application-bnd> is overwritten by the CICS-generated <application-bnd>. SAF authorization using role mapping is preferred, for more information see "Authorization using SAF role mapping" on page 358 for more details.

   a. Configure the WebSphere Liberty angel process to provide authentication and authorization services to the Liberty JVM server, for more information, see The Liberty server angel process.

   b. Add the `cicsts:security-1.0` and the `ldapRegistry-3.0` feature to the featureManager list in the `server.xml`.

   ```
   <featureManager>
      ...
      <feature>cicsts:security-1.0</feature>
      <feature>ldapRegistry-3.0</feature>
   </featureManager>
   ...
   ```

   c. Configure Liberty to use LDAP authentication by defining the LDAP server in the `server.xml`, for example:

   ```
   <ldapRegistry id="ldap"
       host="host.domain.com" port="389"
       ldapType="IBM Tivoli Directory Server"
       baseDN="ou=users,dc=domain,dc=com"
       ignoreCase="true">
   </ldapRegistry>
   ```

   Full details on configuring LDAP user registries with the Liberty are available in Configuring LDAP user registries in Liberty.

   d. Configure Liberty to use distributed identity filters to map the distributed identities to SAF user IDs by setting the `mapDistributedIdentities` attribute in the `safCredentials` configuration element to `true` in the `server.xml`.

   e. Remove the safRegistry element, if present. Save the changes to the `server.xml`.

   f. Make the necessary RACF definitions, including setting up the RACMAPs to map distributed identities to SAF user IDs as which are described in Configuring LDAP user registries in Liberty.

   g. If JEE application role security from EJB roles is required for authorization then refer to the topic "Authorization using SAF role mapping" on page 358.

   Applications use Liberty LDAP security for authentication, and JEE application role security in an <application-bnd> element are respected for authorization of the distributed identity. In CICS, transactions run under the RACMAP mapped user ID, providing integration with CICS transaction and resource security.

   What to do next

   Back to top

3. **LDAP for authentication and authorization**

   LDAP security can be used in a CICS Liberty JVM server for both authentication and authorization using JEE application role security. URIMAP definitions can then be used to set the user ID under which transactions run. The `mapDistributedIdentities` attribute is not set in this scenario.

This scenario might be useful if migrating a distributed application into a CICS Liberty JVM server, without requiring any significant security resource changes.

a. Add the `cicsts:security-1.0` and the `ldapRegistry-3.0` feature to the featureManager list in the `server.xml`.

```
<featureManager>
   ...
   <feature>cicsts:security-1.0</feature>
   <feature>ldapRegistry-3.0</feature>
</featureManager>
...
```

b. Configure Liberty to use LDAP authentication by defining the LDAP server in the `server.xml`, for example:

```
<ldapRegistry id="ldap"
    host="host.domain.com" port="389"
    ldapType="IBM Tivoli Directory Server"
    baseDN="ou=users,dc=domain,dc=com"
    ignoreCase="true">
</ldapRegistry>
```

Full details on configuring LDAP user registries with Liberty are available in Configuring LDAP user registries in Liberty.

c. Remove the safRegistry element, if present. Save the changes to the `server.xml`.

d. If JEE application role security from EJB roles is required for authorization then refer to the topic “Authorization using SAF role mapping” on page 358.

Applications use Liberty LDAP security for authentication, and JEE application role security in an `<application-bnd>`element are respected for authorization. In CICS transactions run under the URIMAP or CICS DFLTUSER user ID as appropriate.

What to do next

Back to top

What to do next

This applies to all three scenarios:

- Modify the Liberty authentication cache.
- Set up URIMAP definitions to map web application URIs to transaction IDs.

This applies to scenarios 1 and 2:

- Set up CICS transaction security definitions to authorize access to URIs based on the mapped user ID.

Back to top

Configuring security for remote JCICSX API development

The JCICSX server is a remote Liberty JVM server in a CICS region. With the JCICSX API, it allows developers to run Java applications on their local workstation as if they were run in CICS, without deploying the applications to CICS. When remote connection is established from a JCICSX development client in the developer's local JVM to a JCICSX server, the remote server can authenticate users and authorize them with access based on their identities to ensure security. It also prevents users from interfering with remote tasks started by other users.

Table of contents

“What authentication and authorization options are available?” on page 368
“What options to choose?” on page 369
“Typical scenarios and procedures” on page 370

What authentication and authorization options are available?

The JCICSX server *authenticates* users to verify their identity. When planning how to authenticate JCICSX users, you have a few options to consider. First, you need to decide which user registry is used to store the user identity information. This topic covers the configuration of two registry options: using user identities from SAF (`safRegistry`) and embedding user identities directly in your `server.xml` (`basicRegistry`).

After configuring where the server is to find user identities, those users can be *authorized* to use the JCICSX server application. By default, the JCICSX server allows all users who are able to be authenticated with the server to access its services. However, this is customizable. The JCICSX server defines the Enterprise Java role JCICSXUSER, which can be used to customize access. This is achieved by mapping a role, which provides access to the application, to a group of users. Also, this role mapping can either be recorded in SAF (`safAuthorization`) or embedded directly in your `server.xml`.

Note: You must have a SAF registry to use SAF authorization.

Therefore, as shown in Figure 32 on page 369, the following authentication and authorization options are available for your remote JCICSX server:

- Using a basic user registry for authentication and `server.xml` for role mapping.
- Using a SAF registry for authentication and `server.xml` for role mapping.
- Using a SAF registry for authentication and SAF authorization for role mapping.

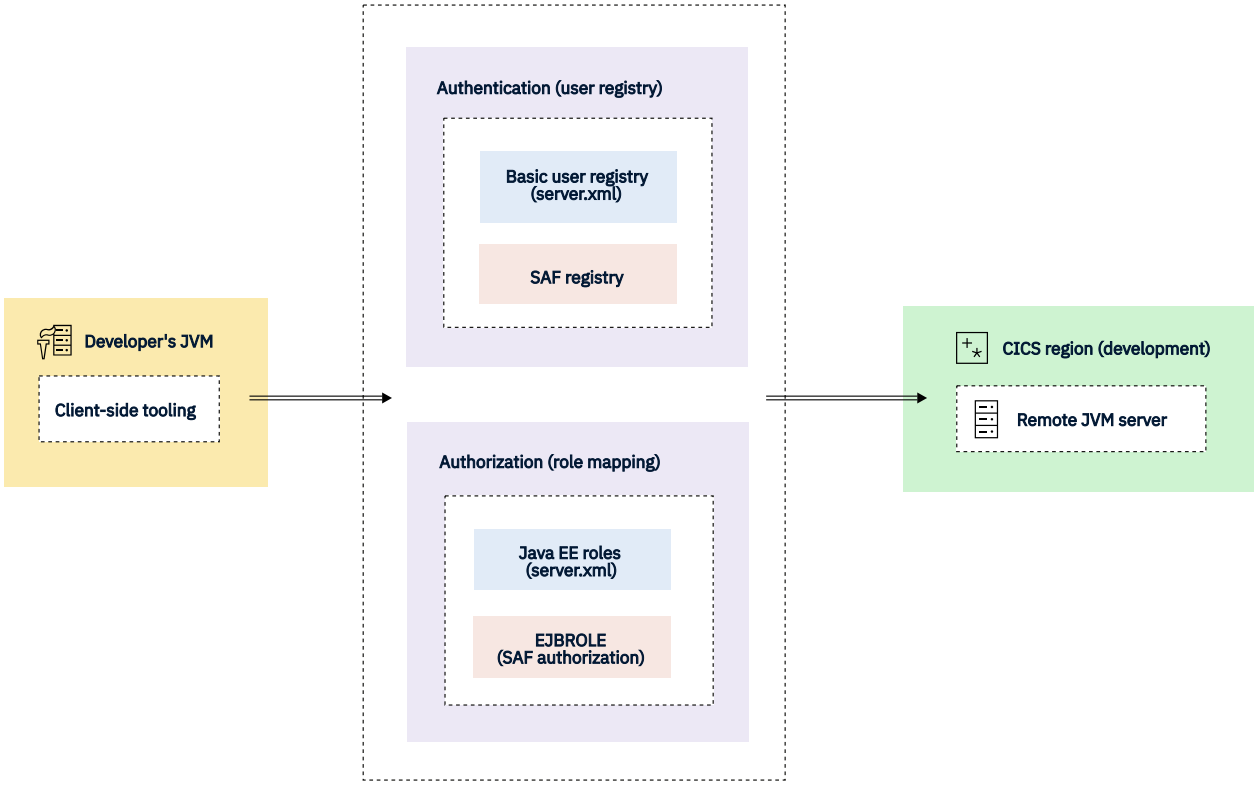


Figure 32. Authentication and authorization of remote JCICSX server

What options to choose?

From a logistical standpoint, it's simple to configure authentication and authorization directly in your `server.xml`. Therefore, it can be a convenient option if you are setting up the JCICSX server in a private development region. However, it has limitations at larger scales because the `server.xml` configuration is difficult to share. While it's more complicated to set up SAF for authentication and authorization, which involves the creation of EJB roles (EJBROLEs) in RACF, you can take advantage of existing information in your SAF database if you already have one. For example, you can authorize existing groups that are defined in SAF to use JCICSX, without having to specify them again. You can also share that information across multiple instances of JCICSX server running in different CICS regions, without having to configure each region independently.

Note: Any security configuration that you specify in the JCICSX server's `server.xml` must co-exist with the security requirements of other applications you have deployed into that server. For example, if you have an application that requires SAF authorization be enabled, you cannot specifically disable it for the JCICSX server and enable SAF authorization in the same Liberty server. In this case, you can create another JVM server that's dedicated to running the JCICSX server in your CICS region to work around this. If you don't create a separate server for JCICSX, you must follow instructions in “Scenario 3: Set up security in all my CICS regions, granting access to specific people” on page 371 to set up SAF authorization for the server.

In addition, when the client starts a session with the JCICSX server, a new CICS task is created to run under transaction CJXA and URI map DFHJXSU. Subsequent JCICSX requests for that session will run under the same task, and must be issued by the same user. Transaction CJXA is a category 2 transaction. If you have transaction attach security turned on, you also need to permit users to run transaction CJXA.

See “Typical scenarios and procedures” on page 370 for a discussion on how you might configure authentication and authorization for a number of typical scenarios.

Typical scenarios and procedures

Three scenarios are provided to cover the options described before.

- “Scenario 1: Allow all users to try it out in a single region with no authentication” on page 370
- “Scenario 2: Allow users to sign on using SAF identities, granting access to authenticated or specific users” on page 371
- “Scenario 3: Set up security in all my CICS regions, granting access to specific people” on page 371

Scenario 1: Allow all users to try it out in a single region with no authentication

To allow developers to test applications quickly in a development region, you can configure the remote JCICSX server to use no authentication. This task configures all security settings in `server.xml`.

Before you begin

Ensure that you have set up a Liberty JVM server to serve as the JCICSX server, by adding the JCICSX server feature (`cicsts:jcicsxServer-1.0`) to the `server.xml` file of your Liberty JVM server:

```
<featureManager>
    <feature>cicsts:jcicsxServer-1.0</feature>
</featureManager>
```

For more information, see Java development using JCICSX.

Procedure

1. If your server is not configured with the `appSecurity-2.0` feature to use Liberty security, no further configuration is needed. Any user can access the server with no credentials provided. For more information about `appSecurity`, see “Configuring security for a Liberty JVM server” on page 345.
2. If your server is configured with the `appSecurity-2.0` feature to use Liberty security:
   a. By default, the server only accepts authentication with a valid certificate. To allow users to be authenticated with a username and password, add the following line to the `server.xml` file:

      ```
      <webAppSecurity allowFailOverToBasicAuth="true"/>
      ```

      Otherwise, users get a 403 error when they access the server with a username and password.
   b. Create a basic user registry to authenticate users in `server.xml`. For instructions, see .
   c. By default, the server allows all the authenticated users defined in your user registry to access the servlet. Override the default setting to allow all users by adding the following snippet to `server.xml`:

      ```
      <authorization-roles id="com.ibm.cics.wlp.jcicsxserver">
        <security-role name="JCICSXUSER">
          <special-subject type="EVERYONE"/>
        </security-role>
      </authorization-roles>
      ```

      It changes the `special-subject` type from `ALL_AUTHENTICATED_USERS` to `EVERYONE` so that any user can access the servlet no matter what usernames or passwords they provide. If you don't specify the `EVERYONE` special subject, unauthenticated users who access the server get a 401 error.

3. If you are using transaction attach security, grant the CICS default user ID access to run the CJXA transaction.

Result

You have now configured remote JCICSX server to use no authentication for a CICS region.

Scenario 2: Allow users to sign on using SAF identities, granting access to authenticated or specific users

This is convenient if you already have a SAF registry to manage user identities. In this scenario, you use the SAF registry for authentication and configure role mapping in `server.xml` so that you don't need to configure new SAF EJBROLEs.

Before you begin

Ensure that you have set up a Liberty JVM server to serve as the JCICSX server, by adding the JCICSX server feature (`cicsts:jcicsxServer-1.0`) to the `server.xml` file of your Liberty JVM server:

```
<featureManager>
    <feature>cicsts:jcicsxServer-1.0</feature>
</featureManager>
```

For more information, see Java development using JCICSX.

Procedure

1. Enable CICS security, which integrates Liberty security, for the JCICSX server and configure it to use the SAF registry. For instructions, see “Configuring security for a Liberty JVM server” on page 345.
2. By default, the server only accepts authentication with a valid certificate. To allow users to be authenticated with a username and password, add the following line to the `server.xml` file:

   ```
   <webAppSecurity allowFailOverToBasicAuth="true"/>
   ```

   Otherwise, users get a 403 error when they access the server with a username and password.
3. After users are authenticated, all the authenticated users are allowed to use the application by default. If you want to restrict the application to specific users, bind users to the `JCICSXUSER` security role in `server.xml`:

   ```
   <authorization-roles id="com.ibm.cics.wlp.jcicsxserver">
     <security-role name="JCICSXUSER">
       <user name="USER"/>
     </security-role>
   </authorization-roles>
   ```
4. If you are using transaction attach security, grant the CICS default user ID access to run the CJXA transaction.

Result

You now have configured the remote JCICSX server to authenticate users using the SAF registry and to authorize them access to the service using role mapping in Enterprise Java.

Scenario 3: Set up security in all my CICS regions, granting access to specific people

In this scenario, you configure the JCICSX servers in all CICS regions that run under the default profile prefix to use SAF for authentication and authorization to grant specific user or user groups access. This is because SAF authorization makes it easy to share security settings across multiple CICS regions. When using SAF authorization, user to role mappings are used to map roles to EJBROLE resource profiles using the SAF role mapper. The server queries SAF to determine if the user has the required READ access to the EJBROLE resource profile. It's also convenient in that you can authorize an existing user group to use the JCICSX server by creating an EJBROLE.

Before you begin

Ensure that you have set up a Liberty JVM server to serve as the JCICSX server, by adding the JCICSX server feature (`cicsts:jcicsxServer-1.0`) to the `server.xml` file of your Liberty JVM server:

```
<featureManager>
    <feature>cicsts:jcicsxServer-1.0</feature>
</featureManager>
```

For more information, see Java development using JCICSX.

Procedure

1. Enable CICS security, which integrates Liberty security, for the JCICSX server and configure it to use the SAF registry. For instructions, see "Configuring security for a Liberty JVM server" on page 345.
2. By default, the server only accepts authentication with a valid certificate. To allow users to be authenticated with a username and password, add the following line to the `server.xml` file:

   ```
   <webAppSecurity allowFailOverToBasicAuth="true"/>
   ```

   Otherwise, users get a 403 error when they access the server with a username and password.
3. Add the`<safAuthorization>` element to `server.xml`, to use SAF authorization for role mapping:

   ```
   <safAuthorization id="saf"/>
   ```

4. Create the EJBROLE in RACF using the following RACF command:

   ```
   RDEFINE EJBROLE BBGZDFLT.com.ibm.cics.wlp.jcicsxserver.JCICSXUSER UACC(NONE)
   ```

   where BBGZDFLT is the default profile prefix, so this security configuration applies to all CICS regions that run under the default profile prefix. The profile prefix can be modified, making it easy for regions with the same profile prefix to share security settings. For more information, see "Authorization using SAF role mapping" on page 358.
5. Grant users READ access to those EJBROLE:

   ```
   PERMIT BBGZDFLT.com.ibm.cics.wlp.jcicsxserver.JCICSXUSER CLASS(EJBROLE) ACCESS(READ)
   ID(<user|group>)
   ```

6. If you are using transaction attach security, grant the CICS default user ID access to run the CJXA transaction.

Result

You have configured the remote JCICSX servers in all CICS regions running under the default profile prefix to authenticate users using the SAF registry and to authorize specific users or user groups with access to the services.

Configuring SSL (TLS) for a Liberty JVM server using a Java keystore

You can configure a Liberty JVM server to use SSL for data encryption, and optionally authenticate with the server by using a client certificate. Certificates can be stored in a Java keystore or in a SAF key ring such as in RACF.

About this task

Enabling SSL in a Liberty JVM server requires adding the **`transportSecurity-1.0`** Liberty feature, a keystore, and an HTTPS port. CICS automatically creates and updates the `server.xml` file. Autoconfiguring always results in the creation of a Java keystore.

It is important to understand that any web request to a Liberty JVM server uses the JVM support for TCP/IP sockets and SSL processing, not CICS sockets domain.

Procedure

- To use autoconfigure to configure SSL, complete the following steps:
  a) Ensure autoconfigure is enabled in the JVM profile by using the JVM system property **-Dcom.ibm.cics.jvmserver.wlp.autoconfigure=true**.
  b) Set the SSL port by setting the JVM system property **-Dcom.ibm.cics.jvmserver.wlp.server.https.port** in the JVM profile.
  c) Restart the JVM server to add the necessary configuration elements to `server.xml`.

Results

SSL for a Liberty JVM server is successfully configured.

Configuring SSL (TLS) for a Liberty JVM server using RACF

You can configure a Liberty JVM server to use SSL for data encryption, and optionally authenticate with the server by using a client certificate. Certificates can be stored in a Java keystore or in a SAF key ring such as a RACF keyring.

About this task

Enabling SSL in a Liberty JVM server requires adding the **transportSecurity-1.0** Liberty feature, a keystore, and an HTTPS port. You edit the `server.xml` file to add the required elements and values. You must follow the manual procedure if you want to use a RACF key ring.

It is important to understand that any web request to a Liberty JVM server uses the JVM support for TCP/IP sockets and SSL processing, not CICS sockets domain.

Procedure

- To manually configure SSL, you need to create a signing certificate. Use this signing certificate to create a server certificate. Then, export the signing certificate to the client web browser where it is used to authenticate the server certificate.
  a) Create a certificate authority (CA) certificate (signing certificate). An example, using RACF commands, follows:

```
RACDCERT GENCERT
  CERTAUTH
  SUBJECTSDN(CN('CICS Sample Certification Authority')
   O('IBM')
   OU('CICS'))
   SIZE(2048)
  WITHLABEL('CICS-Sample-Certification')
```

  The SIZE of the certificate should be a minimum of 2048 bits. For more information, see the RACF RACDCERT GENCERT (Generate certificate) command.
  b) Create a server certificate that uses the signing certificate from step 2, where <userid> is the CICS region user ID. The hostname is the host name of the server that the Liberty server HTTPS port is configured to use.

```
RACDCERT ID(<userid>)
GENCERT
  SUBJECTSDN(CN('<hostname>')
   O('IBM')
   OU('CICS'))
   SIZE(2048)
   SIGNWITH (CERTAUTH LABEL('CICS-Sample-Certification'))
  WITHLABEL('<userid>-Liberty-Server')
```

  The SIZE of the certificate should be a minimum of 2048 bits. For more information, see the RACF RACDCERT GENCERT (Generate certificate) command.
  c) Connect the signing certificate and server certificate to a RACF key ring.

You can use RACF with the following command, and replace the value of <keyring> with the name of the key ring you want to use. Replace the value of <userid> with the CICS region user ID.

```
RACDCERT ID(<userid>) CONNECT(RING(<keyring>)
       LABEL('CICS-Sample-Certification')
       CERTAUTH)

RACDCERT ID(<userid>) CONNECT(RING(<keyring>)
       LABEL('<userid>-Liberty-Server'))
```

Export the signing certificate to a CER file:

```
RACDCERT CERTAUTH EXPORT(LABEL('CICS-Sample-Certification'))
       DSN('<userid>.CERT.LIBCERT')
       FORMAT(CERTDER)
       PASSWORD('password')
```

FTP the exported certificate in binary to your workstation, and import it into your browser as a certificate authority certificate.

d) Edit the `server.xml` file and add the SSL feature, and the keystore. Set the HTTPS port (value is 9443 in the following example) and restart your CICS region. The SAF key ring must be specified in the URL form `safkeyring://<userid>/<keyring>`.

If you are running Java 11, the location must be `location="safkeyringjce://<userid>/<keyring>"`.

The <userid> value must be set to the CICS region user ID and the <keyring> value must be set to the name of the key ring. The password field is not used for accessing the SAF key ring and must be set to `password`.

```
<featureManager>
  ...
       <feature>transportSecurity-1.0</feature>
</featureManager>
...
<httpEndpoint host="*" httpPort="9080" httpsPort="9443"
    id="defaultHttpEndpoint"/>
...
.
<keyStore filebased="false" id="racfKeyStore"
        location="safkeyring://<userid>/<keyring>"
        password="password"
        readOnly="true"
        type="JCERACFKS"/>
<ssl id="defaultSSLConfig" keyStoreRef="racfKeyStore"
      sslProtocol="SSL_TLS"
      serverKeyAlias="<userid>-Liberty-Server" />
```

Results

SSL for a Liberty JVM server is successfully configured.

Configuring SSL (TLS) for remote JCICSX API development

When configuring a Liberty JVM server in CICS for remote JCICSX API development, you can configure it to use SSL for data encryption.

About this task

To enable the remote development functionality of the JCICSX API, a Liberty JVM server is required in CICS to receive requests from the developer's local Liberty JVM server. To enable SSL communication between the remote Liberty JVM server in CICS and the local Liberty JVM server on the developer's machine, the remote server must be configured to use SSL and its certificate must be trusted by the local Liberty server.

Before you begin

- Configure the remote Liberty JVM server for user authentication and authorization. For instructions, see "Configuring security for remote JCICSX API development" on page 367
- Enable SSL in the Liberty JVM server in CICS. For instructions, see "Configuring SSL (TLS) for a Liberty JVM server using a Java keystore" on page 372.
- Ensure that a local Liberty JVM server is configured to make remote JCICSX requests. This is usually done by the application developer on their local machine. For instructions, see **Extra configuration for remote development (local workstation)** in Java development using JCICSX.

Procedure

- **For system programmers, configure the remote Liberty JVM server as follows:**

  As a system programmer, you need to generate a certificate for the remote Liberty JVM server with its host name registered.

  a) **Register the host name of the remote server in the `server.xml` file.**

  By default the host name of the Liberty server is `localhost`. In this case your SSL connection will fail because the certificate will be registered to `localhost` while your client will be trying to connect to the host name of your CICS region. To override the default `localhost` host name, add the `defaultHostName` variable to your `server.xml` file:

  ```
  <variable name="defaultHostName" value="your-hostname"/>
  ```

  where *your-hostname* is the host name of the remote Liberty JVM server. For more information, see Setting the default host name of a Liberty server.

  b) **Generate a new copy of the remote Liberty JVM server's public certificate.**

  After setting the default host name of your server, you must regenerate the certificate for it.

  a. Stop the Liberty JVM server.

  b. Delete the Java keystore that stores the certificates. It is the `key.p12` file located in `{server.config.dir}/resources/security`.

  c. Start the liberty server.

  d. Verify that a new `key.p12` file is regenerated.

  c) **Verify that the host name is correct in the certificate using OpenSSL or the Java keytool utility:**

  – If you're using OpenSSL, input this command to show the certificates of the remote Liberty JVM server:

  ```
  $ openssl s_client -showcerts -connect remotejcicsxserver.com:portNo
  ```

  – If you're using the Java keytool utility:

  a. Navigate to the folder of the keystore on the remote Liberty server at: `{server.config.dir}/resources/security`.

  b. If the local Liberty server is at 19.0.0.3 or later, which is the minimum version required to use the client-side tooling of remote JCICSX development, and that autoconfigure is enabled for the remote Liberty server to use SSL, the remote Liberty server will have created a keystore using default values. In this case, use this command to show the certificates stored in the auto-created Java keystore:

  ```
  keytool -list -keystore key.p12 -storepass defaultPassword -storetype PKCS12 -v
  ```

  Otherwise, substitute values in for `storepass` and `storetype` according to your custom configuration.

  c. In the output, verify that `CN = your-hostname` shows the host name of the remote JVM server, instead of the default `localhost` value. Otherwise, repeat Step "2" on page 375.

- **For application developers, configure the local Liberty JVM server to trust the remote server's public certificate as follows:**

  By default, the local Liberty JVM server might refuse to connect to the remote Liberty JVM server because it does not trust the public certificate that the remote Liberty provides during the SSL handshake. Therefore, the application developer must download a copy of the certificate from the remote Liberty server and add it to the truststore used by the local Liberty server.

  a) Download a copy of the public certificate from the remote Liberty JVM server:

  To use OpenSSL:

  a. Run the following command to show the certificates of the remote server, where *your-hostname* and *your-port* are the host name and port number of your remote Liberty JVM server:

  ```
  openssl s_client -showcerts -connect your-hostname:your-port
  ```

  b. From the output, copy the first certificate. Include the following lines and the information between these lines:

  ```
  "-----BEGIN CERTIFICATE-----"
  "-----END CERTIFICATE-----"
  ```

  c. Paste the certificate into a new file with a `.cer` extension, for example, `publicKey.cer`.

  Note: Note: Be sure not to include additional lines in this file; otherwise the certificate won't be added to you local Liberty truststore successfully.

  If you have access to the remote Liberty server, you can also use the Java keytool utility to download the certificate:

  a. Navigate to the keystore on your remote Liberty JVM server. The file path is `{server.config.dir}/resources/security/key.p12`.

  b. Use the Java keytool utility to create a public certificate:

  ```
  keytool -rfc -export -keystore key.p12 -alias default -file /your/save/location/
  public-remote.cer -v -storepass yourKeyStorePassword -storeType yourType
  ```

  where *yourType* is the keystore type, which defaults to PKCS12. For more information, see Liberty default keystore type changed to PKCS12.

  b) Navigate to the folder of the keystore on the local Liberty JVM server: `{server.config.dir}/resources/security`.

  c) Import the public certificate that the system programmer downloaded into the truststore of the local Liberty, using the following command:

  ```
  keytool -importcert -file /cert/location/public-remote.cer -keystore key.p12 -storepass
  localPassword -storetype yourType -trustcacerts -v
  ```

  where *yourType* is the keystore type, which defaults to PKCS12. For more information, see Liberty default keystore type changed to PKCS12.

  d) Restart the local Liberty JVM server to pick up the new certificate.

What to do next

The application developer can add a JCICSX resource to the local Liberty JVM server to check whether the connection is working. Samples can be found at JCICSX samples in GitHub.

Setting up SSL (TLS) client certificate authentication in a Liberty JVM server

SSL client certificate authentication allows the client and server to provide certificates to the opposite party for mutual verification. It is often used in situations where an extra level of authentication is required because of security concerns.

Before you begin

You must complete the task Configuring SSL (TLS) for a Liberty JVM server using RACF. If you do not already have your CICS Liberty security set up, you must complete Configuring security for a Liberty JVM server before proceeding.

About this task

The following setup information assumes that you are using RACF keystores to store your certificates for SSL client certificate authentication.

Procedure

1. Create a personal certificate using a signing certificate and associate the personal certificate with a RACF user ID.

   Then, export the personal certificate to a data set in CER format and then FTP in binary to your work station. Import the personal certificate to the web browser as a personal certificate. When the certificate is imported into the web browser, it can supply an SSL client certificate and connect to the HTTPS port in the Liberty server. Use the following RACF command, where <clientuserid> is the RACF user ID and <hostname> is the host name of the client computer.

   ```
   RACDCERT ID(<clientuserid>)
   GENCERT
     SUBJECTSDN(CN('<hostname>')
      O('IBM')
      OU('CICS'))
      SIZE(2048)
      SIGNWITH (CERTAUTH LABEL('CICS-Sample-Certification'))
     WITHLABEL('<clientuserid>-certificate')
   ```

   Export the personal certificate as you have done earlier in this step.

   ```
   RACDCERT ID(<clientuserid>)
     EXPORT(LABEL('<clientuserid>-certificate'))
      DSN('USERID.CERT.CLICERT')
      FORMAT(PKCS12DER)
      PASSWORD('password')
   ```

   Update the `server.xml` SSL element to support SSL client certificate authentication:

   ```
     <ssl id="defaultSSLConfig" keyStoreRef="racfKeyStore"
        sslProtocol="SSL_TLS"
        serverKeyAlias="<userid>-Liberty-Server"
        clientAuthenticationSupported="true"/>
   ```

   Additionally, if you want to ensure all clients must supply a valid SSL client certificate, add the **`clientAuthentication`** attribute to the SSL element as follows:

   ```
     <ssl id="defaultSSLConfig" keyStoreRef="racfKeyStore"
        sslProtocol="SSL_TLS"
        serverKeyAlias="<userid>-Liberty-Server"
        clientAuthenticationSupported="true"
        clientAuthentication="true"/>
   ```

2. You can authenticate a web request in CICS under the identity of the client user ID in step 2. Then, deploy the web application with a `login-config` element for CLIENT-CERT in the `web.xml`. The `web.xml` file can be found inside the source files for the web application that you are deploying.

```
<login-config>
          <auth-method>CLIENT-CERT</auth-method>
</login-config
```

Instead, if you want to allow failover to HTTP basic authentication if SSL client certificate authentication is not configured, add the webAppSecurity element to `server.xml`.

```
<webAppSecurity allowFailOverToBasicAuth="true" />
```

3. Finally, set up CICS transaction security to authorize access to the CICS transaction based on the authenticated client user ID.

   For further information, see "Authorizing users to run applications in a Liberty JVM server" on page 354.

Using the syncToOSThread function

You can use the syncToOSThread function of Liberty in a CICS Liberty JVM server. SyncToOSThread enables a Java subject, authenticated by Liberty, to be synchronized with the operating system (OS) thread identity. Without syncToOSThread, the operating system thread identity defaults to be the CICS region user ID, this is the identity used to authorize access to resources outside of CICS control such as zFS files. With syncToOSThread in effect, the user's subject is used to access these operating system resources.

About this task

Enabling syncToOSThread requires the Liberty appSecurity-1.0 and zosSecurity-1.0 features. These features are included with the cicsts:security-1.0 feature. You must also define the syncToOSThread configuration element in the Liberty `server.xml` and add a special `<env-entry/>` to the application's deployment descriptor (`web.xml`). In addition, the SAF registry must be used for authentication, the angel process must be up and running, and the server must be connected to the angel process. For more information about the angel process, see Process types on z/OS.

Procedure

1. Configure the syncToOSThread configuration element in the Liberty `server.xml` and add the required `<env-entry/>` to each web application's deployment descriptor by following steps 1 and 2 in
2. Grant the Liberty server permission to perform syncToOSThread operations by configuring SAF with either of the following profiles:

   - Grant the CICS region user ID CONTROL access to the BBG.SYNC.*<profilePrefix>* profile in the FACILITY class, where *<profilePrefix>* is specified on the `<safCredentials />` element. This allows the Liberty server to synchronize any Java subject with the OS thread identity:

     ```
     PERMIT BBG.SYNC.<profilePrefix> ID(<serverUserId>) ACCESS(CONTROL) CLASS(FACILITY)
     ```

   - Grant the CICS region user ID READ access to the BBG.SYNC.*<profilePrefix>* profile in the FACILITY class. Additionally, grant the CICS region user ID READ access to one or more BBG.SYNC.*<AuthUserid/>* profiles in the SURROGATE class, one for each authenticated user ID to be synchronized with the OS identity:

     ```
     PERMIT BBG.SYNC.<profilePrefix> ID(<serverUserId>) ACCESS(READ) CLASS(FACILITY)
     PERMIT BBG.SYNC.<AuthUserid> ID(<serverUserId>) ACCESS(READ) CLASS(SURROGAT)
     ```

Restriction: A servlet configured as the welcome page in `web.xml`, does not support the syncToOSThread function.

Enabling a Java security manager

By default, Java applications have no security restrictions placed on activities requested of the Java API. To use Java security to protect a Java application from performing potentially unsafe actions, you can enable a security manager for the JVM in which the application runs.

About this task

The security manager enforces a security policy, which is a set of permissions (system access privileges) that are assigned to code sources. A default policy file is supplied with the Java platform. However, to enable Java applications to run successfully in CICS when Java security is active, you must specify an additional policy file that gives CICS the permissions it requires to run the application.

You must specify this additional policy file for each kind of JVM that has a security manager enabled. CICS provides some examples that you can use to create your own policies.

Notes: Enabling a Java security manager is not supported in a Liberty JVM server.

- The OSGi security agent example creates an OSGi middleware bundle called `com.ibm.cics.server.examples.security` in your project that contains a security profile. This profile applies to all OSGi bundles in the framework in which it is installed.
- The `example.permissions` file contains permissions that are specific to running applications in a JVM server, including a check to ensure that applications do not use the `System.exit()` method.
- CICS must have read and execute access to the directory in zFS where you deploy the OSGi bundle.

For applications that run in the OSGi framework of a JVM server:

Procedure

1. Create a plug-in project in the IBM CICS SDK for Java and select the supplied OSGi security agent example.
2. In the project, select the `example.permissions` file to edit the permissions for your security policy.
   a) Validate that the CICS zFS and Db2 installation directories are correctly specified.
   b) Add other permissions as necessary.
3. Deploy the OSGi bundle to a suitable directory in zFS such as `/u/bundles`.
4. Edit the JVM profile for the JVM server to add the OSGi bundle to the `OSGI_BUNDLES` option before any other bundles:

   ```
   OSGI_BUNDLES=/u/bundles/com.ibm.cics.server.examples.security_1.0.0.jar
   ```

5. Add the following Java property to the JVM profile to enable security.

   ```
   -Djava.security.policy=all.policy
   ```

6. Add the following Java environment variable to the JVM profile to enable security in the OSGi framework:

   ```
   org.osgi.framework.security=osgi
   ```

7. To allow the OSGi framework to start with Java 2 security, add the following policy:

   ```
   grant { permission java.security.AllPermission; };
   ```

8. Save your changes and enable the JVMSERVER resource to install the middleware bundle in the JVM server.
9. Optional: Activate Java 2 security.
   a) To activate a Java 2 security policy mechanism, add it to the appropriate JVM profile. You must also edit your Java 2 security policy to grant appropriate permissions.
   b) To use JDBC or SQLJ from a Java application with a Java 2 security policy mechanism active, use the IBM Data Server Driver for JDBC and SQLJ.
   c) To activate a Java 2 security policy mechanism, edit the JVM profile.

d) Edit the Java 2 security policy to grant permissions to the JDBC driver, by adding the lines that are shown in Example 1. In place of db2*xxx*, specify a directory below which all your Db2 libraries are located. The permissions are applied to all the directories and files below this level. This enables you to use JDBC and SQLJ.

e) Edit the Java 2 security policy to grant read permissions, by adding the lines that are shown in Example 2. If you do not add read permission, running a Java program produces AccessControlExceptions and unpredictable results. You can use JDBC and SQLJ with a Java 2 security policy.

Example 1:

```
grant codeBase "file:/usr/lpp/db2xxx/-" {
  permission java.security.AllPermission;
};
```

Example 2:

```
grant {

// allows anyone to read properties
permission java.util.PropertyPermission "*", "read";

};
```

Results

When the Java application is called, the JVM determines the code source for the class and consults the security policy before granting the class the appropriate permissions.

Chapter 12. Security for Node.js applications

Node.js applications typically handle user authentication themselves, and do not often pass on user credentials to the back-end systems they interact with. Therefore, in most cases securing a Node.js application is handled in the same way when it is running in CICS as when it is running in any other environment.

Node.js applications in CICS always run under the CICS region user ID. This is important for Node.js applications that interact with the file system. The CICS region user ID must have the correct permissions for any files that are accessed by the Node.js application.

Security for invoke requests to CICS

CICS tasks that are started by using a locally optimized invoke request from a Node.js application that is running in CICS will:

1. Run (by default) under the CNJW transaction ID
2. Run (by default) under the CICS region's default user ID (typically CICSUSER)

Both the default transaction ID and the default user ID can be modified, on a per URI basis, by using a URIMAP resource. The URI is passed to the invoke function by the Node.js application. For example, if a URIMAP is installed which indicates that requests for path `/examples/updateAccount` are to be mapped to transaction ID *TEST* and user ID *WORKER* then any work that is started for that URI path will override the default values as requested. In this example, the Node.js application must supply a URI with path `/examples/updateAccount` as a parameter to the invoke function. Therefore, the complete URI passed to the invoke function might be `http://example.org:12345/examples/updateAccount`.

If the URIMAP indicates that the target transaction ID is CPIH (which is the default value when USAGE(PIPELINE) is specified on a URIMAP), then any tasks for that URI invoked from Node.js will run under the CNJW transaction ID.

There is no mechanism for specifying an alternative user ID for each individual request to the same URI.

For more information on using the invoke function, see Calling CICS services.

Chapter 13. Security for CICS web support

When CICS is connected to the Internet, apply security measures to prevent unauthorized access to CICS applications and data and also to prevent third parties obtaining private information.

Consider security throughout the development process for your CICS web support architecture, as part of the design of your CICS web support applications and utility programs, and when creating resource definitions for the relevant CICS facilities. The subtopics summarize the measures that you can use to enhance the security of your CICS web support implementation.

CICS as an HTTP server: authentication and identification

For CICS as an HTTP server, you specify authentication schemes by the AUTHENTICATE attribute of the TCPIPSERVICE definition. Identification is obtained in connection with the authentication process, or it can be supplied by CICS if authentication is not needed.

Obtaining authentication and identification from web clients is a key step in protecting your CICS system from access by unauthorized users.

Use TCPIPSERVICE resource definitions to specify the security measures that are applied for CICS as an HTTP server. For each port that you use for CICS web support, the TCPIPSERVICE resource definition specifies these attributes:

- Whether or not SSL is used for the port
- The authentication scheme that is used for the port
- The realm for basic authentication

Authentication

Two authentication schemes are supported by CICS for use with the HTTP protocol:

- **Basic authentication** is part of HTTP that enables a client to authenticate and identify itself to a server by providing a user ID and password or password phrase. This information is encoded using base-64 encoding, which is simple to decode. Therefore, using basic authentication as the sole means of authentication is appropriate only when the password cannot be intercepted. In most environments, use basic authentication with SSL, so that SSL encryption protects the user ID and password information.
- **SSL client certificate authentication** is a more secure method of authenticating a client, using a client certificate that is issued by a trusted third party (or Certificate Authority), and sent using SSL encryption. Refer to SSL authentication. A client certificate does not contain a user ID that can be used for identification in CICS. To achieve identification, you can associate the client certificate with a user ID in RACF or an equivalent security manager, either before the certificate is used, or automatically (using basic authentication) when the client makes its request. The RACF user ID becomes the client user ID each time the certificate is used, as described in Associating a RACF user ID with a certificate.

Creating TCPIPSERVICE resource definitions for CICS web support tells you how to set up a TCPIPSERVICE definition for CICS web support that specifies one of these authentication schemes.

When you use basic authentication or client certificate authentication, CICS handles the process of requesting authentication from the user, decoding the authentication information if necessary, checking the supplied authentication against the security manager database, and rejecting the request if the authentication is not acceptable. An analyzer program or user-written application program is called only after the authentication is verified and accepted.

All the user IDs used by web clients must have a user profile in RACF or your equivalent external security manager. Refer to RACF profiles.

Note: CICS uses password verification to verify a user ID during the processes described here. CICS enforces a full verification request once a day for each user ID that is used to log on to the CICS region. The full verification request using the RACROUTE REQUEST=VERIFY macro makes RACF record the date and time of last access for the user ID, and write user statistics.

For basic authentication, if the password or password phrase supplied by the user has expired, CICS prompts the user for a new password or password phrase and helps the user to resubmit the request. The CICS-supplied utility program DFHWBPW is used. You can customize the text on the web pages that CICS displays to the user during this process, as described in "Password expiry management for HTTP basic authentication" on page 385.

For client certificate authentication, CICS verifies the supplied certificate by checking it against the security manager database, and, optionally, against any certificate revocation list that you have set up. A user-written application can examine information obtained by this process, if this information is useful for determining how to process the request. Use the EXTRACT CERTIFICATE command to retrieve these items:

- Components of the issuer's or the subject's distinguished name. SSL authentication explains distinguished names.
- The RACF user ID associated with the certificate.

Identification

Identification takes place when you obtain a user ID for the web client. The ID is obtained from the web client:

- During basic authentication
- By the association of a user ID with a client certificate

For application-generated responses only, CICS can supply a user ID on behalf of the web client:

- In an analyzer program that is used in the processing path for the application-generated response. (This ID can override a user ID obtained for the web client.)
- In the URIMAP definition for the request. (This ID cannot override a user ID obtained for the web client.)
- As the CICS default user ID, if no other can be determined.

Note that, if you supply a user ID on behalf of the web client, the identity of the client is not authenticated. Supply a user ID only when communicating with your own client system, which has already authenticated its users and communicates with the server in a secure environment. Identifying HTTP users explains in more detail how the user ID is determined, depending on the settings for the TCPIPSERVICE definition.

When the client has been identified, the client user ID can be authorized for access to CICS resources like any other user ID, using RACF or an equivalent external security manager. You can choose to apply resource-level security to any or all of the individual resources that the web client is accessing in CICS, such as web pages stored as CICS document templates, or z/OS UNIX files, or CICS commands used by the application that provides the response. "CICS system and resource security for CICS web support" on page 387 explains how to secure these resources and how to remove resource level security if you do not want it.

CICS as an HTTP client: authentication and identification

When you make an HTTP client request through CICS, a server or proxy might require you to perform basic authentication, proxy authentication, or SSL client certificate authentication.

You can perform basic authentication using the AUTHENTICATE option of your WEB SEND or WEB CONVERSE command. Your user application carries out proxy authentication. You supply a client certificate using a URIMAP definition.

Your client application might be asked to authenticate itself in the following ways:

- **Basic authentication** allows you to provide a user name and password for access to specific information. When you make a request to a server, the server might send you a response with a

401 status code, and a WWW-Authenticate header. The header names the realm for which basic authentication is required. To receive the information you requested, provide the user name and password, and CICS resends the request with an Authorization header, specifying your user name and password, to allow you access to the realm. CICS can also send an Authorization header directly to a server that is expecting it, thus eliminating the need for a 401 response. CICS converts the user name and password to ASCII and applies base-64 encoding, as required by the basic authentication protocol. So you can supply your credentials in normal characters through the WEB SEND or WEB CONVERSE command, or through the XWBAUTH user exit. See Providing credentials for basic authentication and HTTP basic authentication.

- **Proxy authentication** is initiated by a proxy server. For proxy authentication, the status code for the response is 407, the challenge header from the proxy server is Proxy-Authenticate, and the response header is Proxy-Authorization. CICS does not support this protocol.
- **SSL client certificate authentication** uses a client certificate, which is issued by a trusted third party (or Certificate Authority). A server might require you to provide this authentication when you are making an HTTPS request. Configuring CICS to use SSL tells you how to obtain a certificate and store it in a key ring in the RACF database or equivalent external security manager. If a server does request a client certificate, CICS supplies the certificate label, which is specified in the URIMAP definition that was used on the WEB OPEN command for the connection. Alternatively, you can directly specify the certificate label as an option in the WEB OPEN command. If you use a URIMAP definition but do not specify a certificate label, the default certificate defined in the key ring for the CICS region user ID is used.

Some servers might ask you to provide other types of authentication or identification. If you cannot provide acceptable authentication or identification to a server, your request is rejected. For basic authentication or proxy authentication, the status code used when a server rejects your request is the same as the status code for the challenge (401 for a server or 407 for a proxy). If you respond to a challenge but then receive a further response with one of these status codes, the authorization information that you used is not valid.

Password expiry management for HTTP basic authentication

When basic authentication is used for an HTTP connection, CICS web support checks the user ID and password in the external security manager. If the password has expired, the CICS-supplied utility program DFHWBPW is used to prompt the user to select a new password. You can customize or replace the pages presented to the user by DFHWBPW.

DFHWBPW is used only for password expiry management when the TCPIPSERVICE definition that applies to the request is defined with the BASIC, AUTOREGISTER, or AUTOMATIC option for the AUTHENTICATE attribute. Although DFHWBPW has a structure similar to a converter program, it is not part of the normal CICS web support processing path, so you do not need to add code to it for any other purpose. When the user has selected a new password, DFHWBPW restarts the request submission by redirecting the client to the URL for the original request, so that the complete processing path for the request occurs as normal.

DFHWBPW presents two web pages to the user:

1. Password prompt page. This page contains two elements:
   a. A message about password validity. The initial message displayed to the user states that the password has expired. A user ID can have both a standard password and a password phrase. Passwords between 9 and 100 characters in length are password phrases; passwords of 8 characters or less are standard passwords. Standard passwords and password phrases operate independently of each other. If a standard password has expired, it must be replaced with a new standard password. Similarly, if a password phrase has expired, it must be replaced with a new password phrase. If the user's attempt to change the password fails (for example, the two supplied copies of the new password do not match), further messages are displayed to explain the problem.
   b. An HTML form for the user to change the password.
2. Confirmation and request refresh page. This page confirms that the expired password has been successfully replaced, and provides a refresh tag and URL link so that the request can be remade automatically or manually.

DFHWBPW builds these web pages using three CICS document templates: DFHWBPW1, DFHWBPW2, and DFHWBPW3. The CICS-supplied definitions for these templates define them as loadable programs; that is, they are of type PROGRAM(DFHWBPW1) and so on. The definitions are in the CICS-supplied resource definition group DFHWEB. You can change these definitions by copying them to another group and using the resource definition ALTER command to change them so that the templates are derived from a different source. Alternatively, you can leave the resource definitions unchanged, and modify the programs that are loaded instead. The three programs DFHWBPW1, DFHWBPW2, and DFHWBPW3 are assembler language data-only modules, and their source is shipped to you in corresponding members of the CICS sample library, SDFHSAMP. You can modify these samples and reassemble and link-edit them into one of your normal CICS program libraries that are concatenated into the DFHRPL data definition statement.

Note: To avoid unnecessary output, ensure that you specify the **NOPROLOG** and **NOEPILOG** parameters when assembling DFHWBPW1, DFHWBPW2 and DFHWBPW3.

The content and function of each of the DFHWBPW templates is as follows:

DFHWBPW1
Part of the password prompt page. Provides the HTML page heading for the page, and sets symbols for the possible password validity messages (using the server-side include technique for setting symbols). The messages provide the following information:

message.1
Password has expired.

message.2
The entered user ID is invalid.

message.3
The two copies of the proposed new password do not match.

message.4
The previous password entered (the one that has just expired) is not correct.

message.5
The proposed new password is not permitted by the external security manager, because of password quality rules.

message.6
The user ID is now revoked.

The DFHWBPW program selects the appropriate symbol to insert into the document for the password prompt page. You can customize DFHWBPW1 to change the page heading and title, or alter the `body` tag to change the page colors or background. You can also change the content of the message symbols.

DFHWBPW2
Part of the password prompt page. Builds an HTML form where the user can input a user ID, the old (expired) password (or password phrase), and two identical copies of a proposed new password. You can customize DFHWBPW2 to change the text used to prompt the user, or otherwise change the layout of the page. However, you must not modify the contents of the `form` tag, or any of the `input` tags. If you do, DFHWBPW might not work as intended.

DFHWBPW3
Confirmation and request refresh page. The text notifies the user that the expired password was successfully replaced, and explains that the user will shortly be prompted by the client to enter the password again. You can customize the text and layout of the page.

DFHWBPW3 restarts the request process. It contains a `meta http-equiv="Refresh"` tag that causes an automatic redirection after ten seconds to the page that the user had originally requested when the expired password was detected. You can change the time limit on this tag or remove it if you do not want users to be redirected automatically. However, the modified page must always contain a link forward to the originally requested page. The URL for that page is in the symbol `&dfhwbpw_target_url;` . Restarting the request process means that, if the web client has cached

the old password, it can be replaced with the new password immediately, and also means that the CICS web support processing path is unaffected.

CICS system and resource security for CICS web support

When CICS is an HTTP server, the CICS system must be protected from access by unauthorized users. If a system is not properly protected, users might be able to access confidential data or obstruct the system to cause denial of service to other users.

To police access to CICS web support in general, you request identification from each user that makes an HTTP client request and then authenticate the identity stated by the user. You use the TCPIPSERVICE definitions for inbound ports to specify these requirements. Refer to "CICS as an HTTP server: authentication and identification" on page 383.

All the user IDs used by web clients must have a user profile in RACF or your equivalent external security manager. Refer to RACF profiles.

When you have obtained an authenticated user ID for a web client, you can use this ID to implement resource-level security for the resources in the CICS region that you are using to provide the response. The procedure varies for each type of response:

- Application-generated responses
- Static responses, using a URIMAP definition that provides a CICS document template as the response
- Static responses, using a URIMAP definition that provides a z/OS UNIX Systems Services file as the response

For application-generated responses, CICS system defaults specify that no resource security checking is carried out, but transaction security checking is carried out (specifically, transaction-attach security for the alias transaction). Assuming that transaction security is active in your CICS region, you must therefore take some actions relating specifically to security for application-generated responses, even if you do not plan to use web client authenticated user IDs for security checking.

For static responses, transaction-attach security does not apply to web client user IDs. However, CICS system defaults specify that resource-level security checking *is* carried out if a user ID is available for web clients. If you are obtaining authenticated user IDs from web clients, you must therefore either set up resource permissions for these user IDs or take action to disable resource-level security checking.

Whether or not you choose to implement resource-level security using web client user IDs for every response provided by CICS web support, you must provide the following protection:

- Implement measures to protect inbound ports against unauthorized or malicious access.
- Protect CICS system components from modification by unauthorized users, and ensure that authorized users have the correct access to them.

Security for inbound ports

A TCPIPSERVICE resource definition defines each port used for CICS web support. The TCPIPSERVICE definition specifies security options for the port, including whether SSL is used and the level of authentication that is requested from clients. Ports must be guarded against unauthorized or malicious access.

Creating TCPIPSERVICE resource definitions for CICS web support explains how to create definitions for ports.

To help keep ports secure:

- Specify the MAXDATALEN attribute on every TCPIPSERVICE definition. This option limits the maximum amount of data that CICS accepts for a single request, and it helps to defend CICS against denial of service attacks involving the transmission of large amounts of data.
- Use Secure Sockets Layer (SSL) wherever you want to ensure that your interaction with the web client remains confidential and cannot be intercepted by a third party. The use of SSL is particularly important

when confidential data is being transmitted or when authorization such as a user ID and password are being passed to the server. Refer to "SSL with CICS web support" on page 392.

If you do experience unusual activity on one or more of your CICS web support ports, use CICS system commands to shut down CICS web support at different levels (a single request, a virtual host, a port, or the whole of CICS web support), without shutting down the CICS system. Refer to Rejecting HTTP requests in Administering.

URIMAP resource definitions name either HTTP or HTTPS as the scheme for the request. A URIMAP specifying HTTP accepts web client requests made using either HTTP or the more secure HTTPS. A URIMAP specifying HTTPS accepts only web client requests that are made using HTTPS.

When a URIMAP definition with HTTPS matches a request from a web client, CICS checks that the inbound port used by the request is using SSL. If SSL is not specified for the port, the request is rejected with a 403 (Forbidden) status code. When the URIMAP definition applies to all inbound ports, this check ensures that a web client cannot use an unsecured port to access a secured resource. No check is carried out for a URIMAP definition that specifies HTTP, so web clients can use either unsecured or secured (SSL) ports to access these resources.

Security for CICS system components

As with any other CICS resource, you must protect CICS system components used in CICS web support from modification by unauthorized users. You must also ensure that authorized users, particularly the CICS region, have the required authority to use these components.

A number of components, such as application programs and resource definitions, are used to control CICS web support. Refer to Components of CICS web support. If you do not secure these components against unauthorized access, the security of your CICS web support architecture might be compromised. For example, a user with access to the TCPIPSERVICE definition for a port might remove the requirement for a web client to use SSL or to provide identification. Implementing RACF protection in a single CICS region explains how to secure CICS transactions, resources, and commands against unauthorized use.

For some CICS system components, you might have to set up additional authorities to allow access to authorized users:

- For URIMAP resources, additional authority might be required to set a user ID for the web client. If surrogate user checking is enabled in the CICS region (with XUSER=YES specified as a system initialization parameter), CICS checks that the user ID used to install the URIMAP definition is authorized as a surrogate of the user ID specified for the USERID attribute.
- You can use document templates to produce the body of a response from CICS as an HTTP server, or the body of a request from CICS as an HTTP client. You define them by DOCTEMPLATE resource definitions. If the document templates are stored in partitioned data sets, the CICS region user ID must have READ authority for the data set.
- You can use z/OS UNIX Systems Services files to produce the body of a static response from CICS as an HTTP server. You can specify them under their own names or define them by DOCTEMPLATE resource definitions. When a z/OS UNIX file is used, the CICS region must have permissions to access z/OS UNIX, and it must have permission to access the z/OS UNIX directory containing the file, and the file itself. Refer to Giving CICS regions access to z/OS UNIX directories and files.

Resource and transaction security for application-generated responses

If you have obtained an authenticated user ID for a web client, which has a profile in your security manager, this user ID is applied to the alias transaction that is used for the application-generated response.

About this task

You either give appropriate permissions to the web client user IDs or you supply your own standard user ID as an override. Whether or not you decide to use web client user IDs for resource security checking, you must ensure that the user ID for the alias transaction has the appropriate permissions.

You define alias transactions by TRANSACTION resource definitions. The alias transaction for each application-generated response is specified by the URIMAP definition for the request or by an analyzer program. The default is the CICS-supplied alias transaction CWBA, which applies when either web-aware applications or COMMAREA applications are used to provide the response.

The user ID under which the alias transaction runs must have authority to perform these tasks:

- Attach the alias transaction, if transaction-attach security is specified for the CICS region. Transaction-attach security is controlled by the system initialization parameter XTRAN. The default is YES (transaction-attach security is active).
- Access any CICS resources used by the alias transaction, if resource security is specified for the alias transaction. Resource security is controlled by the RESSEC attribute in the TRANSACTION resource definition for the alias transaction. The default is NO (no resource security), and NO is also the supplied setting for CWBA.
- Access any CICS system programming commands used by the alias transaction, if command security is specified for the alias transaction. These system programming commands are used in the user-written application program that produces the response. Command security is controlled by the CMDSEC attribute in the TRANSACTION resource definition for the alias transaction. The default is NO (no command security), and NO is also the supplied setting for CWBA.

When a web client makes a request to CICS web support, and the response is provided by an application, CICS selects a user ID 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. If authentication is required for the connection but the client does not provide an authenticated user ID, the request is rejected.
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.

Depending on your CICS web support architecture, you might be using one or several of these types of user ID, for different requests. If you obtain an authenticated user ID for a web client, it is used for the alias transaction unless you take action to override it.

Take the following security actions for application-generated responses:

Procedure

1. If you are obtaining authenticated user IDs for web clients, but you do *not* want to use these for security checking for your application-generated responses, you use an analyzer program to override web client user IDs with a standard user ID for the relevant alias transactions. (You can use the CICS default user ID.) Place the analyzer program in the processing paths for the requests where you want to supply this override.

   See Analyzer programs in Developing system programs. Make sure that this user ID has a user profile defined in your security manager.

   When you have set up a standard user ID, you can give the required permissions, as described in the remaining steps of this procedure, to the standard user ID.
2. If you are not obtaining authenticated user IDs for web clients, select suitable user IDs to be standard user IDs for your alias transactions. Unless you just want to use the CICS default user ID, specify your chosen user IDs in the URIMAP definitions for the requests, or set up an analyzer program to specify them.

   Make sure that the standard user IDs have user profiles defined in your security manager.
3. Assuming that transaction-attach security is specified for the CICS region, ensure that all the possible user IDs for your alias transactions have authority to attach the transaction.

All user IDs might include web client user IDs, if you are obtaining them and are not overriding them, or a standard user ID that you have specified in a URIMAP definition or analyzer program, or just the CICS default user ID. See Transaction security.

4. Optional: Apply resource-level security checking for the resources used by an alias transaction:
   a) Identify all the CICS resources used by the alias transaction, and determine which of them are subject to resource security checking in your CICS region.

      Here are some resources that might be used by an application program for CICS web support and the system initialization parameters that control resource security checking for them:

      - CICS document templates (XDOC system initialization parameter)
      - Other application programs invoked by the main application program to perform business logic (XPPT system initialization parameter)
      - Temporary storage queues used to share application state across an HTTP request sequence (XTST system initialization parameter)
      - Files managed by CICS file control (XFCT system initialization parameter)

      Resource security checking for zFS files (XHFS system initialization parameter) does not apply when zFS files are used by an application program, because the files can be manipulated by an application program only when they are defined as CICS document templates, and CICS document template security controls access to them in this situation.

      If you are using an analyzer program, it is the main program for the alias transaction, and so is not subject to resource security checking (only to the transaction-attach security checking). However, note that the user-written web application program itself, and any converter program that you use, is subject to separate resource security checking. Similarly, if you are using a converter program but no analyzer program, the converter program is the main program for the alias transaction, but the application programs called by the converter program are subject to separate resource security checking.

   b) Give all the user IDs that are permitted to attach the alias transaction permission to use the secured resources used by the alias transaction.
   c) Specify RESSEC(YES) in the TRANSACTION resource definition for the transaction.
5. Optional: To apply command security checking for any CICS system programming commands used by an alias transaction:
   a) Confirm that command security is active in the CICS region. Command security is activated by the XCMD system initialization parameter.
   b) Identify the CICS system programming commands used by the application program or programs, analyzer program (if used), and converter program (if used), that are associated with the transaction.

      CICS resources subject to command security checking has a checklist of commands.
   c) Give all the user IDs that are permitted to attach the alias transaction permission to use the commands used by the alias transaction.
   d) Specify CMDSEC(YES) in the TRANSACTION resource definition for the alias transaction.

What to do next

For any security checking to take place in a CICS region, set the SEC=YES system initialization parameter.

Resource-level security for static responses using document templates

If you have implemented basic authentication or client certificate authentication, and you also want to control users' access to specific web pages, you can use web client authenticated user IDs to control access to individual CICS document templates that you are using to provide static responses.

About this task

For static responses delivered by CICS web support using a CICS document template specified in a URIMAP definition, resource security checking is enabled by default.

The **XRES** system initialization parameter controls resource security for CICS document templates. The default for this parameter is YES, meaning that resource security is active. If you do not want to use resource security checking for CICS document templates used for any purpose in your CICS region, you can deactivate it by setting this system initialization parameter to NO.

The transaction for all static responses is the default web listener transaction CWXN, or any alternative transaction that you have specified in place of CWXN using the TRANSACTION attribute on your TCPIPSERVICE definitions. For CICS document templates, you can also control resource security checking by the RESSEC attribute in the TRANSACTION resource definition. For CWXN, as supplied by CICS, RESSEC(YES) is specified, meaning that resource security is active. If you do not want to use resource security checking for static responses, the best way to deactivate it is to replace CWXN in your TCPIPSERVICE definitions with an alternative transaction that specifies the program DFHWBXN and has RESSEC(NO). This setting deactivates resource security checking for CICS document templates for static responses only. Note that the RESSEC attribute cannot control security checking for z/OS UNIX files specified by the HFSFILE attribute.

You can retrieve document templates 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 set up resource-level security for static responses using CICS document templates:

Procedure

1. Identify the authenticated user IDs used by web clients. These IDs must be the basis of your resource security checking. (You cannot supply an override using an analyzer program, as you can with application-generated responses.)

   Authenticated user IDs already have a user profile defined in your security manager.
2. Identify all the CICS document templates that you are using to provide static responses.
3. Implement security for CICS document templates in your CICS region, following the instructions in Security using the XRES resource security parameter.

   You must define a profile to your security manager for each CICS document template that you are using to provide a static response, and give permissions to access appropriate CICS document templates to each authenticated user ID.
4. Ensure that RESSEC(YES) is specified in the TRANSACTION resource definition of CWXN or in the alternative transaction that you have specified in place of CWXN.

   RESSEC(YES) is specified in CWXN as supplied by CICS, but, for TRANSACTION resource definitions in general, the default is RESSEC(NO).

   This step activates resource security checking for your static responses, so ensure that, whenever a web client supplies a user ID, you have set up the appropriate permissions.

What to do next

For any security checking to take place in a CICS region, set the SEC system initialization parameter to YES.

SSL with CICS web support

You can use the Secure Sockets Layer (SSL) with HTTP to enable encryption, message authentication, and client and server authentication using certificates. When you have configured CICS to use SSL, its facilities are available for both CICS as an HTTP server, and CICS as an HTTP client.

Support for security protocols explains the facilities that SSL provides and Configuring CICS to use SSL tells you how to make SSL work with CICS.

When CICS is an HTTP server, you can use SSL to protect an interaction with a web client. You specify appropriate security options on the TCPIPSERVICE definition for the port on which CICS receives the client requests.

As well as specifying the use of SSL, you can require basic authentication or require a client certificate. To give more assistance to web clients, you can allow a client to provide a client certificate, and then register itself to the security manager to supply identification for the CICS environment. You can also allow a client to use self-registration or basic authentication as needed to supply identification. CICS handles all these activities, so, if you are providing an application-generated response, your application does not have to handle this registration. Refer to Creating TCPIPSERVICE resource definitions for CICS web support.

When CICS is an HTTP client, a server might require the use of SSL for some connections. If that is the case, you need to perform some or all of these actions:

- Use HTTPS as the scheme for the connection.
- Supply a list of cipher suites that you want to use for the connection. You can specify them in the URIMAP definition that you use on the WEB OPEN command for the connection.
- Supply a client certificate. Client certificates are not a requirement for all SSL transactions, but a server might require one for particular transactions. If a server does request a client certificate, you can specify the label of a suitable certificate in the URIMAP definition that you use on the WEB OPEN command for the connection, or on the WEB OPEN command itself. The client certificate must be stored in your security manager key ring. If you use a URIMAP definition but do not specify a certificate label, the default certificate defined in the key ring for the CICS region user ID is used.

Introduction to Application Transparent Transport Layer Security (AT-TLS)

Application Transparent Transport Layer Security (AT-TLS) can be used to create secure socket sessions on behalf of CICS. Instead of implementing Transport Layer Security (TLS) in CICS, AT-TLS provides encryption and decryption of data based on policy statements that are coded in the Policy Agent. When AT-TLS is used to secure socket sessions, CICS SSL/TLS start parameters such as KEYRING and MINTLSLEVEL/ENCRYPTION are no longer required as the implementation of TLS is provided by AT-TLS policy statements and all encryption and decryption is done outside of the CICS address space.

AT-TLS MODES

When AT-TLS is active for a CICS socket connection, CICS sends and receives cleartext (unencrypted data), while AT-TLS encrypts and decrypts data at the TCP transport layer. For more information about AT-TLS and AT-TLS policy setup, see AT-TLS policy configuration in z/OS Communications Server: IP Configuration Guide and Policy Agent and policy applications in z/OS Communications Server: IP Configuration Reference.

Most address spaces (such as CICS) do not need to be aware of the security negotiations and encryption that is done by TCP/IP on its behalf. However, some address spaces need to be aware of AT-TLS or have control over the security functions that are being performed by TCP/IP. For example, if the address space is a server that requires client authentication, it might want to access the client certificate and the user ID associated with the client certificate. CICS issues an AT-TLS query on a new client connection when a TCPIPSERVICE is defined with SSL(ATTLSAWARE).

Address spaces such as CICS that are taking advantage of AT-TLS can be separated into three different types (AT-TLS Basic mode, AT-TLS Aware mode, and AT-TLS Controlling mode) as described in Table 50

on page 393. The type is based on whether awareness of the service is needed and, if so, the amount of control that the address space is given over the security functions.

AT-TLS Basic mode is where the address space (such as CICS) does not issue any AT-TLS calls to query a socket for its AT-TLS status.

AT-TLS Aware mode is where the address space issues an AT-TLS calls to query a socket for its AT-TLS status. The address space can access items such as the client certificate, and also the certificate User ID.

AT-TLS Controlling mode is where the address space issues AT-TLS calls to control the secure session on a socket.

Table 50. Detailed description of AT-TLS modes and their CICS support

Mode type	**AT-TLS calls issued**	**`ApplicationControlled` setting in AT-TLS policy**	**Supported by CICS TS for z/OS**
AT-TLS Basic	Address space does not issue any AT-TLS calls	Off	All CICS releases
AT-TLS Aware	Address space issues query requests	Off	From CICS TS 5.3 For CICS as an HTTP server only
AT-TLS Controlling	Address space issues query and control requests	On	Not supported by CICS TS

For detailed information about AT-TLS modes (types), see Application Transparent Transport Layer Security data protection in z/OS Communications Server: IP Configuration Guide.

CICS support for AT-TLS Basic

In the basic mode, the address space is unaware that AT-TLS is encrypting or decrypting data.

With basic mode, the address space (such as CICS) is oblivious to the fact that TCP/IP is performing a TLS handshake and encryption/decryption of message flows on the socket.

With the basic mode of AT-TLS, a CICS TCPIPSERVICE would be defined with SSL(NO). One drawback of this is that CICS cannot access client certificates as it is unaware of a certificate that is associated with the socket. Also, when CICS uses HTTP redirection with this mode of operation, there are failures because CICS assumes that the client connection is HTTP rather than HTTPS. A problem that might arise (for example) is when you are using AUTHENTICATE(BASIC) on an HTTP TCPIPSERVICE, and CICS discovers that the user's password expired. A dialog is triggered with the user to request a new password. At the end of this dialog, there is an error because the resubmission of the original HTTP request specifies HTTP as the scheme instead of HTTPS.

CICS support for AT-TLS Aware

The only support of AT-TLS aware mode is for CICS as an HTTP server.

CICS supports the AT-TLS Aware mode with the additional option of SSL(ATTLSAWARE) on the TCPIPSERVICE definition. SSL(ATTLSAWARE) is only allowed if the TCPIPSERVICE also specifies PROTOCOL(HTTP).

When a TCPIPSERVICE is defined with SSL(ATTLSAWARE), CICS issues an AT-TLS query to obtain information such as AT-TLS security status, negotiated CIPHER suite, partner certificate, and derived RACF user ID.

When an HTTP TCPIPSERVICE is defined with SSL(ATTLSAWARE), CICS can access client certificates and their associated RACF USERIDs. The result is all the certificate-related TCPIPSERVICE AUTHENTICATE options (CERTIFICATE | AUTOREGISTER | AUTOMATIC) are supported by SSL(ATTLSAWARE).

When a TCPIPSERVICE is defined with SSL(ATTLSAWARE), CICS detects that the client is using an HTTPS connection. This means that redirection failures which can occur when using AT-TLS in basic mode (such as the expired password dialog) are fixed by using SSL(ATTLSAWARE).

AT-TLS Controlling

CICS does not support AT-TLS Controlling mode.

CICS AT-TLS query

All new client connections to the TCPIPSERVICE defined with SSL(ATTLSAWARE) are queried to extract their AT-TLS attributes. When a client establishes a new connection to a CICS TCPIPSERVICE defined with SSL(ATTLSAWARE), it will trigger the query and CICS accepts the new client connection.

- Connection status (AT-TLS secured/not secured).
- Client Authentication type (NONE | PASSTHRU | FULL | REQUIRED | SAFCHECK).
- Client certificate. If present, the client certificate is saved in the CICS certificate repository. Attributes of the certificate can be retrieved later on by applications that issue **EXEC CICS EXTRACT CERTIFICATE**.
- Client certificate USERID can be used with the AUTHENTICATE option of the TCPIPSERVICE to establish the security context for the new web task. For example, AUTHENTICATE(CERTIFICATE) requires a CERTIFICATE USERID to be present to allow a web request to be processed.
- Negotiated CIPHER number. This number is registered in a performance monitoring record in the existing field **SOCIPHER**.

Configurations for AT-TLS and CICS TCPIPService

There are certain combinations of AT-TLS policy and CICS TCPIPService, which are valid. The table here shows these valid combinations and expected results in CICS. It also shows combinations that are invalid.

Table 51. Combinations for AT-TLS and CICS TCPIPService

AT-TLS policy for the port	CICS TCPIPSERVICE	Valid combination	Expected result in CICS
`HandShakeRole=Server`	SSL(NO | ATTLSAWARE)	Yes	The connection is successfully established. No client certificate is available.

`HandShakeRole=Server`
or
`HandShakeRole=ServerWithClientAuth`
	SSL(YES | ClientAuth)	No	CICS issues DFHWB0732 and rejects the connection.

`HandShakeRole=ServerWithClientAuth` with `ClientAuthType=(REQUIRED | SAFCHECK)`	SSL(ATTLSAWARE)	Yes	The connection is successfully established. Client certificate is available.

Table 51. Combinations for AT-TLS and CICS TCPIPService (continued)			
AT-TLS policy for the port	**CICS TCPIPSERVICE**	**Valid combination**	**Expected result in CICS**
`HandShakeRole=ServerWithClientAuth` with `ClientAuthType=(FULL)`	SSL(ATTLSAWARE)	Yes	The connection is successfully established. Client certificate is available if client sends it to the server.
`HandShakeRole=ServerWithClientAuth` with any of `ClientAuthType`	SSL(NO)	Yes	The connection is successfully established. No client certificate is available.
NO AT-TLS policy for the port, or the client is exempted from using an AT-TLS policy.	SSL(ATTLSAWARE)	No	CICS rejects the connection with an HTTP 403 error. CICS issues DFHSO0147 for the first connection and DFHWB0365 for every non-secure connection.
`HandShakeRole=ServerWithClientAuth` and `ClientAuthType=PASSTHRU`	SSL(ATTLSAWARE)	No	CICS rejects the connection because this configuration bypasses client certificate validation. CICS issues DFHSO0149 and TCPIPService is closed.
NO AT-TLS policy configured.	SSL(YES | ClientAuth)	Yes	CICS processes SSL-connection as documented.

Note: For TCPIPService, configuring AT-TLS `HandShakeRole=Client` is incorrect.

When you are using a TCPIPSERVICE that specifies SSL(ATTLSAWARE), CICS expects all connections to be cryptographically secured by AT-TLS. If a client connection arrives which is not secured, it is rejected with an HTTP 403 error. CICS also logs the error with message DFHWB0365.

CICS does not support AT-TLS policies that use `ClientAuthType=PassThru`. This configuration bypasses client certificate validation, which is not acceptable to CICS. If CICS detects that this type of client authentication is being used when you are receiving a client connection, it closes the connection with the client then closes the TCPIPSERVICE. Message DFHSO0149 is written to the console when this error is detected.

How to refresh the SSL environment and cache when AT-TLS is in use

When you are using a TCPIPSERVICE that specifies SSL(ATTLSAWARE), the **PERFORM SSL REBUILD** command does not apply.

So to refresh the SSL environment and the certificate CICS is using, follow these steps:

1. Place the new certificate into the key ring defined in your AT-TLS policy.

2. Refresh the key ring within the security manager.
3. Change or add an **EnvironmentUserInstance** value in the policy rule for this CICS traffic.
4. Issue either of the following modify commands:

   ```
   F PAGENT,REFRESH
   or
   F PAGENT,UPDATE
   ```

If you omit step 3 or do not make other changes to the AT-TLS configuration, then the UPDATE option has no effect. UPDATE only refreshes the environment if the AT-TLS configuration has changed. If a certificate in the keyring is the only change that is made, then the REFRESH option can be used.

It is worth noting that a recycle of the CICS region will pick up an AT-TLS certificate change.

AT-TLS diagnostics

There are a number of tools for diagnosing AT-TLS problems.

For diagnosing AT-TLS problems, see Diagnosing Application Transparent Transport Layer Security (AT-TLS) in z/OS Communications Server: IP Diagnosis Guide.

AT-TLS messages contain return codes that are useful in diagnosing problems. Return codes below 5000 come from system SSL. For more information on return codes, see SSL function return codes in z/OS Cryptographic Services System SSL Programming.

Socket Domain Trace Points for AT-TLS

The socket domain trace points listed, are relevant to AT-TLS. For more information, see, Socket domain trace points.

- SO 0CAC (level-1)
- SO 0CAB (EXC)
- SO 0CA9 (level-2)
- SO 0CAA (level-2)

Diagnostic Messages

The messages DFHSO0147 and DFHSO0149 are relevant to AT-TLS and detailed here, DFHSO messages. Also, message DFHWB0365 is detailed here, DFHWB messages.

CICS provides some diagnostic messages when you encounter errors that are discovered by CICS:

DFHWB0365

date time applid tranid A client connects to a TCPIPSERVICE defined with SSL(ATTLSAWARE) but the connection is not secured by AT-TLS. Host IP address: *hostaddr*. Client IP address: *clientaddr*. TCPIPSERVICE: *tcpipservice*.

DFHSO0147 W

applid A non-secure client connection is received for ATTLSAWARE TCPIPSERVICE *tcpipservice*. Client IP address: *clientaddr*. TTLS_IOCTL *value X'ttlsioctl'*.

DFHSO0149 W

applid A client connection that uses **CLIENTAUTHTYPE(PASSTHRU)** is detected for ATTLSAWARE TCPIPSERVICE *tcpipservice*. TTLS_IOCTL *value X'ttlsioctl'*. The TCPIPSERVICE is closed.

Here are two examples of errors on AT-TLS connections:

1. The following diagnostics are seen when a client connects to an AT-TLS secured port, which is configured by using *HandShakeRole ServerWithClientAuth* and *ClientAuthType Required*. This

configuration requires the client to provide a certificate. In this case, the client fails to provide a certificate. Here is the information that is shown in the AT-TLS message log:

```
EZD1287I TTLS Error RC:  403 Initial Handshake  034
LOCAL: ::FFFF:9.20.5.0..25931
REMOTE: ::FFFF:9.174.17.124..50077
JOBNAME: SSYCZCCM RULE: CICSD
USERID: HORN GRPID: 0000000D ENVID: 00000013 CONNID: 00395D99
```

The return code of 403 is a system SSL error, and corresponds to error GSK_ERR_NO_CERTIFICATE, which means `no certificate received from partner`. Nothing is seen in the CICS log. CICS never receives this connection as it is being rejected by AT-TLS.

2. The following diagnostics appear when a client connection is made to a TCPIPSERVICE defined with **SSL(ATTLSAWARE)** where the TCPIPSERVICE port is NOT secured by AT-TLS. This time the client is connecting to a port, which is not policed by AT-TLS and means that there are no AT-TLS diagnostics. However, CICS detects that the client connection is not secured by AT-TLS so it issues the following messages:

   - DFHSO0147 W IY2CZCCM 041 A non-secure client connection is received for ATTLSAWARE TCPIPSERVICE ATTLS2. Client IP address: 9.174.17.124. TTLS_IOCTL value X'0100000102010000'
   - DFHWB0365 06/23/2015 10:14:22 IY2CZCCM CWXN A client connects to a TCPIPSERVICE defined with SSL(ATTLSAWARE) but the connection is not secured by AT-TLS. Host IP address: 9.20.5.0. Client IP address: 9.174.17.124. TCPIPSERVICE: ATTLS2.

   The first message is only issued once for any TCPIPSERVICE. The second message is issued every time a client connects and CICS finds that the connection is NOT secured by AT-TLS.

Migration from CICS SSL to AT-TLS

You can move an existing CICS Transport Layer Security (TLS) (SSL) implementation for an inbound socket connection to Application Transparent Transport Layer Security (AT-TLS).

When CICS is used to establish a TLS (SSL) environment to perform a TLS handshake for an inbound socket connection, the attributes that are used for the handshake are extracted from two sources: region level SIT parameters and TCPIPSERVICE resource attributes.

The following two tables show the CICS SIT parameters and TCPIPSERVICE resource attributes that are used for a TLS handshake and their AT-TLS level equivalents.

Table 52. SIT parameters and AT-TLS equivalents

SIT Parameter	**AT-TLS equivalents**
MINTLSLEVEL	TLSv1 TLSv1.1 TLSv1.2
ENCRYPTION (deprecated: use **MINTLSLEVEL**)	TLSV1 TLSV1.1 TLSV1.2
KEYRING	TTLSKeyRingParms
CRLPROFILE	TTLSGskLdapParms
SSLDELAY	GSK_V3_SESSION_TIMEOUT
MAXSSLTCBS	Cannot be configured in AT-TLS; TCB numbers grow dynamically.
SSLCACHE=SYSPLEX	GSK_SYSPLEX_SIDCACHE ON

Table 52. SIT parameters and AT-TLS equivalents (continued)	
SIT Parameter	**AT-TLS equivalents**
NISTSP800131A=CHECK	FIPS140 ON

Table 53. TCPIPSERVICE resource attributes and AT-TLS equivalents	
TCPIPSERVICE resource attribute	**AT-TLS equivalents**
SSL=YES	HandShakeRole Server
SSL=CLIENTAUTH	HandShakeRole ServerWithClientAuth with ClientAuthType FULL ClientAuthType REQUIRED and ClientAuthType SAFCHECK are also supported.
CERTIFICATE	CertificateLabel
CIPHERS	TTLSCipherParms

Considerations for using keyrings

The CICS region user ID still requires access to the keyring that is specified in the AT-TLS policy. If you are migrating from CICS SSL to AT-TLS, you can continue to use the existing CICS-owned keyrings and reference them in the AT-TLS policies. If you want to set up new keyrings in TCPIP, the CICS region user ID will require access to this new keyring. The server certificate will remain as either a CICS-owned or SITE certificate.

Examples

The following examples show how to move an existing CICS TLS implementation to AT-TLS, and then remove the CICS TLS implementation.

- "Example 1: AT-TLS policy rules for TLS/SSL server authentication" on page 398
- "Example 2: AT-TLS policy rules for TLS/SSL client authentication" on page 401

Example 1: AT-TLS policy rules for TLS/SSL server authentication

An example configuration to use CICS to secure inbound HTTP connections might use simple server authentication on the TCPIPSERVICE resource (SSL(YES)). This configuration does not support client certificates. Figure 33 on page 398 and Figure 34 on page 399 show the CICS configuration statements that are needed to establish the CICS-TLS environment for simple server authentication.

```
MINTLSLEVEL=TLS10  (or its deprecated equivalent ENCRYPTION=STRONG)
KEYRING=CICSKeyRing (includes the certificate named CICS-2048-certificate)
SSLDELAY=600
MAXSSLTCBS=8
SSLCACHE=CICS
NISTSP800131A=NOCHECK
```

Figure 33. CICS startup parameters

```
TCpipservice   : HTTPSSL
GROup          : JULESWEB
DEScription  ==> CICS WEB TCPIPSERVICE WITH SSL SUPPORT
POrtnumber   ==> 25008
STatus       ==> Open
PROtocol     ==> Http
SSl          ==> Yes
CErtificate  ==> CICS-2048-certificate
CIphers      ==> 35363738392F303132330A1613100D15120F0C
AUthenticate ==> Basic
```

Figure 34. SSL-related TCPIPSERVICE resource attributes

To use AT-TLS to secure your inbound HTTP connections instead of CICS, you might use the following AT-TLS policy, and then update the TCPIPSERVICE resource definition to SSL(NO) or SSL(ATTLSAWARE).

Note: The following example AT-TLS policy uses the TLSV1.2 option. Using the TLSV1.2 option helps to achieve optimum performance; also, it is a prerequisite if you need to conform to NIST SP800-131A.

Full details of the NIST standards are available at the NIST Computer Security Resource Center (nist.gov).

Figure 35 on page 400 shows the AT-TLS configuration that replicates the CICS environment for the TCPIPSERVICE named HTTPSSL.

```
TTLSRule SIMPLECICS
{
LocalPortRange 25008
Direction Inbound
Priority 256
TTLSGroupActionRef CICSGroupAct1
TTLSEnvironmentActionRef CICSEnvironmentAct1
}
TTLSGroupAction CICSGroupAct1
{
TTLSEnabled On
FIPS140 off
}
TTLSEnvironmentAction CICSEnvironmentAct1
{
HandShakeRole Server
TTLSKeyRingParmsRef CICSKeyRingParms1
TTLSCipherParmsRef CICSCipherParms1
TTLSEnvironmentAdvancedParmsRef CICSEnvAdvParms1
TTLSGskAdvancedParmsRef CICSGskAdvParms1
}
TTLSKeyRingParms CICSKeyRingParms1
{
Keyring CICSKeyRing
}
TTLSCipherParms CICSCipherParms1
{
V3CipherSuites TLS_RSA_WITH_AES_256_CBC_SHA
V3CipherSuites TLS_DH_DSS_WITH_AES_256_CBC_SHA
V3CipherSuites TLS_DH_RSA_WITH_AES_256_CBC_SHA
V3CipherSuites TLS_DHE_DSS_WITH_AES_256_CBC_SHA
V3CipherSuites TLS_DHE_RSA_WITH_AES_256_CBC_SHA
V3CipherSuites TLS_RSA_WITH_AES_128_CBC_SHA
V3CipherSuites TLS_DH_DSS_WITH_AES_128_CBC_SHA
V3CipherSuites TLS_DH_RSA_WITH_AES_128_CBC_SHA
V3CipherSuites TLS_DHE_DSS_WITH_AES_128_CBC_SHA
V3CipherSuites TLS_DHE_RSA_WITH_AES_128_CBC_SHA
V3CipherSuites TLS_RSA_WITH_3DES_EDE_CBC_SHA
V3CipherSuites TLS_DHE_RSA_WITH_3DES_EDE_CBC_SHA
V3CipherSuites TLS_DHE_DSS_WITH_3DES_EDE_CBC_SHA
V3CipherSuites TLS_DH_RSA_WITH_3DES_EDE_CBC_SHA
V3CipherSuites TLS_DH_DSS_WITH_3DES_EDE_CBC_SHA
V3CipherSuites TLS_DHE_RSA_WITH_DES_CBC_SHA
V3CipherSuites TLS_DHE_DSS_WITH_DES_CBC_SHA
V3CipherSuites TLS_DH_RSA_WITH_DES_CBC_SHA
V3CipherSuites TLS_DH_DSS_WITH_DES_CBC_SHA
}
TTLSEnvironmentAdvancedParms CICSEnvAdvParms1
{
SSLv3 Off
TLSV1 On
TLSV1.1 On
TLSV1.2 On
CertificateLabel CICS-2048-certificate
}
TTLSGskAdvancedParms CICSGskAdvParms1
{
GSK_SYSPLEX_SIDCACHE off
GSK_V3_SESSION_TIMEOUT 600

}
```

Figure 35. AT-TLS configuration

Before you activate this AT-TLS policy, alter the CICS TCPIPSERVICE resource as follows:

```
TCpipservice   : HTTPSSL
GROup          : JULESWEB
DEScription  ==> CICS WEB TCPIPSERVICE WITH AT-TLS SSL SUPPORT
POrtnumber   ==> 25008
STatus       ==> Open
PROtocol     ==> Http
SSl          ==> NO|ATTLSAWARE
CErtificate  ==>
CIphers      ==>
AUthenticate ==> Basic
```

If SSL is set to NO, CICS does not check whether AT-TLS is securing inbound client connections.

If SSL is set to ATTLSAWARE, CICS checks whether AT-TLS is securing inbound client connections. If a client connection is not secured by AT-TLS, it is rejected with an HTTP 403 error and message DFHWB0365 is written to the CICS log.

Also, if SSL is set to ATTLSAWARE, CICS checks for the presence of a client certificate. The previous example AT-TLS configuration does not support the use of client certificates. Therefore, ensure that the TCPIPSERVICE definition does not specify an AUTHENTICATE option that requires client certificates. The previous example TCPIPSERVICE resource specifies AUTHENTICATE(BASIC), which does not require a client certificate.

When the AT-TLS policy is active and the TCPIPSERVICE resource is redefined to remove the SSL attributes, you can also remove all the related SSL SIT parameters. However, first ensure that nothing else in the CICS region depends on these parameters.

If your CICS-SSL system is started with **NISTSP800131A=CHECK**, CICS sets **MINTLSLEVEL=TLS12** and also sets FIPS140 on. To reflect these settings in the example AT-TLS POLICY configuration, modify it as follows:

```
TTLSGroupAction CICSGroupAct1
{
TTLSEnabled On
FIPS140 on
}

TTLSEnvironmentAdvancedParms CICSEnvAdvParms1
{
SSLv3 Off
TLSV1 Off
TLSV1.1 Off
TLSV1.2 On
CertificateLabel CICS-2048-certificate
}
```

Example 2: AT-TLS policy rules for TLS/SSL client authentication

An example configuration to use CICS to secure inbound HTTP connections might use client authentication on the TCPIPSERVICE resource (SSL(CLIENTAUTH)).

This configuration supports client certificates. Figure 36 on page 401 and Figure 37 on page 401 show the CICS configuration statements that are needed to establish the CICS-TLS environment for client authentication.

```
MINTLSLEVEL=TLS10  (or its deprecated equivalent ENCRYPTION=STRONG)
KEYRING=CICSKeyRing (includes the certificate named CICS-2048-certificate)
SSLDELAY=600
MAXSSLTCBS=8
SSLCACHE=CICS
NISTSP800131A=NOCHECK
```

Figure 36. CICS startup parameters

```
TCpipservice   : CLAUTH
GROup          : JULESWEB
DEScription  ==> CICS Web TCPIPSERVICE with SSL CLIENTAUTH support
POrtnumber   ==> 25009
STatus       ==> Open
PROtocol     ==> Http
SSl          ==> Clientauth
CErtificate  ==> CICS-2048-certificate
CIphers      ==> 35363738392F303132330A1613100D15120F0C
AUthenticate ==> Certificate
```

Figure 37. SSL-related TCPIPSERVICE resource attributes

To use AT-TLS to secure your inbound HTTP connections instead of CICS, you might use the following AT-TLS policy, and then update the TCPIPSERVICE resource definition to use SSL(ATTLSAWARE).

Note: The following example AT-TLS policy uses the TLSV1.2 option. Using the TLSV1.2 option helps to achieve optimum performance; also, it is a prerequisite if you need to conform to NIST SP800-131A.

Full details of the NIST standards are available at the NIST Computer Security Resource Center (nist.gov).

Figure 38 on page 402 shows the AT-TLS client authentication configuration that replicates the CICS environment for the TCPIPSERVICE named CLAUTH.

```
TTLSRule CLIENTAUTHCICS
{
LocalPortRange 25009
Direction Inbound
Priority 256
TTLSGroupActionRef CICSGroupAct2
TTLSEnvironmentActionRef CICSEnvironmentAct2
}
TTLSGroupAction CICSGroupAct2
{
TTLSEnabled On
FIPS140 off
}
TTLSEnvironmentAction CICSEnvironmentAct2
{
HandShakeRole ServerWithClientAuth
TTLSKeyRingParmsRef CICSKeyRingParms2
TTLSCipherParmsRef CICSCipherParms2
TTLSEnvironmentAdvancedParmsRef CICSEnvAdvParms2
TTLSGskAdvancedParmsRef CICSGskAdvParms2
}
TTLSKeyRingParms CICSKeyRingParms2
{
Keyring CICSKeyRing
}
TTLSCipherParms CICSCipherParms2
{
V3CipherSuites TLS_RSA_WITH_AES_256_CBC_SHA
V3CipherSuites TLS_DH_DSS_WITH_AES_256_CBC_SHA
V3CipherSuites TLS_DH_RSA_WITH_AES_256_CBC_SHA
V3CipherSuites TLS_DHE_DSS_WITH_AES_256_CBC_SHA
V3CipherSuites TLS_DHE_RSA_WITH_AES_256_CBC_SHA
V3CipherSuites TLS_RSA_WITH_AES_128_CBC_SHA
V3CipherSuites TLS_DH_DSS_WITH_AES_128_CBC_SHA
V3CipherSuites TLS_DH_RSA_WITH_AES_128_CBC_SHA
V3CipherSuites TLS_DHE_DSS_WITH_AES_128_CBC_SHA
V3CipherSuites TLS_DHE_RSA_WITH_AES_128_CBC_SHA
V3CipherSuites TLS_RSA_WITH_3DES_EDE_CBC_SHA
V3CipherSuites TLS_DHE_RSA_WITH_3DES_EDE_CBC_SHA
V3CipherSuites TLS_DHE_DSS_WITH_3DES_EDE_CBC_SHA
V3CipherSuites TLS_DH_RSA_WITH_3DES_EDE_CBC_SHA
V3CipherSuites TLS_DH_DSS_WITH_3DES_EDE_CBC_SHA
V3CipherSuites TLS_DHE_RSA_WITH_DES_CBC_SHA
V3CipherSuites TLS_DHE_DSS_WITH_DES_CBC_SHA
V3CipherSuites TLS_DH_RSA_WITH_DES_CBC_SHA
V3CipherSuites TLS_DH_DSS_WITH_DES_CBC_SHA
}
TTLSEnvironmentAdvancedParms CICSEnvAdvParms2
{
SSLv3 Off
TLSV1 On
TLSV1.1 On
TLSV1.2 On
CertificateLabel CICS-2048-certificate
ClientAuthType Full
}
TTLSGskAdvancedParms CICSGskAdvParms2
{
GSK_SYSPLEX_SIDCACHE off
GSK_V3_SESSION_TIMEOUT 600
}
```

Figure 38. AT-TLS client authentication configuration

Before you activate this example AT-TLS policy, alter the CICS TCPIPSERVICE resource definition as follows:

```
TCpipservice   : CLAUTH
GROup          : JULESWEB
DEScription  ==> CICS Web TCPIPSERVICE with SSL CLIENTAUTH support
```

```
POrtnumber   ==> 25009
STatus       ==> Open
PROtocol     ==> Http
SSl          ==> ATTLSAWARE
CErtificate  ==>
CIphers      ==>
AUthenticate ==> Certificate
```

In this example, SSL must be set to ATTLSAWARE so that CICS retrieves a client certificate from AT-TLS, because AUTHENTICATE is set to CERTIFICATE (a client certificate is required). If a client connection is not secured by AT-TLS, it is rejected with an HTTP 403 error and message DFHWB0365 is written to the CICS log.

With **SSL(ATTLSAWARE)**, CICS checks for a client certificate. If this check maps to a RACF USERID, CICS runs the web user transaction under this USERID.

The previous example AT-TLS policy is defined with `ClientAuthType Full`. This ClientAuthType replicates the SSL environment and handshake behavior that occurs when CICS uses SSL. However CICS also supports `ClientAuthType Required` and `ClientAuthType SAFCheck`.

CICS does not support the use of `ClientAuthType PassThru`. If a TCPIPSERVICE port is configured by using `ClientAuthType PassThru` and the TCPIPSERVICE resource is defined with SSL(ATTLSAWARE), when the first client connection arrives, CICS detects the unsupported configuration. CICS then closes the TCPIPSERVICE and issues message DFHSO0149.

When the AT-TLS policy is active and the TCPIPSERVICE resource is redefined to remove the SSL attributes, you can also remove all the related SSL SIT parameters. However, first ensure that nothing else in the CICS region depends on these parameters.

If your CICS-SSL system is started with **NISTSP800131A=CHECK**, CICS sets **MINTLSLEVEL=TLS12** and it also sets FIPS140 on. To reflect these settings in the example AT-TLS POLICY configuration, modify it as follows:

```
TTLSGroupAction CICSGroupAct2
{
TTLSEnabled On
FIPS140 on
}

TTLSEnvironmentAdvancedParms CICSEnvAdvParms2
{
SSLv3 Off
TLSV1 Off
TLSV1.1 Off
TLSV1.2 On
CertificateLabel CICS-2048-certificate
ClientAuthType Full
}
```

Security for Atom feeds

CICS web support provides a suitable security protocol and authentication method to control web client access to Atom collections and if required, to Atom feeds. You can use CICS resource and command security to protect the resources that you use to deliver the Atom feed or collection.

RFC 5023 recommends that you use authentication to protect Atom collections. When you make Atom feed data available as an editable collection, a web client can insert new entries, modify existing entries, or delete entries. You must therefore ensure that you verify the identity of web clients and permit only trusted clients to have access to the collection, especially if you have included business data in your collection. Ordinary Atom feeds, which web clients cannot edit, are typically made available to any subscribers without security restrictions, although you might need to restrict access to Atom feeds if they include confidential business data or are intended only for certain users.

RFCs 4287 and 5023 discuss the use of digital signatures and encryption for Atom documents. CICS does not provide support for digital signatures and encryption of Atom documents, but, in compliance with RFC 4287, CICS does not reject an Atom document that contains a signature.

CICS web support has the following security functions that you can use to protect Atom feeds or collections from unauthorized access or updates:

SSL or TLS security protocol
RFC 5023 recommends the use of the Transport Layer Security (TLS) 1.0 as a minimum level of security protocol for collections. For a list of security protocols supported by CICS, see Support for security protocols.

HTTP basic authentication
RFC 5023 recommends the use of HTTP basic authentication as a minimum level of authentication for collections.

Client certificate authentication
Client certificate authentication is a more secure method of authenticating a client, using a client certificate which is issued by a trusted third party (or Certificate Authority), and sent using SSL encryption. SSL authentication explains how this works.

When you set up these functions in CICS web support, you can apply them to an Atom feed or collection using attributes of the TCPIPSERVICE definition for the port where CICS receives web client requests for the Atom feed or collection. For information on setting up SSL support for CICS web support, see Configuring CICS to use SSL.

Chapter 14. Support for securing web services

CICS Transaction Server for z/OS provides support for a number of related technologies that you can use to secure SOAP and JSON messages.

Some of these technologies are available as part of the HTTP protocol, and are equally applicable to both SOAP and JSON. Some use the Web Services Security (WSS): SOAP Message Security 1.0 specification, and are only available for SOAP. For information on the shared TCP/IP and HTTP security options, see Security for TCP/IP clients and Security for CICS web support.

For information about using SAML assertions, see Configuring CICS for SAML.

SOAP web services security

Web Services Security (WSS): SOAP Message Security 1.0 describes the use of *security tokens* and *digital signatures* to protect and authenticate SOAP messages. For more information, see the Web Services Security: SOAP Message Security 1.0 specification. Web Services Security: SOAP Message Security 1.0

Web Services Security protects the *privacy* and *integrity* of SOAP messages by, respectively, protecting messages from unauthorized disclosure and preventing unauthorized and undetected modification. WSS provides this protection by digitally signing and encrypting XML elements in the message. The elements that can be protected are the body or any elements in the body or the header. You can give different levels of protection to different elements in the SOAP message.

The Web Services Trust Language (WS-Trust) specification enhances Web Services Security further by providing a framework for requesting and issuing security tokens, and managing trust relationships between web service requesters and providers. This extension to the authentication of SOAP messages enables web services to validate and exchange security tokens of different types byusing a trusted third party. This third party is called a *Security Token Service* (STS). For more information about the Web Services Trust Language, see the Web Services Trust Language specification.

CICS Transaction Server for z/OS provides support for these specifications by using a CICS-supplied security handler in the pipeline:

- For outbound messages, CICS provides support for digital signing and encryption of the entire SOAP body. CICS can also exchange a username token for a security token of a different type with an STS.
- For inbound messages, CICS supports messages in which the body, or elements of the body and header, are encrypted or digitally signed. CICS can also exchange and validate security tokens with an STS.

CICS also provides a separate Trust client interface so that you can interact with an STS without using the CICS security handler.

Note: Web Services Security is potentially not conformant with SP800-131A. Web Services Security is configured by adding a handler into the pipeline and CICS has no control over the processing in a customer-written handler. If you use digital signatures, you can specify only the algorithms `dsa-sha1` and `rsa-sha1`. These algorithms are not SP800-131A-conformant. The two-key tripledes encryption algorithm, which can be used to encrypt a SOAP body, is also non-conformant.

Enabling CICS for WS-Security processing

Some prerequisites are needed in your CICS region to enable it for the full range of WS-Security processing for SOAP messages. You must install the IBM XML Toolkit for z/OS v1.10 and add 3 libraries to the DFHRPL concatenation.

Procedure

1. Install the free IBM XML Toolkit for z/OS v1.10.

You can download it from the following site: https://www.ibm.com/servers/eserver/zseries/software/xml/. You must install version 1.10. Later versions do not work with Web Services Security support in CICS.

2. Add the following libraries to the DFHRPL concatenation:
   - *hlq*.SIXMLOD1, where *hlq* is the high-level qualifier of the XML Toolkit.
   - *hlq*.SCEERUN, where *hlq* is the high-level qualifier of the Language Environment®.
   - *hlq*.SDFHWSLD, where *hlq* is the high-level qualifier of the CICS installation; for example CICSTS56.

   The first two libraries contain DLLs that are required at run time by the security handler. IXM4C57 is provided by the XML Toolkit and is found in *hlq*.SIXMLOD1; C128N is provided by the Language Environment run time and is found in *hlq*.SCEERUN.

   The *hlq*.SDFHWSLD library enables CICS to find the DFHWSSE1 and DFHWSXXX Web Services Security modules.
3. You might need to increase the value of the **EDSALIM** system initialization parameter.

   The three DLLs that are loaded require approximately 15 MB of EDSA storage.

Results

If you do not have the libraries specified, you see the following message:

```
CEE3501S The module module_name was not found.
```

The *module_name* varies depending on which library is missing.

Planning to secure SOAP web services

You can decide the best way to secure your web services. CICS supports a number of options, including a configurable security message handler and a separate Trust client interface.

About this task

CICS implements Web Services Security (WS-Security or WSS) at a pipeline level, rather than for each web service. Answer the following questions to decide how best to implement security.

Procedure

1. Is the performance of your pipeline processing important?

   The use of WSS to secure your web services incurs a significant performance impact.

   The main advantage of implementing WSS is that, by encrypting part of a SOAP message, you can send the message through a chain of intermediate nodes, all of which might have legitimate reasons to look at the SOAP header to make routing or processing decisions, but are not allowed to view the content of the message. By encrypting only those sections that need to be confidential, you derive the following benefits:
   - You do not incur the overhead of encrypting and decrypting at every node in a chain of intermediate processes.
   - You can route a confidential message over a public network of untrusted nodes, where only the ultimate recipient of the data can understand it.

   As an alternative to using WSS, you can use SSL to encrypt the whole data stream.
2. If you want to use WSS, what level of security do you want?

   The options range from basic authentication, where the message header includes a user name and a password, through to combining digital signatures and encryption in the message. The options that the CICS security handler supports are described in “Options for securing SOAP messages” on page 407.
3. Does the CICS-supplied security handler meet your requirements?

   If you want to perform more advanced security processing, you must write your own custom security handler. This handler must perform the necessary authentication of messages, either directly with

RACF or using a Security Token Service, and handle the processing of digital certificates and encrypted elements. See "Writing a custom security handler" on page 420 for details.

4. Does your pipeline include an MTOM handler?

   If you are planning to enable both the MTOM handler and the security handler in your pipeline configuration file, any MIME Multipart or Related messages are processed in compatibility mode, because the security handler cannot parse the XOP elements in the body of the message. This processing can have a further effect on the performance of the pipeline processing.

Options for securing SOAP messages

CICS supports both the signing and encryption of SOAP messages, so you can select the level of security that is most appropriate for the data that you are sending or receiving in the SOAP message.Signing and encryption of SOAP messages are not supported for provider mode Axis2 web service Java applications or for provider web services that attach to the pipeline using Axis2 MessageContext.

You can choose from these options:

Trusted authentication
In service provider pipelines, CICS can accept a username token in the SOAP message header as trusted. This type of security token typically contains a user name and password, but in this case the password is not required. CICS trusts the provided user name and places it in container DFHWS-USERID, and the message is processed in the pipeline.

In service requester pipelines, CICS can send a username token without the password in the SOAP message header to the service provider.

Basic authentication
In service provider mode, CICS can accept a username token in the SOAP message header for authentication on inbound SOAP messages. This type of security token contains a user name and password. CICS verifies the username token using an external security manager, such as RACF. If successful, the user name is placed in container DFHWS-USERID and the SOAP message is processed in the pipeline. If CICS cannot verify the username token, a SOAP fault message is returned to the service requester.

Username tokens that contain passwords are not supported in service requester mode or on outbound SOAP messages.

HTTP basic authentication
In service provider mode, CICS can accept basic authentication information over an HTTP protocol. The service requester uses a URIMAP definition to specify that credentials (user identification information) can be captured by the global user exit, XWBAUTH. XWBAUTH passes this information to CICS on request and CICS sends the information in an HTTP authorization header to the service provider.

Advanced authentication
In service provider and requester pipelines, you can verify or exchange security tokens with a Security Token Service (STS) for authentication purposes. This authentication enables CICS to accept and send messages that have security tokens in the message header that are not normally supported; for example, Kerberos tokens or SAML assertions.

For an inbound message, you can select to verify or exchange a security token. If the request is to exchange the security token, CICS must receive a username token back from the STS. For an outbound message, you can exchange a username token only for a security token.

Signing with X.509 certificates
In service provider and service requester mode, you can provide an X.509 certificate in the SOAP message header to sign the body of the SOAP message for authentication. This type of security token is known as a *binary security token*. To accept binary security tokens from inbound SOAP messages, the public key associated with the certificate must be imported into an external security manager, such as RACF, and associated with the key ring that is specified in the **KEYRING** system initialization parameter. For outbound SOAP messages, you generate and publish the public key to the intended recipients. The Integrated Cryptographic Service Facility (ICSF) is used to generate public keys.

When you specify the label associated with an X.509 digital certificate, do not use the following characters:

```
< > : ! =
```

You can also include a second X.509 certificate in the header and sign it using the first certificate. With this second certificate, you can run the work in CICS under the user ID associated with the second X.509 certificate. The certificate that you are using to sign the SOAP message must be associated with a trusted user ID, and have surrogate authority to assert that work runs under a different identity, the *asserted identity*, without the trusted user ID having the password associated with that identity.

Encrypting
In service provider and service requester mode, you can encrypt the SOAP message body using a symmetric algorithm such as Triple DES or AES. A symmetric algorithm is where the same key is used to encrypt and decrypt the data. This key is known as a *symmetric key*. It is then included in the message and encrypted using a combination of the public key of the intended recipient and the asymmetric key encryption algorithm RSA 1.5. This encryption provides you with increased security, because the asymmetric algorithm is complex and it is difficult to decrypt the symmetric key. However, you obtain better performance because the majority of the SOAP message is encrypted with the symmetric algorithm, which is faster to decrypt.

For inbound SOAP messages, you can encrypt an element in the SOAP body and then encrypt the SOAP body as a whole. This sort of encryption might be particularly appropriate for an element that contains sensitive data. If CICS receives a SOAP message with two levels of encryption, CICS decrypts both levels automatically. This sort of encryption is not supported for outbound SOAP messages.

CICS does not support inbound SOAP messages that have an encrypted element in the message header only and no encrypted elements in the SOAP body.

Signing and encrypting
In service provider and service requester mode, you can choose to both sign and encrypt a SOAP message. CICS always signs the SOAP message body first and then encrypts it. The advantage of this method is that it gives you both message confidentiality and integrity.

ICRX-based identity propagation
In service provider mode, you can use an unauthenticated ICRX (Extended Identity Context Reference) identity token in the same circumstances that you would use an unauthenticated WS-Security user ID token. An ICRX identity token is a z/OS identifier that maps to a user ID. CICS resolves the ICRX identity token to a user ID and places a copy in the DFHWS-ICRX container. CICS also populates the DFHWS-USERID container. For more information about an ICRX identity token, see Identity propagation and distributed security.

Authentication using a Security Token Service

CICS can interoperate with a Security Token Service (STS), such as Tivoli® Federated Identity Manager, to provide more advanced authentication of web services.

An STS is a web service that acts as a trusted third party to broker trust relationships between a web service requester and a web service provider. In a similar manner to a certificate authority in an SSL handshake, the STS guarantees that the requester and provider can "trust" the credentials that are provided in the message. This trust is represented through the exchange of security tokens. An STS can issue, exchange, and validate these security tokens, and establish trust relationships, allowing web services from different trust domains to communicate successfully. For more details, see the Web Services Trust Language specification.

CICS acts as a Trust client and can send two types of web service request to an STS. The first type of request is to validate the security token in the WS-Security message header; the second type of request is to exchange the security token for a different type. These requests enable CICS to send and receive messages that contain different security tokens from a wide variety of trust domains, such as SAML assertions and Kerberos tokens.

You can either configure the CICS security handler to define how CICS interacts with an STS or write your own message handler to use a separately provided Trust client interface. Whichever method you select, use SSL to secure the connection between CICS and the STS.

How the security handler calls the STS

The CICS security handler uses the information in the pipeline configuration file to send a web service request to the Security Token Service (STS). The type of request that is sent depends on the action that you want the STS to perform.

In a service provider pipeline

In a service provider pipeline, the security handler supports two types of actions, depending on the way you configure the security handler:

- Send a request to the STS to validate the first instance of a security token, or the first security token of a specific type, in the WS-Security header of the inbound message.
- Send a request to the STS to exchange the first instance of a security token, or the first security token of a specific type, in the WS-Security header of the inbound message, for a security token that CICS can understand.

The security handler dynamically creates a pipeline to send the web service request to the STS. This pipeline exists until a response is received from the STS, after which it is deleted. If the request is successful, the STS returns an identity token or the status of the validity of the token. The security handler places the RACF ID that is derived from the token in the DFHWS-USERID container.

If the STS encounters an error, it returns a SOAP fault to the security handler. The security handler then passes a fault back to the web service requester.

In a service requester pipeline

In a service requester pipeline, the security handler can request only to exchange a token with the STS. The pipeline configuration file defines what type of token the STS issues to the security handler.

If the request is successful, the RACF ID is placed in the DFHWS-USERID container and the token is included in the outbound message header. If the STS encounters an error, it returns a SOAP fault to the security handler. The security handler then passes the fault back through the pipeline to the web service requester application.

The security handler can request only one type of action from the STS for the pipeline. It can also exchange only one type of token for an outbound request message, and is limited to handling the first token in the WS-Security message header, either the first instance or the first instance of a specific type. These options cover the most common scenarios for using an STS, but might not offer you the processing that you require for handling inbound and outbound messages.

If you want to provide more specific processing to handle many tokens in the inbound message headers or exchange multiple types of tokens for outbound messages, use the Trust client interface. Using this interface, you can create a custom message handler to send your own web service request to the STS.

The Trust client interface

The Trust client interface enables you to interact with a Security Token Service (STS) directly, rather than using the security handler. In this way, you have the flexibility to provide more advanced processing of tokens than the processing offered by the security handler.

The Trust client interface is an enhancement to the CICS-supplied program DFHPIRT. This program is usually used to start a pipeline when a web service requester application has not been deployed using the CICS web services assistant. But it can also act as the Trust client interface to the STS.

You can invoke the Trust client interface by linking to DFHPIRT from a message handler or header processing program, passing a channel called DFHWSTC-V1 and a set of security containers. Using these containers, you have the flexibility to request either a validate or issue action from the STS, select which token type to exchange, and pass the appropriate token from the message header. DFHPIRT dynamically creates a pipeline, composes a web service request from the security containers, and sends it to the STS.

DFHPIRT waits for the response from the STS and passes this back in the DFHWS-RESTOKEN container to the message handler. If the STS encounters an error, it returns a SOAP fault. DFHPIRT puts the fault in the DFHWS-STSFAULT container and returns to the linking program in the pipeline.

You can use the Trust client interface without enabling the security handler in your service provider and service requester pipelines, or you can use the Trust client interface in addition to the security handler.

Signing of SOAP messages

For inbound messages, CICS supports digital signatures on elements in the SOAP body and on SOAP header blocks. For outbound messages, CICS signs all elements in the SOAP body.

A SOAP message is an XML document, consisting of an <Envelope> element, which contains an optional <Header> element and a mandatory <Body> element.

The WSS: SOAP Message Security specification permits the contents of the <Header> and the <Body> to be signed at the element level. That is, in a given message, individual elements can be signed or not, or can be signed with different signatures or using different algorithms. For example, in a SOAP message used in an online purchasing application, it is appropriate to sign elements that confirm receipt of an order, because these elements might have legal status. However, to avoid the overhead of signing the entire message, other information might safely remain unsigned.

For inbound messages, the security message handler can verify the digital signature on individual elements in the SOAP <Header> and the <Body>:

- Signed elements it encounters in the <Header>.
- Signed elements in the SOAP <Body>. If the handler is configured to expect a signed body, CICS rejects any SOAP message in which the body is not signed and issues a SOAP fault.

For outbound messages, the security message handler can sign the SOAP <Body> only; it does not sign the <Header>. The algorithm and key used to sign the body are specified in the handler configuration information.

Signature algorithms

CICS supports the signature algorithms required by the XML Signature specification. Each algorithm is identified by a universal resource identifier (URI).

Algorithm	URI
Digital Signature Algorithm with Secure Hash Algorithm 1 (DSA with SHA1) Supported on inbound SOAP messages only.	`http://www.w3.org/2000/09/xmldsig#dsa-sha1`
Rivest-Shamir-Adleman algorithm with Secure Hash Algorithm 1 (RSA with SHA1)	`http://www.w3.org/2000/09/xmldsig#rsa-sha1`

Example of a signed SOAP message

This is example shows a SOAP message that has been signed by CICS.

```
<?xml version="1.0" encoding="UTF8"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
<SOAP-ENV:Header>
 <wsse:Security xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
                xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd"
                xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd"
                xmlns:xenc="http://www.w3.org/2001/04/xmlenc#" SOAP-ENV:mustUnderstand="1">
 <wsse:BinarySecurityToken 1
              EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-
security-1.0#Base64Binary"
```

```
              ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0#X509"
              wsu:Id="x509cert00">MIIChDCCAe2gAwIBAgIBADANBgkqhkiG9w0BAQUFADAwMQswCQYDVQQGEwJHQjEMMAoGA1UEChMD
                                  SUJNMRMwEQYDVQQDEwpXaWxsIFlhdGVzMB4XDTA2MDEzMTAwMDAwMFoXDTA3MDEzMTIzNTk1OVow
                                  MDELMAkGA1UEBhMCR0IxDDAKBgNVBAoTA0lCTTETMBEGA1UEAxMKV2lsbCBZYXRlczCBnzANBgkq
                                  hkiG9w0BAQEFAAOBjQAwgYkCgYEArsRj/n+3RN75+jaxuOMBWSHvZCB0egv8qu2UwLWEeiogePsR
                                  6Ku4SuHbBwJtWNr0xBTAAS9lEa70yhVdppxOnJBOCiERg7S0HUdP7a8JXPFzA+BqV63JqRgJyxN6
                                  msfTAvEMR07LIXmZAte62nwcFrvCKNPCFIJ5mkaJ9v1p7jkCAwEAAaOBrTCBqjA/BglghkgBhvhC
                                  AQ0EMhMwR2VuZXJhdGVkIGJ5IHRoZSBTZWN1cml0eSBTZXJ2ZXIgZm9yIHovT1MgKFJBQ0YpMDgG
                                  ZQVRFU0BVSy5JQk0uQ09ggdJQk0uQ09NhgtXV1cuSUJNLkNPTYcECRRlBjAO
 </wsse:BinarySecurityToken>
 <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
  <ds:SignedInfo xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
                 xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
                 xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd"
                 xmlns:xenc="http://www.w3.org/2001/04/xmlenc#">
   <ds:CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
    <c14n:InclusiveNamespaces xmlns:c14n="http://www.w3.org/2001/10/xml-exc-c14n#" PrefixList="ds wsu xenc SOAP-ENV "/>
   </ds:CanonicalizationMethod>
   <ds:SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
   <ds:Reference URI="#TheBody">
    <ds:Transforms>
     <ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
       <c14n:InclusiveNamespaces xmlns:c14n="http://www.w3.org/2001/10/xml-exc-c14n#" PrefixList="wsu SOAP-ENV "/>
     </ds:Transform>
    </ds:Transforms>
    <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/> 2
    <ds:DigestValue>QORZEA+gpafluShspHxhrjaFlXE=</ds:DigestValue> 3
   </ds:Reference>
  </ds:SignedInfo>
  <ds:SignatureValue>drDH0XESiyN6YJm27mfK1ZMG4Q4IsZqQ9N9V6kEnw2lk7aM3if77XNFnyKS4deglbC3ga11kkaFJ 4
                     p4jLOmYRqqycDPpqPm+UEu7mzfHRQGe7H0EnFqZpikNqZK5FF6fvYlv2JgTDPwrOSYXmhzwegUDT
                     lTVjOvuUgXYrFyaO3pw=</ds:SignatureValue>
   <ds:KeyInfo>
    <wsse:SecurityTokenReference>
      <wsse:Reference URI="#x509cert00"
                      ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0#X509"/
> 5
    </wsse:SecurityTokenReference>
   </ds:KeyInfo>
  </ds:Signature>
 </wsse:Security>
</SOAP-ENV:Header>
<SOAP-ENV:Body xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd"
wsu:Id="TheBody">
 <getVersion xmlns="http://msgsec.wssecfvt.ws.ibm.com"/>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

1. The binary security token contains the base64binary encoding of the X.509 certificate. This encoding includes the public key that the intended recipient of the SOAP message uses to verify the signature.
2. The algorithm that is used during the hashing process to produce the message digest.
3. The value of the message digest.
4. The digest value is then encrypted with the user's private key and included here as the signature value.
5. References the binary security token that contains the public key that is used to verify the signature.

CICS support for encrypted SOAP messages

For inbound messages, CICS can decrypt any encrypted elements in the SOAP body, and encrypted SOAP header blocks where the body is also encrypted. For outbound messages, CICS encrypts the entire SOAP body.

A SOAP message is an XML document, consisting of an <Envelope> element, which contains an optional <Header> element, and a mandatory <Body> element.

The WSS: SOAP Message Security specification allows some of the contents of the <Header> element and all of the contents of the <Body> element to be encrypted at the element level. That is, in a given message, individual elements can have different levels of encryption, or can be encrypted using different algorithms. For example, in a SOAP message used in an online purchasing application, it is appropriate to encrypt an individual's credit card details to ensure that they remain confidential. However, to avoid the overhead of encrypting the entire message, some information might safely be encrypted using a less secure (but faster) algorithm and other information might safely remain unencrypted.

For inbound messages, the CICS-supplied security message handler can decrypt individual elements in the SOAP <Body>, and can decrypt elements in the SOAP <Header> if the SOAP body is also encrypted. The security message handler always decrypts these elements:

- Elements it encounters in the <Header> element in the order in which the elements are found.
- Elements in the SOAP <Body> element. If you want to reject a SOAP message that does not have an encrypted <Body>, configure the handler to expect an encrypted body using the <expect_encrypted_body> element.

For outbound messages, the security message handler supports encryption of the contents of the SOAP <Body> only; it does not encrypt any elements in the <Header> element. When the security message handler encrypts the <Body> element, all elements in the body are encrypted with the same algorithm and using the same key. The algorithm, and information about the key, are specified in the configuration information about the handler.

Encryption algorithms

CICS supports the encryption algorithms required by the XML Encryption specification. Each algorithm is identified by a universal resource identifier (URI).

Algorithm	**URI**
Triple Data Encryption Standard algorithm (Triple DES)	http://www.w3.org/2001/04/xmlenc#tripledes-cbc
Advanced Encryption Standard (AES) algorithm with a key length of 128 bits	http://www.w3.org/2001/04/xmlenc#aes128-cbc
Advanced Encryption Standard (AES) algorithm with a key length of 192 bits	http://www.w3.org/2001/04/xmlenc#aes192-cbc
Advanced Encryption Standard (AES) algorithm with a key length of 256 bits	http://www.w3.org/2001/04/xmlenc#aes256-cbc

Example of an encrypted SOAP message

This example of a SOAP message has been encrypted by CICS.

```
<?xml version="1.0" encoding="UTF8"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
<SOAP-ENV:Header>
 <wsse:Security xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
                xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd"
                xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd"
                xmlns:xenc="http://www.w3.org/2001/04/xmlenc#" SOAP-ENV:mustUnderstand="1">

  <wsse:BinarySecurityToken
                EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-
security-1.0#Base64Binary" 1
                ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0#X509"
                wsu:Id="x509cert00">MIIChDCCAe2gAwIBAgIBADANBgkqhkiG9w0BAQUFADAwMQswCQYDVQQGEwJHQjEMMAoGA1UEChMD
                                    SUJNMRMwEQYDVQQDEwpXaWxsIF1hdGVzMB4XDTA2MDEzMTAwMDAwMFoXDTA3MDEzMTIzNTk1OVow
                                    MDELMAkGA1UEBhMCR0IxDDAKBgNVBAoTA0lCTTETMBEGA1UEAxMKV2lsbCBZYXRlczCBnzANBgkq
                                    hkiG9w0BAQEFAAOBjQAwgYkCgYEArsRj/n+3RN75+jaxuOMBWSHvZCB0egv8qu2UwLWEeiogePsR
                                    6Ku4SuHbBwJtWNr0xBTAAS9lEa70yhVdppxOnJBOCiERg7S0HUdP7a8JXPFzA+BqV63JqRgJyxN6
                                    msfTAvEMR07LIXmZAte62nwcFrvCKNPCFIJ5mkaJ9v1p7jkCAwEAAaOBrTCBqjA/BglghkgBhvhC
                                    AQ0EMhMwR2VuZXJhdGVkIGJ5IHRoZSBTZWN1cml0eSBTZXJ2ZXIgZm9yIHovT1MgKFJBQ0YpMDgG
                                    A1UdEQQxMC+BEVdZQVRFU0BVSy5JQk0uQ09NggdJQk0uQ09NhgtXV1cuSUJNLkNPTYcECRRlBjAO
                                    BgNVHQ8BAf8EBAMCAfYwHQYDVR0OBBYEFMiPX6VZKP5+mSOY1TLNQGVvJzu+MA0GCSqGSIb3DQEB
                                    BQUAA4GBAHdrS409Jhoe67pHL2gs7x4SpV/NOuJnn/w25sjjop3RLgJ2bKtK6RiEevhCDim6tnYW
                                    NyjBL1VdN7u5M6kTfd+HutR/HnIrQ3qPkXZK4ipgC0RWDJ+8APLySCxtFL+J0LN9Eo6yjiHL68mq
                                    uZbTH2LvzFMy4PqEbmVKbmA87alF
  </wsse:BinarySecurityToken>
  <xenc:EncryptedKey xmlns:xenc="http://www.w3.org/2001/04/xmlenc#">
   <xenc:EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-1_5"/> 2
   <ds:KeyInfo xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
    <wsse:SecurityTokenReference>
     <wsse:Reference URI="#x509cert00"
                    ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0#X509"/> 
3
    </wsse:SecurityTokenReference>
   </ds:KeyInfo>
   <xenc:CipherData>
    <xenc:CipherValue>M6bDQtJrvX0pEjAEIcf6bq6MP3ySmB4TQOa/B5UlQj1vWjD56V+GRJbF7ZCES5ojwCJHRVKW1ZB5 4
                      Mb+aUzSWlsoHzHQixc1JchgwCiyIn+E2TbG3R9m0zHD3XQsKTyVaOTlR7VPoMBd1ZLNDIomxjZn2
```

```
                      p7JfxywXkObcSLhdZnc=</xenc:CipherValue>
   </xenc:CipherData>
   <xenc:ReferenceList>
    <xenc:DataReference URI="#Enc1"/>
   </xenc:ReferenceList>
  </xenc:EncryptedKey>
 </wsse:Security>
</SOAP-ENV:Header>
<SOAP-ENV:Body>
 <xenc:EncryptedData xmlns:xenc="http://www.w3.org/2001/04/xmlenc#" Id="Enc1" Type="http://www.w3.org/2001/04/
xmlenc#Content">
  <xenc:EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#tripledes-cbc"/> 5
  <xenc:CipherData>
   <xenc:CipherValue>kgvqKnMcgIUn7rl1vkFXF0g4SodEd3dxAJo/mVN6ef211B1MZelg7OyjEHf4ZXwlCdtOFebIdlnK 6
                     rrksql1Mpw6So7ID8zav+KPQUKGm4+E=</xenc:CipherValue>
  </xenc:CipherData>
 </xenc:EncryptedData>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

1. The binary security token contains the base64binary encoding of the X.509 certificate. This encoding includes the public key that was used to encrypt the symmetric key.
2. States the algorithm that was used to encrypt the symmetric key.
3. References the binary security token that contains the public key used to encrypt the symmetric key.
4. The encrypted symmetric key that was used to encrypt the message.
5. The encryption algorithm that was used to encrypt the message.
6. The encrypted message.

Configuring RACF for Web Services Security

You must configure an external security manager, such as RACF, to create public-private key pairs and X.509 certificates for signing and encrypting outbound SOAP messages and to authenticate and decrypt signed and encrypted inbound SOAP messages.

Before you begin

Before you perform this task, you must have RACF set up to work with CICS. Specify the **DFLTUSER**, **KEYRING**, and **SEC=YES** system initialization parameters in the CICS region that contains your web services pipelines.

Note: Multiple certificates with the same Distinguished Name on the same **KEYRING** are not supported.

Procedure

1. To authenticate inbound SOAP messages that are signed:
   a) Import the X.509 certificate into RACF as an ICSF key.
   b) Attach the certificate to the key ring specified in the **KEYRING** system initialization parameter, using the **RACDCERT** command:

      ```
      RACDCERT ID(userid1)
      CONNECT(ID(userid2) LABEL('label-name') RING(ring-name)
      ```

      where:

      - *userid1* is the default user ID of the key ring or has authority to attach certificates to the key ring for other user IDs.
      - *userid2* is the user ID that you want to associate with the certificate.
      - *label-name* is the name of the certificate.
      - *ring-name* is the name of the key ring that is specified in the **KEYRING** system initialization parameter.

   c) Optional: If you want to use asserted identities, ensure that the user ID associated with the certificate has surrogate authority to allow work to run under other user IDs.

Also, make sure that any additional certificates included in the SOAP message header are also imported into RACF.

The SOAP message can contain a binary security token in the header that either includes the certificate or contains a reference to the certificate. This reference can be the KEYNAME (the certificate label in RACF), a combination of the ISSUER and SERIAL number, or the SubjectKeyIdentifier. CICS can recognize the SubjectKeyIdentifier only if it has been specified as an attribute in the definition of the certificate in RACF.

2. To sign outbound SOAP messages:
   a) Create an X.509 certificate and a public-private key pair using the following **RACDCERT** command:

```
RACDCERT ID(userid2) GENCERT
SUBJECTSDN(CN('common-name')
           T('title')
           OU('organizational-unit')
           O('organization')
           L('locality')
           SP('state-or-province')
           C('country'))
WITHLABEL('label-name')
```

   where *userid2* is the user ID that you want to associate with the certificate.

   When you specify the certificate *label-name* value, do not use the following characters:

```
< > : ! =
```

   b) Attach the certificate to the key ring specified in the **KEYRING** system initialization parameter.

   Use the **RACDCERT** command.

   c) Export the certificate and publish it to the intended recipient of the SOAP message.

You can edit the pipeline configuration file so that CICS automatically includes the X.509 certificate in the binary security token of the SOAP message header for the intended recipient to validate the signature.

3. To decrypt inbound SOAP messages that are encrypted, the SOAP message must include the public key that is part of a key pair, where the private key is defined in CICS.
   a) Generate a public-private key pair and certificate in RACF for encryption.

   The key pair and certificate must be generated using ICSF.

   b) Attach the certificate to the key ring specified in the **KEYRING** system initialization parameter. Use the **RACDCERT** command.

   c) Export the certificate and publish it to the generator of the SOAP messages that you want to decrypt.

The generator of the SOAP message can then import the certificate that contains the public key and use it to encrypt the SOAP message. The SOAP message can contain a binary security token in the header that either includes the public key or contains a reference to it. This reference can be the KEYNAME, a combination of the ISSUER and SERIAL number, or the SubjectKeyIdentifier. CICS can recognize the SubjectKeyIdentifier only if it has been specified as an attribute in the definition of the public key in RACF.

4. To encrypt outbound SOAP messages:
   a) Import the certificate that contains the public key that you want to use for encryption into RACF as an ICSF key.

   The intended recipient must have the private key associated with the public key to decrypt the SOAP message.

   b) Attach the certificate that contains the public key to the key ring specified in the **KEYRING** system initialization parameter. Use the **RACDCERT** command.

CICS uses the public key in the certificate to encrypt the SOAP body and sends the certificate containing the public key as a binary security token in the SOAP message header. The public key is defined in the pipeline configuration file.

What to do next

This configuration for signing and encrypting outbound messages requires that the certificate used is owned by the CICS region user ID. The certificate must be owned by the CICS region userid because RACF allows only the certificate owner to extract the private key, which is used for the signing or encryption process.

If CICS needs to sign or encrypt a message using a certificate that it does not own, you can share a single certificate between CICS systems by following the instructions in Using an existing certificate that is not owned by the CICS region user ID.

Configuring provider mode web services for identity propagation

Identity propagation with a web service request relies on trust-based configurations; for example, using a client-certified SSL connection from IBM DataPower. In this task, you configure a PIPELINE resource to expect an ICRX identity token in the WS-Security header, sent from a trusted client.

Before you begin

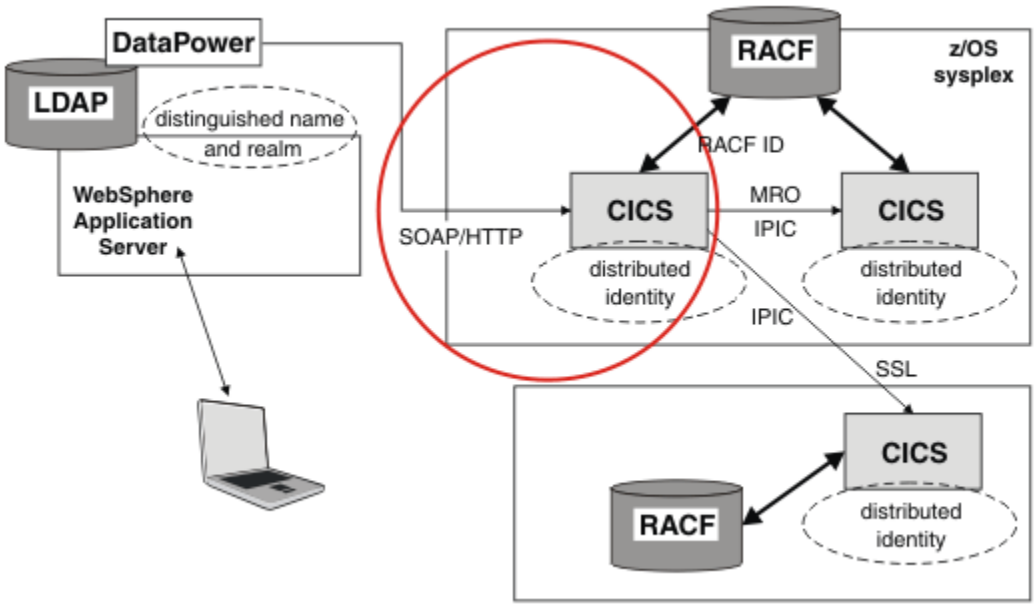
You must configure your RACF RACMAP settings before you configure your web service connections, otherwise you receive the RACF ICH408I message for every unmapped request that is sent to RACF. For more information about configuring the RACF **RACMAP** command, see Configuring RACF for identity propagation.

You must configure a *trust* relationship between the IBM DataPower appliance and CICS, for example, using SSL client certification between IBM DataPower and CICS. The digital certificate that IBM DataPower uses to identify itself must be associated with a user ID, and that user ID must be granted surrogate authority to assert identities. For more information about surrogate authority, see Surrogate user security.

About this task

This task explains how to use CICS with a IBM DataPower appliance to provide a web service configuration that can propagate distributed identities in a secure and robust way. The circle in the diagram indicates that this task explains the CICS-specific configuration.

Figure 39. Configuring CICS to expect an ICRX identity token from IBM DataPower.



IBM DataPower acts as an intermediary between CICS and other applications. Remote web service requester applications connect to the IBM DataPower appliance using the SOAP protocol. IBM DataPower authenticates the credentials supplied by the remote client and mapping the credentials to a z/OS ICRX identity token, which identifies the distributed identity of a user. The SOAP message is then forwarded to CICS over the trusted SSL connection with an ICRX identity token in a WS-Security header. For more information about ICRX identity tokens, see z/OS Security Server RACF Data Areas.

CICS receives the SOAP message from IBM DataPower. The PIPELINE configuration file specifies *blind* trust, because the only possible client is the IBM DataPower appliance, and IBM DataPower is communicating with CICS over a secure SSL connection. Therefore, you do not need to specify additional authentication in the PIPELINE configuration file. The WS-Security handler program locates the first ICRX found in the WS-Security header and uses the ICRX to identify the user.

Procedure

1. Create a PIPELINE resource, or edit an existing PIPELINE resource to specify the basic-ICRX mode, which allows the PIPELINE to receive an ICRX.

   The most typical combination is the blind trust with the basic-ICRX mode. For more information about the PIPELINE resource element, see The <authentication> element.

   Here is an example PIPELINE configuration file, showing blind trust with the basic-ICRX mode:

```
<?xml version="1.0" encoding="EBCDIC-CP-US"?>
<provider_pipeline xmlns="http://www.ibm.com/software/htp/cics/pipeline">
   <service>
     <service_handler_list>
       <wsse_handler>
         <dfhwsse_configuration version="1">
           <authentication trust="blind" mode="basic-ICRX"/>
         </dfhwsse_configuration>
       </wsse_handler>
     </service_handler_list>
     <terminal_handler>
       <cics_soap_1.2_handler/>
     </terminal_handler>
   </service>
   <apphandler>DFHPITP</apphandler>
</provider_pipeline>
```

   Here is an example SOAP message with an ICRX identity, using blind trust:

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope
  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Header>
    <wsse:Security
        xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-
secext-1.0.xsd"
        SOAP-ENV:mustUnderstand="1">

      <wsse:BinarySecurityToken EncodingType="http://docs.oasis-open.org/wss/2004/01/
oasis-200401-wss
-soap-message-security-1.0#Base64Binary"
        wsu:Id="ICRX"
        xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wsswssecurity-
utility-1.0.xsd"
        ValueType="http://www.IBM.com/xmlns/prod/zos/saf#ICRXV1">

        ICRX IS HERE

      </wsse:BinarySecurityToken>

    </wsse:Security>
  </SOAP-ENV:Header>
  <SOAP-ENV:Body>
```

```
   APPLICATION SPECIFIC XML IS HERE

  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

2. Ensure that IBM DataPower is configured to be able to send ICRX information. See Sample network topologies for using identity propagation.

Results

Web service requests from IBM DataPower with an ICRX identity token in the WS-Security header, connected over a client-certified SSL connection, can now flow.

Configuring the pipeline for Web Services Security

To configure a pipeline to support Web Services Security (WSS), you must add a security handler to your pipeline configuration files. You can use the security handler supplied with CICS, as described, or create your own.

Before you begin

Before you define the CICS-supplied security handler, you must identify or create the pipeline configuration files to which you add configuration information for WSS.

Procedure

1. Add a <wsse_handler> element to your pipeline.

   The handler must be included in the <service_handler_list> element in a service provider or requester pipeline.

   Code the following elements:

   ```
   <wsse_handler>
     <dfhwsse_configuration version="1">

     </dfhwsse_configuration>
   </wsse_handler>
   ```

   The <dfhwsse_configuration> element is a container for the other elements in the configuration.

2. Optional: Code an <authentication> element.

   - In a service requester pipeline, the <authentication> element specifies the type of authentication that must be used in the security header of outbound SOAP messages.
   - In a service provider pipeline, the element specifies whether CICS uses the security tokens in an inbound SOAP message to determine the user ID under which work is processed.

   a) Code the **trust** attribute to specify whether asserted identity is used and the nature of the trust relationship between service provider and requester.

   For details of the **trust** attribute, see The <authentication> element.

   b) Optional: If you specified **trust=none**, code the **mode** attribute to specify how credentials found in the message are processed.

   For details of the **mode** attribute, see The <authentication> element.

   c) In the <authentication> element, code these elements:

   i) An optional, empty <suppress/> element.

   If this element is specified in a service provider pipeline, the handler does not attempt to use any security tokens in the message to determine under which user ID the work runs.

   If this element is specified in a service requester pipeline, the handler does not attempt to add to the outbound SOAP message any of the security tokens that are required for authentication.

ii) In a requester pipeline, an optional <algorithm> element that specifies the URI of the algorithm that is used to sign the body of the SOAP message. You must specify this element if the combination of trust and mode attribute values indicate that the messages are signed. You can specify only the RSA with SHA1 algorithm in this element. The URI is http://www.w3.org/2000/09/xmldsig#rsa-sha1.

iii) An optional <certificate_label> element that specifies the label that is associated with an X.509 digital certificate installed in RACF. If you specify this element in a service requester pipeline and the <suppress> element is not specified, the certificate is added to the security header in the SOAP message. If you do not specify a <certificate_label> element, CICS uses the default certificate in the RACF key ring.

This element is ignored in a service provider pipeline.

3. Optional: Code an <sts_authentication> element as an alternative to the <authentication> element.

   You must not code both in your pipeline configuration file. This element specifies that a Security Token Service (STS) is used for authentication and determines the type of request that is sent.

   a) Optional: In service provider mode only, code the **action** attribute to specify whether the STS verifies or exchanges a security token.

      For details of the **action** attribute, see The <sts_authentication> element.

   b) Within the <sts_authentication> element, code these elements:

      i) An <auth_token_type> element. This element is required when you specify a <sts_authentication> element in a service requester pipeline and is optional in a service provider pipeline. For more information, see The <auth_token_type> element.

      - In a service requester pipeline, the <auth_token_type> element indicates the type of token that STS issues when CICS sends it the user ID contained in the DFHWS-USERID container. The token that CICS receives from the STS is placed in the header of the outbound message.
      - In a service provider pipeline, the <auth_token_type> element is used to determine the identity token that CICS takes from the message header and sends to the STS to exchange or validate. CICS uses the first identity token of the specified type in the message header. If you do not specify this element, CICS uses the first identity token that it finds in the message header. CICS does not consider the following as identity tokens:
        - wsu:Timestamp
        - xenc:ReferenceList
        - xenc:EncryptedKey
        - ds:Signature

      ii) In a service provider pipeline only, an optional, empty <suppress/> element. If this element is specified, the handler does not attempt to use any security tokens in the message to determine the user ID that the work runs under. The <suppress/> element includes the identity token that is returned by the STS.

4. Optional: Code an <sts_endpoint> element.

   Use this element only if you have also specified an <sts_authentication> element. In the <sts_endpoint> element, code the following elements:

   - An <endpoint> element. This element contains a URI that points to the location of the Security Token Service (STS) on the network. It is recommended that you use TLS to keep the connection to the STS secure, rather than using HTTP.

     To use SAML support, set the endpoint to cics://PROGRAM/DFHSAML.

     You can also specify an IBM MQ endpoint, by using the JMS format of URI.
   - An optional <jvmserver> element. This element identifies the JVM server that is configured to run the SAML token service. If this element is not included, the default sample resource JVM

server DFHXSTS is assumed. This element is valid only if you are using SAML: if you use it in other situations, an error occurs.

5. Optional: If you require inbound SOAP messages to be digitally signed, code an empty <expect_signed_body/> element.

   The <expect_signed_body/> element indicates that the <body> of the inbound message must be signed. If the body of an inbound message is not correctly signed, CICS rejects the message with a security fault.

6. Optional: If you want to reject inbound SOAP messages that are digitally signed, code an empty <reject_signature/> element.
7. Optional: If you require inbound SOAP messages to be encrypted, code an empty <expect_encrypted_body/> element.

   The <expect_encrypted_body/> element indicates that the <body> of the inbound message must be encrypted. If the body of an inbound message is not correctly encrypted, CICS rejects the message with a security fault.

8. If you want to reject inbound SOAP messages that are partially or fully encrypted, code an empty <reject_encryption/> element.
9. Optional: If you require outbound SOAP messages to be signed, code a <sign_body> element.

   a) In the <sign_body> element, code an <algorithm> element.

   b) Following the <algorithm> element, code a <certificate_label> element.

   Here is an example of a completed <sign_body> element:

```
<sign_body>
  <algorithm>http://www.w3.org/2000/09/xmldsig#rsa-sha1</algorithm>
  <certificate_label>SIGCERT01</certificate_label>
</sign_body>
```

10. Optional: If you require outbound SOAP messages to be encrypted, code an <encrypt_body> element.

    a) In the <encrypt_body> element, code an <algorithm> element.

    b) Following the <algorithm> element, code a <certificate_label> element.

    Here is an example of a completed <encrypt_body> element:

```
<encrypt_body>
  <algorithm>http://www.w3.org/2001/04/xmlenc#tripledes-cbc</algorithm>
  <certificate_label>ENCCERT02</certificate_label>
</encrypt_body>
```

Example

The following example shows a completed security handler in which most of the optional elements are present:

```
<wsse_handler>
    <dfhwsse_configuration version="1">
      <authentication trust="signature" mode="basic">
        <suppress/>
        <certificate_label>AUTHCERT03</certificate_label>
      </authentication>
      <expect_signed_body/>
      <expect_encrypted_body/>
      <sign_body>
        <algorithm>http://www.w3.org/2000/09/xmldsig#rsa-sha1</algorithm>
        <certificate_label>SIGCERT01</certificate_label>
      </sign_body>
      <encrypt_body>
        <algorithm>http://www.w3.org/2001/04/xmlenc#tripledes-cbc</algorithm>
        <certificate_label>ENCCERT02</certificate_label>
      </encrypt_body>
    </dfhwsse_configuration>
</wsse_handler>
```

Writing a custom security handler

If you want to use your own security procedures and processing, you can write a custom message handler to process secure SOAP messages in the pipeline.

You need to decide the level of security that your security handler must support, and ensure that an appropriate SOAP fault is returned when a message includes security that is not supported. The message handler must also be able to cope with security on inbound and outbound messages.

Here is a likely set of steps that your security handler would implement:

1. Retrieve the DFHREQUEST or DFHRESPONSE container using an **EXEC CICS GET CONTAINER** command.
2. Parse the XML to find the security token that is in the WS-Security message header. The header starts with the `<wsse:Security>` element. The security token might be a user name and password, a digital certificate, or an encryption key. A message can have many tokens in the security header, so your handler needs to identify the correct one to process.
3. Perform the appropriate processing, depending on the security that is implemented in the message:
   - To perform basic authentication of a Kerberos token, issue an **EXEC CICS VERIFY TOKEN** command. This command checks that the supplied Kerberos token is valid. If the command is successful, update the DFHWS-USERID container with an **EXEC CICS PUT CONTAINER**. Otherwise, issue an **EXEC CICS SOAPFAULT CREATE** command.
   - To perform basic authentication of a password or password phrase, issue an **EXEC CICS VERIFY PHRASE** command. This command checks the user name and password in the security header of the message. If the command is successful, update the DFHWS-USERID container with an **EXEC CICS PUT CONTAINER**. Otherwise, issue an **EXEC CICS SOAPFAULT CREATE** command.
   - You might also want to write an audit record each time a service is requested, for example, you could write a message to a CICS user journal.
   - To perform advanced authentication, either by exchanging or validating a range of tokens with a Security Token Service, use the Trust client interface which enables you to interact with the STS directly. . See "Invoking the Trust client from a message handler" on page 420 for details.
   - Validate the credentials of the digital certificate if the message is signed.
   - If parts of the message are encrypted, decrypt the message using the information in the security header. The How CICS complies with Web Services Security specifications specification provides information about how to do this
4. Define your security handler program in CICS and update the pipeline configuration file, ensuring that it is correctly placed in the XML. In a service requester pipeline configuration file, the security handler must be configured to run at the end of the pipeline. In a service provider pipeline configuration file, the security handler must be configured to run at the beginning of the pipeline.

For examples of custom message handlers, see IBM Redbooks: Implementing CICS Web services.

Invoking the Trust client from a message handler

CICS provides an interface so that you can write your own message handler to invoke a Security Token Service (STS). With this interface you can perform more advanced processing than the CICS-supplied security handler.

About this task

You can use the Trust client instead of the security handler or in addition to it. To use the Trust client interface:

Procedure

1. Extract the correct token from the security message header of the inbound or outbound message.

2. Link to program DFHPIRT, passing the channel DFHWSTC-V1 and the following required containers:
   - DFHWS-STSURI, containing the location of the STS on the network.
   - DFHWS-STSACTION, containing the URI of the type of request that the STS must perform. The two supported actions are `issue` and `validate`.
   - DFHWS-IDTOKEN, containing the token that must either be verified or exchanged by the STS.
   - DFHWS-TOKENTYPE, containing the type of token that the STS must send back in the response.
   - DFHWS-SERVICEURI, containing the URI of the web service operation that is being invoked.

   You can optionally include the DFHWS-XMLNS container to provide the namespaces of the SOAP message that contains the security token. This container is described in more detail in The header processing program interface.
3. DFHPIRT returns with the response from the STS.

   A successful response is stored in the DFHWS-RESTOKEN container.

   If the STS encounters a problem with the request, it returns a SOAP fault. DFHPIRT puts the SOAP fault in the DFHWS-STSFAULT container. If the STS provides a reason for issuing the SOAP fault, the reason is put in the DFHWS-STSREASON container.

   If an abend occurs, a DFHERROR container is returned that contains details of the processing error.

   Your message handler must handle these responses and perform suitable processing in the event of an error. For example, the message handler might return a suitable SOAP fault to the web service requester.
4. Process the response as appropriate.

   In provider mode, your pipeline processing must ensure that a user name that CICS can understand is placed in the DFHWS-USERID container by the time the message reaches the application handler. In requester mode, your message handler must add the correct token to the outbound message security header.

What to do next

When you have written your message handler, deploy the program in CICS and update the appropriate pipeline configuration files. In service requester pipelines, define your message handler to occur at the end of the pipeline processing but before the CICS-supplied security handler. In service provider pipelines, define your message handler at the beginning of the pipeline but after the CICS-supplied security handler.

Security for z/OS Connect

z/OS Connect is a WebSphere Liberty application, and has the same configuration and considerations as other WebSphere Liberty applications. In addition, z/OS Connect for CICS 1.0 and z/OS Connect Enterprise Edition have some extra security requirements.

z/OS Connect for CICS 1.0 and z/OS Connect Enterprise Edition have a RESTful management interface to allow dynamic service discovery. This interface is hosted at the same host name and port number as the individual JSON Services. The use of Transport Layer Security (TLS) to protect this interface, and the individual JSON Services, is encouraged.

By default, all client connections to z/OS Connect for CICS 1.0 and z/OS Connect Enterprise Edition must use the HTTPS protocol. The default behavior is to require a client-certified TLS connection to CICS. If this default is retained, client certificates must be associated with a SAF user ID. The application runs by using this certificate-derived identity.

Both z/OS Connect for CICS 1.0 and z/OS Connect Enterprise Edition can be configured to support the HTTP basic authentication protocol. This protocol allows a client to connect by using TLS in combination with a SAF user ID and password. To enable support for HTTP basic authentication, add the following line to the Liberty server configuration file (`server.xml`) : `<webAppSecurity allowFailOverToBasicAuth="true"/>`

Users of z/OS Connect for CICS 1.0 and z/OS Connect Enterprise Edition must be a member of the `zos.connect.access.roles.zosConnectAccess` EJBROLE. For more information, see Authorization using SAF role mapping

See Configuring authorization for applications in Liberty for Liberty information, Configuring security for z/OS Connect for z/OS Connect information, and Configuring security for z/OS Connect EE in z/OS Connect Enterprise Edition V3.0 product documentation for z/OS Connect EE information.

For further information, see Authorization using SAF role mapping, and Configuring security for a Liberty JVM server.

Configuring permissions for z/OS Connect Services and APIs

The CICS security model requires some additional actions in the way that you configure permissions for Services and APIs with z/OS Connect for CICS 1.0 and z/OS Connect Enterprise Edition.

About this task

When z/OS Connect is used to inject work into CICS, the following two identities are associated with the work at different stages of the processing:

- An initial, temporary identity is allocated during the process of attaching the work.
- An authenticated identity is then used to run the remainder of the work.

You can configure these identities in several ways, depending on your preferences and system environment.

Procedure

1. Optional: Create an alternative initial user ID for z/OS Connect.

   By default, the initial identity is the default CICS user ID, but you might choose to assign a different user ID to avoid giving the default CICS user ID permission to run transaction CPIH, or its equivalent.

   a) Authorize the alternative initial user ID to run transaction CPIH and any other transactions that are initiated through z/OS Connect.

      The initial user ID requires permission to run the target transaction for the Service or API.
2. Assign a default initial user ID. You can choose either or both of the following methods:

   - Set a user ID override value in the JVM profile for the JVMSERVER resource that hosts z/OS Connect.

     The following is an example override, where ZOSCUSER is the default initial user ID:
     `-Dcom.ibm.cics.jvmserver.http.userid=ZOSCUSER`

     Note: If you set a default initial user ID in the JVM profile, you do not need to provide a USERID value for each URIMAP. However, If you provide both a USERID for a URIMAP and an override value in the JVM profile, the USERID specified for a given URIMAP takes precedence.
   - Set the USERID field for a given URIMAP resource that targets z/OS Connect.

     When an HTTP request is received by z/OS Connect, CICS matches it against the URIMAP resources that are installed. If the URIMAP that is found specifies the USERID attribute, that user ID is used as the initial user ID, instead of the default initial user ID for the JVM server.

     Here is an example configuration for a URIMAP resource named ZOSCDEFT, where JVMSERVER is the USAGE value, a generic value is set for the PATH attribute, CPIH is the target transaction, and ZOSCUSER is the default initial user ID:

     ```
     NAME: ZOSCDEFT
     USAGE: JVMSERVER
     SCHEME: HTTP
     PORT: NO
     HOST: *
     PATH: /zosConnect/*
     ```

```
TRANSACTION: CPIH
USERID: ZOSCUSER
```

Note: URIMAP resources that are installed by using the PIPELINE SCAN mechanism are unlikely to be configured with a default user ID. In this scenario, you might consider specifying a user ID override value on the JVMSERVER.

Note: It is possible to store an initial user ID in a WSBind file: the user of DFHLS2JS or DFHJS2LS might provide a value for the **USERID** input parameter. If the **USERID** parameter is used, any URIMAPs that are produced during a PIPELINE SCAN include the requested initial user ID.

Results

You have now configured your environment so that CICS recognizes the URIs for your Services and APIs, and associates an initial user ID for use when the target transaction is attached.

Chapter 15. Security in BTS

Security considerations for CICS business transaction services are the authority to access BTS resources, the user IDs under which a process and its constituent activities run, the authority to attach the process and its constituent activities and the authority to use BTS system programming commands.

CICS security is described in detail in CICS TS security. Users of external security managers (ESMs) other than the Resource Access Control Facility (RACF) must read this information with the documentation for their own ESM.

Resource security in BTS

You can protect BTS resources such as processes, activities, and containers in the same way as resources accessed by CICS file control commands. That is, resource-level security for a process, its activities, and their containers is based on the CICS file definition that specifies the repository data set to which records for processes of this type are written.

Users who run programs that define or acquire processes or activities of a particular process-type need UPDATE access to the corresponding CICS file.

Note: When a task issues an ACQUIRE command, CICS allows the appropriate record to be read from the BTS repository, even if the userid associated with the request has only READ access. However, when the task issues a sync point the record is written back to the data set and, if the userid does not have UPDATE access, the task abends.

Users who inquire on or browse processes or activities of a particular process-type need at least READ access to the corresponding CICS file.

Process and activity user IDs

You can use either the RUN or LINK command to activate a process or activity. The command that you use affects the user ID under which the process or activity runs.

User IDs for activities activated by RUN commands

When a process or activity is activated by a RUN command, it might run using a different user ID than the transaction that issues the RUN.

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 user ID option of the DEFINE PROCESS or DEFINE ACTIVITY command. If the user ID option is omitted, its value defaults to the userid of the transaction that issues the DEFINE command.

The user ID obtained from the DEFINE command is referred to as the **defined process userid** or the **defined activity userid**. In the remainder of this section, we use the term “defined user ID” to mean either a defined process user ID or a defined activity user ID.

If the user ID option of DEFINE PROCESS or ACTIVITY 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 defined user ID. The RACF profile used for surrogate checking of a BTS process or activity is `userid.DFHSTART` in the SURROGAT class.

The following example RACF commands authorize a user as a surrogate user of a defined process userid and of a defined activity userid:

```
RDEFINE SURROGAT defined_process_userid.DFHSTART UACC(NONE)
          OWNER(defined_process_userid)

PERMIT defined_process_userid.DFHSTART CLASS(SURROGAT)
          ID(define_process_command_userid) ACCESS(READ)

RDEFINE SURROGAT defined_activity_userid.DFHSTART UACC(NONE)
```

```
          OWNER(defined_activity_userid)

PERMIT defined_activity_userid.DFHSTART CLASS(SURROGAT)
          ID(define_activity_command_userid) ACCESS(READ)
```

User IDs for activities activated by LINK commands

When a process or activity is activated by a LINK command, it runs under the userid of the transaction that issues the LINK.

Resource-level security checking *in* a process or activity is based on the user ID under whose authority the process or activity runs - that is, the defined userid or the userid of the transaction that issues the LINK command. This user ID must have UPDATE access to the CICS file that corresponds to the process-type.

Attach-time security for processes and activities

You can use Attach-time security to check if a transaction has authority to attach (activate) a process or activity. Attach-time security applies only when a process or activity is activated by a RUN command, not when it is activated by a LINK.

If attach-time security is required for a process, the defined userid - that is, the userid obtained from the DEFINE PROCESS command, must be given UPDATE access to the CICS file that corresponds to the BTS data set on which details of the process and its constituent activities are stored.

Command security in BTS

You can use CICS command-level security to protect your BTS system programming commands: **EXEC CICS CREATE PROCESSTYPE**, **DISCARD PROCESSTYPE**, **INQUIRE PROCESSTYPE**, and **SET PROCESSTYPE**.

Chapter 16. Security for the CICS-MQ adapter

The CICS-MQ adapter provides information to IBM MQ specifically for use in IBM MQ security.

Information provided is as follows:

- Whether CICS resource-level security is active for this transaction. For more information, see Security of resource definitions.
- User IDs.
  - For terminal tasks where a user has not signed on, the user ID is the CICS user ID associated with the terminal and is one of the following:
    - The default CICS user ID as specified on the CICS **DFLTUSER** system initialization parameter.
    - A preset security user ID specified on the terminal definition.
  - For nonterminal tasks, the adapter acquires the user ID with a call to the user domain.

Implementing security for CICS-MQ adapter transactions

If you want a user to administer the CICS-MQ adapter, you must grant the user authorization to the appropriate CICS transactions.

If required, you can restrict access to specific functions of the adapter. For example, if you want to allow users to display the current status of the adapter, but nothing else, give them access to CKQC, CKBM, CKRT, and CKDP only.

Define these transactions to CICS with RESSEC(NO) and CMDSEC(NO). For more details, see Security of resource definitions and CICS command security.

Transaction	Function
CKAM	Alert monitor
CKBM	Controls the adapter functions
CKCN	Connect
CKDL	Line mode display
CKDP	Full screen display
CKQC	Controls the adapter functions
CKRS	Statistics
CKRT	Controls the adapter functions
CKSD	Disconnect
CKSQ	CKTI START/STOP
CKTI	Trigger monitor

As well as administrators, user IDs connecting to IBM MQ, the user ID set in the **PLTPIUSR** system initialization parameter, and the CICSPlex SM MAS agent user ID must also be authorized to run the CKTI and CKAM transactions.

CICS-MQ adapter user IDs

The user ID associated with the CICS-MQ adapter is the user ID associated with the calling transaction that is accessing IBM MQ.

User ID checking for IBM MQ resources

If a CICS-MQ resource (MQCONN or MQMONITOR) is used to access IBM MQ, the user ID used by IBM MQ is the user ID of the transaction issuing the MQI command:

- For PLTPI programs, this is the **PLTPIUSR** system initialization parameter.
- For PLTSD programs, this is the user ID associated with the shutdown transaction.
- For other programs, this is the user ID associated with the running transaction.

User ID of the CICS-MQ trigger monitor

When security checking is active, you are recommended to always use an MQMONITOR to start CKTI instances. If you use an MQMONITOR, do not use CKQC. Using an MQMONITOR ensures that a single user ID, the **MONUSERID** attribute of the MQMONITOR, is always used for the CKTI instance irrespective of how it was started.

User ID of the CICS-MQ trigger monitor without an MQMONITOR

Starting CKTI without an MQMONITOR is still supported, but not recommended.

If an instance of CKTI is started without using an MQMONITOR, the user ID associated with the CKTI transaction is the user ID of the transaction starting CKTI:

- For PLTPI programs, this is the **PLTPIUSR** system initialization parameter.
- For CKQC users, this is the signed-on user ID running the transaction.
- If a sequential terminal is used to run CKQC, this is the user ID used to run the sequential terminal. The user ID must be a preset user ID rather than the default of the CICS default user ID. See The DFHTCT TYPE=TERMINAL macro.

Command security for MQCONN and MQMONITOR resources

Use CICS command security to control users' ability to issue SPI commands against MQCONN and MQMONITOR resource definitions. For example, you can use it to control which users are allowed to issue CREATE and DISCARD commands against the MQCONN resource definition for the CICS region.

When command security is enabled for a transaction, the external security manager checks that the user ID associated with the transaction is authorized to use the command on the MQCONN or MQMONITOR resource as appropriate. Resource security is not available for MQCONN and MQMONITOR resources.

CICS command security covers the **EXEC CICS CREATE MQCONN**, **DISCARD MQCONN**, **SET MQCONN**, **INQUIRE MQCONN**, **CREATE MQMONITOR**, **DISCARD MQMONITOR**, **SET MQMONITOR**, and **INQUIRE MQMONITOR** commands. For an explanation of command security and instructions to set up command security for a CICS region, see CICS command security. For a listing of the level of authority required for each command, see Resource and command check cross-reference.

When command security is active, the user ID for the running transaction that issues the **EXEC CICS SET MQCONN** command to start the connection to IBM MQ must have the following authority:

1. The authority to use the **EXEC CICS SET MQCONN** command; otherwise, the start of the connection will fail with a response of NOTAUTH and a RESP2 of 100.
2. The authority to use the **EXEC CICS EXTRACT EXIT** command; otherwise, the start of the connection will fail with a response of INVREQ and a RESP2 of 9. In this case, CICS issues messages DFHXS1111 and DFHMQ0302.

In addition, if MQMONITORs are being used, the user ID under which the MQMONITOR is running (specified in the **MONUSERID** parameter on the MQMONITOR definition) requires authorization for command security. This applies to MQMONITORs that are used to control the CICS-MQ trigger monitor, the CICS-MQ bridge, or user-written MQMONITOR programs. The MONUSERID must have the following authority:

1. The authority to use the **EXEC CICS SET MQMONITOR** command to set the status of the MQMONITOR to STARTED or STOPPED; otherwise, the MQMONITOR task will fail, and in the case of the CICS-MQ trigger monitor, CICS issues message DFHMQ0125.
2. In the case of the CICS-MQ trigger monitor, the authority to use the **EXEC CICS START** command with the **TRANSID** option set to the transaction that is specified in the trigger message; otherwise, CICS issues message DFHMQ0102 and the trigger message will be sent to the dead-letter queue.

Surrogate user security for MQMONITOR resources

Use CICS surrogate user security to control which transactions the CICS-MQ trigger monitor is allowed to start. This applies only when a trigger monitor instance has been started by an MQMONITOR.

When security checking is active and **XUSER=YES** is specified as a system initialization parameter, CICS® performs surrogate user checks when an **EXEC CICS START** command with the USERID option is used to start a transaction. If the CICS-MQ trigger monitor has been started by using an MQMONITOR, it uses information from the MQMONITOR definition to start the user transaction. The trigger monitor issues an **EXEC CICS START** command with the USERID option specifying a value obtained from the **USERID** attribute of the MQMONITOR.

CICS requires that the user ID associated with the transaction issuing the **START** request be a surrogate of the user ID associated with the started transaction. The CICS-MQ trigger monitor, when started by an MQMONITOR, always runs under the user ID specified in the **MONUSERID** attribute of the MQMONITOR. Therefore, for CKTI to be able to start the user transaction specified in the trigger message, the MONUSERID must be a surrogate of the user ID associated with the started user task. As the USERID specified in the MQMONITOR definition is used by default to start the user transaction if no suitable user ID is available from any other source, the MONUSERID must be a surrogate of the USERID as well. If the surrogate security check fails, CICS issues message DFHMQ0102 and the trigger message will be sent to the dead-letter queue.

IBM MQ connection security for the CICS-MQ adapter

IBM MQ carries out connection security checking either when an application program tries to connect to a queue manager by issuing an MQCONN or MQCONNX request, or when the channel initiator or the CICS-MQ adapter issues a connection request.

If you are using queue manager level security, you can turn connection security checking off for a particular queue manager, but, if you do that, any user can connect to that queue manager.

Only the CICS address space user ID is used for the connection security check, not the individual CICS terminal user ID.

You can turn IBM MQ connection security checking on or off at either queue manager or queue-sharing group level.

Chapter 17. Security for the CICS-MQ bridge

When you start the CICS-MQ bridge, you can specify the level of authentication. If requested, the bridge monitor checks the user ID and password extracted from the IBM MQ request message before running the CICS program named in the request message.

When you run the CICS-MQ bridge monitor transaction (for example, CKBR or your transaction name), you can specify the **AUTH** parameter to select one of the following levels of authentication:

LOCAL
This level is the default. The bridge monitor starts the bridge task with the CICS default user ID. CICS user programs that the bridge task runs have the authority associated with this user ID. The IBM MQ request message cannot request higher authority because any user IDs or passwords in the message are ignored. If the bridge task runs a CICS program that tries to access protected resources, the CICS program might fail.

IDENTIFY
If the message descriptor (MQMD) in the request message specifies a user ID, the bridge monitor starts the bridge task with that user ID. CICS user programs that the bridge task runs have the authority associated with that user ID. The user ID is treated as trusted; that is, the bridge monitor does not authenticate the ID by using password or PassTicket information. If the MQMD does not specify a user ID, the bridge monitor starts the bridge task with the CICS default user ID, in the same way as the LOCAL option.

VERIFY_UOW
The bridge monitor uses the password or PassTicket to authenticate the user ID if all the following conditions apply:

- The message descriptor (MQMD) in the request message specifies a user ID.
- The request message includes an IBM MQ CICS information header (MQCIH).
- The Authenticator field in the MQCIH contains a password or PassTicket.

If authentication succeeds, the bridge monitor starts the bridge task with that user ID. If authentication fails, the bridge monitor fails the request with a MQCRC_SECURITY_ERROR return code.

If any one of the conditions listed earlier is not met, the bridge monitor starts the bridge task with the CICS default user ID, in the same way as the LOCAL option. Only the first request message in the unit of work is checked; the bridge ignores user ID and password or PassTicket information in subsequent messages that are part of the same unit of work.

VERIFY_ALL
This level is the same as VERIFY_UOW, except that the bridge task also checks that the user ID is the same in every request message in the same unit of work, and reauthenticates the user ID for each request message, using the password or PassTicket that the message contains.

If you require different levels of authentication for different applications, use multiple bridge monitors with different transaction IDs. You can use CICS surrogate security to restrict the combinations of transaction and user ID that a bridge monitor transaction and user ID can start.

Table 54 on page 431 shows the user ID under which the bridge monitor is started. The user ID depends on the method that you use to run the bridge monitor transaction, typically CKBR.

Table 54. CICS-MQ bridge monitor security

Bridge monitor start method	At a signed on terminal	User ID for bridge monitor
From a terminal or **EXEC CICS LINK** in a program	Yes	Signed-on user ID
From a terminal or **EXEC CICS LINK** in a program	No	CICS default user ID

Table 54. CICS-MQ bridge monitor security (continued)		
Bridge monitor start method	**At a signed on terminal**	**User ID for bridge monitor**
EXEC CICS START with user ID	–	User ID from START
EXEC CICS START without user ID	–	User ID from START
The CICS-MQ trigger monitor CKTI	–	User ID from START
The CICS MQ monitor (MQMONITOR)	–	• The **MONUSERID** attribute of the MQMONITOR resource, if security checking is active for the CICS region (that is, the **SEC** system initialization parameter is set to YES) • User ID that started the MQMONITOR resource, if security checking is disabled for the CICS region (that is, **SEC** is set to NO)

User IDs and passwords in request messages

When you use the IDENTIFY, VERIFY_UOW, or VERIFY_ALL authentication options, the bridge task and the CICS programs that it runs are started with the user ID that is specified in the message descriptor (MQMD) in the request message. With the VERIFY_UOW and VERIFY_ALL options, the bridge monitor also checks the password specified in the IBM MQ CICS information header (MQCIH) in the request message.

To use these levels of authentication, the IBM MQ applications must provide a user ID in the MQMD, and they must provide an MQCIH including the password. You must define these user IDs to RACF. To control the user IDs used, the IBM MQ applications must open the request queue for the bridge monitor using open options that include MQOO_SET_IDENTITY_CONTEXT. The applications must also include a value of MQPMO_SET_IDENTITY_CONTEXT in their put message options.

If the bridge monitor finds a problem with the user ID or the password in a request message, it acts as follows:

- For the IDENTIFY level of authentication, if a message contains a user ID that was revoked, abend AICO might occur. The error reply has return code MQCRC_BRIDGE_ERROR with reason MQFB_CICS_BRIDGE_FAILURE.
- For the VERIFY_UOW or VERIFY_ALL level of authentication, if the user ID or password is invalid, the request fails with return code MQCRC_SECURITY_ERROR.
- If a request message omits either the user ID or the password, the bridge monitor runs the bridge task with the LOCAL level of authentication, even if you started the bridge monitor with one of the other authentication options. In that situation, the CICS programs started by the bridge task run with the user ID under which the bridge monitor was started.

You can use a PassTicket in place of a password to avoid the need to flow passwords in messages.

- The IBM MQ application must provide the PassTicket in the MQCIH in the request message.

- To generate a PassTicket, an application ID is required. The default application ID is the CICS APPLID. You can specify an alternative application ID by using the **PASSTKTA** parameter when you start the CICS-MQ bridge monitor transaction (for example, CKBR or your transaction name).
- If you use multiple bridge monitors for the same request queue, you must use the **PASSTKTA** parameter to specify the same application ID for each bridge monitor.

PassTicket validation is performed by using IBM MQ services, not **EXEC CICS VERIFY**, because the CICS service does not allow you to specify an APPLID. For more information about PassTickets, see Generating and using PassTickets for secure sign-on and Setting up security on z/OS in IBM MQ product documentation.

Authority

You must give the following permissions to the user IDs that you use with the CICS-MQ bridge. The user IDs include the user ID under which the bridge monitor transaction is started, as listed in Table 54 on page 431, and any user IDs that IBM MQ applications specify in request messages.

- The user ID under which the bridge monitor transaction is started must have authority to start the CKBP and CKBC transactions for CICS DPL programs, and any alternative transactions that IBM MQ applications specify in the TransactionId field in the MQCIH structure in request messages.
- If IBM MQ applications are specifying user IDs in request messages, the user ID under which the bridge monitor transaction is started must be defined to RACF as a surrogate of all the user IDs used in request messages. A surrogate user is one that has the authority to start work on behalf of another user, without knowing the password of the other user. For more information about surrogate user security, see Surrogate user security.
- The user IDs for the bridge monitor and for all bridge tasks need authority to get messages from the request queue.
- The user IDs for a bridge task need authority to put messages to its reply-to queue.
- To ensure that any error replies are received, the user ID under which the bridge monitor transaction is started must have authority to put messages to all reply-to queues.
- The user IDs for bridge tasks must have authority to put messages to the dead-letter queue.
- The user ID under which the bridge monitor transaction is started needs authority to put messages to the dead-letter queue, unless you want the bridge to stop if an error occurs.
- The user IDs for the bridge monitor and for all bridge tasks must have authority to put messages to the backout requeue queue, if one is defined.

Chapter 18. Kerberos support

CICS Transaction Server for z/OS provides support for Kerberos.

CICS supports Kerberos using the external security manager (ESM). The level of support depends on the support provided by the ESM. If your ESM is RACF, support is based on Kerberos Version 5 and Generic Security Services (GSS).

CICS can verify a Kerberos token by configuring a service provider pipeline or by using the API command **VERIFY TOKEN**.

The Kerberos service is enabled in the region by setting the **KERBEROSUSER** system initialization parameter. When you specify **KERBEROSUSER**, use a non-protected user ID to be associated with the Kerberos service principal for the CICS region.

Configuring RACF for Kerberos

You must configure an external security manager, such as RACF, to enable support for Kerberos.

Note: These instructions apply only to the use of local principals.

Before you begin

- You must set up a Kerberos environment for z/OS Security Server RACF by using the z/OS Integrated Security Services Network Authentication Service. For information about RACF, Kerberos and z/OS Integrated Security Services Network Authentication Service, see RACF and z/OS Integrated Security Services Network Authentication Service.
- The configuration `/etc/skrb/krb5.conf` defines the local realm for the LPAR or sysplex.
- The SKRBKDC started task must be running. For more information, see Setting up a Kerberos Key Distribution Center in z/OS Network File System Guide and Reference.

Procedure

1. Enable RACF protection for Kerberos. Activate RACF protection for the KERBLINK class by using the RACF **SETROPTS** command:

   ```
   SETROPTS CLASSACT(KERBLINK)
   ```

   For more information about the **SETROPTS** command, see z/OS Security Server RACF Command Language Reference.
2. To set up a CICS region to use Kerberos, define a service principal name and associate it with a user ID, as follows:
   a) Specify the **KERBEROSUSER** system initialization parameter.

      This parameter enables support for the Kerberos service in the region and specifies a user ID to be associated with the Kerberos service principal. You should use a non-protected user ID.

   b) Use the **ALTUSER** command to associate the service principal name with the user ID you specify in **KERBEROSUSER**.

      ```
      ALTUSER user_id KERB(KERBNAME(service_principal))
      ```
3. Associate a RACF user ID with the client principal by using the **ALTUSER** command:

   ```
   ALTUSER userid PASSWORD(password) NOEXPIRED KERB(KERBNAME(client_principal))
   ```

   Alternatively, you can associate a default user ID with all unassociated principals by using a command as follows:

```
RDEFINE KERBLINK /.../realm APPLDATA('userid')
```

where *userid* is the local user ID to be associated with all unmapped principals for the realm *realm*.

4. In order for this to be activated, a Kerberos key must be created. This is done automatically when the user next changes their password.

Configuring CICS web services for Kerberos

To configure a provider pipeline to implement Kerberos authentication, you must add a security handler to your pipeline configuration file.

Before you begin

You must configure an external security manager, such as RACF, to enable support for Kerberos. For more information, see Configuring RACF for Kerberos.

You must identify or create the pipeline configuration file to add the configuration information for Kerberos.

Procedure

1. Add a <wsse_handler> element to your pipeline. The handler must be included in the <service_handler_list> element. Code the following elements:

```
<wsse_handler>
    <dfhwsse_configuration version="1">
    </dfhwsse_configuration>
</wsse_handler>
```

The <dfhwsse_configuration> element is a container for the other elements in the configuration.

2. Code an <authentication> element.
   a) Code the **trust** attribute to specify `trust="basic"`.
   b) Code the **mode** attribute to specify `mode="basic-kerberos"`.
   c) Optional: Code an empty <suppress/> element.
      If you do not specify this element, the work runs under the user ID associated with the token.

Results

The CICS provider pipeline is configured for Kerberos authentication. Inbound web service requests received by the pipeline must include a valid Kerberos token. If they do not, the request is rejected with the appropriate SOAP fault. Depending on pipeline configuration options, the target application runs under the user ID associated with the token.

Example

The following example shows how you might configure the provider pipeline:

```
<provider_pipeline xmlns="http://www.ibm.com/software/htp/cics/pipeline">
    <service>
        <service_handler_list>
            <wsse_handler>
                <dfhwsse_configuration version="1">
                   <authentication trust="basic" mode="basic-kerberos"/>
                </dfhwsse_configuration>
            </wsse_handler>
        </service_handler_list>
        <terminal_handler>
            <cics_soap_1.1_handler/>
        </terminal_handler>
    </service>
    <apphandler>DFHPITP</apphandler>
</provider_pipeline>
```

Developing Kerberos applications

You can write a custom security handler to verify a Kerberos token, or you can write your own application to verify the token.

Before you begin

In the following steps, you can extract the RACF user ID of the Kerberos principal, by using the ISUSERID opintion of the **EXEC CICS VERIFY TOKEN** command. To do this, you must first define an association between the user ID and the principal. Use the RACF **RACLINK** command to set up the association. For more information, see Defining User ID Associations in z/OS Security Server RACF Security Administrator's Guide.

Procedure

Use one of the following techniques, according to your requirements:

- Write a security handler that uses the **VERIFY TOKEN** command, as described in Writing a custom security handler. If you want to run under the user ID of the Kerberos principal that is associated with the token, use the ISUSERID option of the **VERIFY TOKEN** command.
- Write your own front-end security program. Such a program might extract the Kerberos token from an HTTP header or an IBM MQ message and then issue the **VERIFY TOKEN** command. If you want to run under the user ID of the Kerberos principal that is associated with the token, use the ISUSERID option of the **VERIFY TOKEN** command to obtain the user ID. The new request can then be started with that user ID.

Using a Kerberos security token in a 3270 emulator sign-on

The use of Kerberos provides stronger security because passwords are not required to flow over the network.

The process is described as follows:

1. The Client terminal emulator applies to a Kerberos authentication server to obtain a Kerberos token.
2. The Kerberos token is returned to the Client terminal emulator, and the content is encoded in Base64 format.
3. The token is then forwarded in a message to the CICS server, where a sign-on transaction receives the Base64 encoded Kerberos token and issues the **SIGNON TOKEN** command.
4. The RACF Kerberos registry validates the Kerberos token and returns the associated RACF USERID to CICS. This USERID is associated with the terminal session for subsequent tasks.

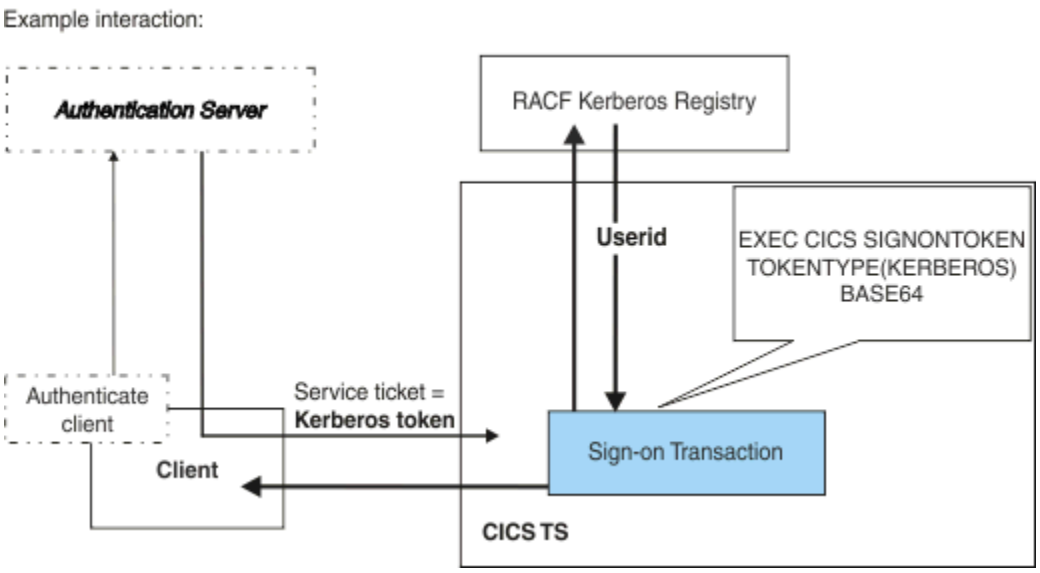


Figure 40. Flow of requests between the Client terminal emulator, the authentication server, and CICS TS

Note: Logon data cannot be used to send the Kerberos token since it is limited to 255 characters.

Chapter 19. Support for JWT using RACF

CICS Transaction Server for z/OS supports the use of JSON Web Tokens (JWTs) generated by RACF. With this capability, you can convert basic authentication credentials of a user to a time-limited secure token and then validate this secure token. This facility is particularly useful where applications that use passwords are being converted to using MFA tokens.

CICS supports only signed JWTs that are generated by using RACF. A JWT is in the form that is described by JSON Web Token (JWT) Specification RFC 7519.

Note: This capability requires RACF APAR OA55926 and SAF APAR OA55927.

Converting basic authentication credentials to JWTs

Using the **VERIFY TOKEN** command, CICS can convert basic authentication credentials of a user to a JWT and then validate the JWT. For more information about **VERIFY TOKEN**, see VERIFY TOKEN.

CICS can convert MFA tokens to JWTs to support the use of MFA tokens on stateless requests that cache credentials. For more information, see Chapter 20, “Support for Multi-Factor Authentication using RACF,” on page 441.

The **VERIFY TOKEN BASICAUTH** command can be used to generate a RACF JWT by using the OUTTOKEN option. The user application must return this information securely to the client (such as in the header). On return, the application can use this token in a **VERIFY TOKEN JWT**. The principle use case is when MFA is used for the basic auth token.

Important: CICS does not provide full JWT support. The signature algorithm must be HS256 or variants of that specification and does not support the use of private and public key pairings. The validation for the JWT must be generated by RACF. The claims must follow the rules as defined in RACF Identity Token (IDT) Support:V1.00 APARs: RACF OA55926, SAF OA55927. If you need to use public and private key pairings, you can use Liberty and Link to Liberty to call a function to achieve this result.

Configuring RACF for JWT

For a CICS region to support JWTs, you must create profiles in the IDTDATA class. Ensure to specify the IDTPARMS SIGTOKEN option because CICS supports only signed JWTs. The IDTDATA class must be active and RACLISTed.

Figure 41 on page 439 shows example RDEFINE statements to create such profiles. For more information, see Security Server RACF Command Language Reference.

```
SETROPTS CLASSACT(IDTDATA)
RDEFINE IDTDATA JWT.applid.userid.SAF IDTPARMS(SIGTOKEN(icsftoken))
```

applid
Specify the APPLID of the CICS region. If all CICS regions are supported, specify an asterisk *.

userid
Specify the CICS task user ID that is allowed to process JWTs. If all user IDs are supported, specify an asterisk *.

Figure 41. Example RDEFINE statements to create profiles for JWT support

Chapter 20. Support for Multi-Factor Authentication using RACF

CICS Transaction Server for z/OS provides support for Multi-Factor Authentication (MFA) using RACF.

If you are an RACF user, see Multi-Factor Authentication for z/OS in z/OS Security Server RACF Security Administrator's Guide for an overview of MFA and the prerequisite for this feature.

If you are using other security products, see the documentation of your ESM for details of support and prerequisites.

The following information describes how to implement MFA in CICS, based on the example of RACF and IBM Multi-Factor Authentication for z/OS.

Support overview

CICS supports in-band MFA tokens. If you use z/OS Out-of-Band authentication, a one-time-use token can be generated and is supported by CICS.

Which logon interfaces support MFA tokens

MFA tokens are supported on the following session-based logon interfaces:

Table 55. Session-based logon interfaces that support MFA

Interface	**CICS level requirement**
CICS Explorer	CICS TS V5.4 with APAR PI87691 or later
CESN and CESL	• CICS TS V4.2 with APAR PI21865 • CICS TS V5.1 with APAR PI21866 • CICS TS V5.2 with APAR PI21866 • CICS TS V5.3 or later
CICSPlex SM Web User Interface	
User-written sign-on programs using **EXEC CICS SIGNON**	

How to use MFA tokens on stateless requests

To use MFA tokens on stateless requests that cache credentials, the MFA token must be converted to a time limited security token, and the logged-on user must use this security token as the cached credentials.

CICS can convert an MFA token to a JWT using the **VERIFY TOKEN** command. For more information about **VERIFY TOKEN**, see VERIFY TOKEN.

Consideration on MFA input fields

Depending on the length, MFA tokens should be entered in the phase or password fields.

Learn more

- IBM Multi-Factor Authentication for z/OS User's Guide
- Configuring CMCI with CMCI JVM server in a WUI region
- Chapter 19, “Support for JWT using RACF,” on page 439

Chapter 21. Configuring CICS for SAML

To configure CICS to use SAML, first configure the JVM server by customizing and installing the sample JVM server profile, then install the CSD group in the appropriate CICS regions.

Before you begin

You must identify the regions where you want to deploy the CICS Security Token Service (STS). Install the STS in regions without any application code. If you have application code in the region where you will be validating your SAML token, define the STS remotely. You might also choose to define the region remotely if you prefer to separate regions that run Java code from other regions. Another reason for having a separate region for the STS is that you could define that region with its own keyring, which contains only those certificates that are required for signature validation and signing SAML tokens.

About this task

CICS provides a linkable interface called DFHSAML. The interface allows CICS web services pipelines and applications to validate and extract information from SAML assertions. CICS support for SAML requires a JVM server that is installed and configured on your system.

Important: Running a SAML JVM server with Java 11 is not supported.

Procedure

1. Create a JVM server profile for the JVM server.

   You can copy the appropriate supplied profile, DFHJVMST, from the installation directory to the directory that is specified by the **JVMPROFILEDIR** system initialization parameter.
2. Install CSD group DFHSAML in the chosen configuration:
   a) Install DFHSAML in the region that is chosen to run the STS.
   b) If you want to use SAML remotely, define a remote program definition for DFHSAML pointing to the region that runs the STS.

   Note: If you are using your own JVM server definition, copy DFHSAML, customize this group, and install the customized group instead of the DFHSAML group. The new group must point to your own JVM server definition. All programs that call the security token extensions support must create DFHSAML JVMSERVER containers with the name of their JVM server.

Results

CICS is configured for SAML.

What to do next

You can validate your configuration, as described in “Validating your configuration of CICS for SAML” on page 443.

Validating your configuration of CICS for SAML

A sample is provided, which you can use to verify that CICS is configured correctly for SAML. Two programs are provided, which can be compiled and then invoked through a transaction.

Before you begin

You must configure your JVM server, as described in Chapter 21, “Configuring CICS for SAML,” on page 443.

About this task

A sample is provided in CSD group DFH$SAML, which contains a program definition for sample programs, a transaction, and a template. You can use this sample to validate your configuration. When you compile and deploy the sample application, it provides an example SAML token assertion to be processed by the CICS security token extensions. The application is started by a CICS transaction.

Procedure

1. Optional: If you customized and installed a JVM server with a name other than DFHXSTS, update program DFH0XST2 to reflect the new server name.
2. Compile the programs DFH0XST1 and DFH0XST2, which are in the samples library, SDFHSAMP. For information about compiling COBOL programs, see Batch compilation for COBOL programs.
3. Install the group DFH$SAML in a region that calls the DFHSAML program.
4. Run transaction XST1.

Results

If the sample transaction XST1 runs successfully, SAML support is configured correctly.

The sample outputs the parsed containers into TSQ DFH0XSTO.

To look at these containers use **CEBR DFH0XSTO**.

If the installation validation is not successful, the DFHSAML-RESPONSE container contains a return code that indicates the reason. For more information about container response codes, see SAML support containers.

If an abend code is returned read the sample for further information.

What to do next

- You can replace the sample SAML token with your own. Create and install a DOCTEMPLATE resource definition that names the file that contains your SAML token. Specify this DOCTEMPLATE 48-byte TEMPLATENAME after the transaction identifier when you run the sample:

  ```
  XST1 templatename
  ```

  If no *templatename* is specified, the default TEMPLATENAME of DFH0XSTI is used.
- If you want to use signature validation, update program DFH0XST2. For more information, see the comments within that program.

Configuring a provider pipeline to use SAML tokens

Configure a provider pipeline that enables the validation and extraction of SAML token assertions and make them available to the CICS application.

Before you begin

If you are adding SAML support to an existing pipeline, identify the pipeline configuration file. If you are creating a new pipeline, create a new pipeline configuration file. CICS provides two sample configuration files, `samlprovider.xml` and `propagatesamlprovider.xml`. The latter contains the **tran_channel** attribute.

Procedure

1. Code an <sts_authentication> element in your pipeline configuration file.

   This element specifies that a Security Token Service (STS) is used for authentication and determines the type of request that is sent. Do not code an <authentication> element.

   a) Code the **action="validate"** attribute to specify that the STS verifies a security token.

If you do not specify this attribute, CICS assumes that the action is to request an identity token.

b) Optional: Code the **token_signature="ignored"** attribute to specify that CICS ignores the signature.

If you do not specify this attribute, the default value is `required`, which means that a valid signature must be supplied.

c) Optional: Code the **extract="no"** attribute to specify that the elements of the SAML token are not put into containers.

If you do not specify this attribute, the default value is `yes`, which means that CICS creates containers with the main elements of the SAML token. For full details of these containers, see SAML support containers.

d) Optional: Code the **tran_channel="yes"** attribute to specify that the SAML assertions are copied into containers in the DFHTRANSACTION channel to allow propagation of SAML information through a CICS application.

If you do not specify this attribute, the default value is no, which means that the assertions are made available in containers in the channel that is passed from the pipeline.

e) Within the <sts_authentication> element, code an <auth_token_type> element.

Within the <auth_token_type> element, code the following elements:

<namespace>
Set the content of this element to either `urn:oasis:names:tc:SAML:1.0:assertion` or `urn:oasis:names:tc:SAML:2.0:assertion`, depending on the version of SAML you are using.

<element>
Set the content of this element to `Assertion`.

2. Code an <sts_endpoint> element.

a) In the <sts_endpoint> element, code an <endpoint> element.

This element contains a URI that points to the location of the Security Token Service (STS) on the network.

To use the CICS Security Token Service to process SAML tokens, set the endpoint to `cics://PROGRAM/DFHSAML`.

The CICS Security Token Service is called by the security handler. If **extract="yes"** is configured in the <sts_authentication> element, containers prefixed with "DFHSAML" are copied to the pipeline channel and are therefore available to later pipeline handlers and the target application.

b) Optional: In the <sts_endpoint> element, code a <jvmserver> element.

This element identifies the server on which to run. If this element is not included, the default sample resource JVM server DFHXSTS is assumed.

Example

```
<?xml version="1.0" encoding="EBCDIC-CP-US"?>
<provider_pipeline xmlns="http://www.ibm.com/software/htp/cics/pipeline">
  <service>
    <service_handler_list>
      <wsse_handler>
        <dfhwsse_configuration version="1">
          <sts_authentication action="validate" token_signature="required"
           extract="yes" tran_channel="yes">
            <auth_token_type>
              <namespace>urn:oasis:names:tc:SAML:2.0:assertion</namespace>
              <element>Assertion</element>
            </auth_token_type>
          </sts_authentication>
          <sts_endpoint>
            <endpoint>cics://PROGRAM/DFHSAML</endpoint>
            <jvmserver>DFHXSTS</jvmserver>
          </sts_endpoint>
        </dfhwsse_configuration>
      </wsse_handler>
    </service_handler_list>
```

```
    <terminal_handler>
      <cics_soap_1.1_handler/>
    </terminal_handler>
  </service>
  <apphandler>DFHPITP</apphandler>
</provider_pipeline>
```

Configuring a requester pipeline to use SAML tokens

Configure a requester pipeline to insert SAML tokens in the SOAP header of outbound messages.

Before you begin

If you are adding SAML support to an existing pipeline, identify the pipeline configuration file. If you are creating a new pipeline, create a new pipeline configuration file. CICS provides two sample configuration files, `samlrequester.xml` and `propagatesamlrequester.xml`. The latter contains the **`tran_channel`** attribute.

About this task

In this task, you configure a requester pipeline to automatically attach SAML tokens to the outbound SOAP messages. CICS allows only previously validated tokens to be sent out in a SOAP message. The validated token is stored in the DFHSAML-OUTTOKEN container and the token inserted in the SOAP header is taken from this container.

Procedure

1. Code an `<sts_authentication>` element in your pipeline configuration file.

   This element specifies that a Security Token Service (STS) is used for authentication and determines the type of request that is sent. Do not code an `<authentication>` element.

   a) Optional: Code the **`tran_channel="yes"`** attribute to specify that the contents of the DFHTRANSACTION channel's DFHSAML-OUTTOKEN container are used as the SAML token for the request.

   If you do not specify this attribute, the default value is no, which means that the SAML token is taken from the DFHSAML-OUTTOKEN container in the channel that is passed to the SOAP pipeline.

   b) Within the `<sts_authentication>` element, code an `<auth_token_type>` element.

   Within the `<auth_token_type>` element, code the following elements:

   <namespace>
   Set the content of this element to either `urn:oasis:names:tc:SAML:1.0:assertion` or `urn:oasis:names:tc:SAML:2.0:assertion`, depending on the version of SAML you are using.

   <element>
   Set the content of this element to `Assertion`.

2. Code an `<sts_endpoint>` element.

   a) In the `<sts_endpoint>` element, code an `<endpoint>` element.

   Set the value of the endpoint to `cics://PROGRAM/DFHSAML`.

   b) Optional: In the `<sts_endpoint>` element, code a `<jvmserver>` element.

   This element identifies the server on which to run. If this element is not included, the default sample resource JVM server DFHXSTS is assumed.

Example

```
<?xml version="1.0" encoding="EBCDIC-CP-US"?>
<requester_pipeline xmlns="http://www.ibm.com/software/htp/cics/pipeline">
  <service>
    <service_handler_list>
      <cics_soap_1.1_handler/>
      <wsse_handler>
```

```
        <dfhwsse_configuration version="1">
          <sts_authentication tran_channel="yes">
            <auth_token_type>
              <namespace>urn:oasis:names:tc:SAML:2.0:assertion</namespace>
              <element>Assertion</element>
            </auth_token_type>
          </sts_authentication>
          <sts_endpoint>
            <endpoint>cics://PROGRAM/DFHSAML</endpoint>
            <jvmserver>DFHXSTS</jvmserver>
          </sts_endpoint>
        </dfhwsse_configuration>
      </wsse_handler>
    </service_handler_list>
  </service>
  <service_parameter_list/>
</requester_pipeline>
```

Configuring the CICS Security Token Service

To control the behavior of the CICS STS, update your STS configuration file and perhaps your `java.policy` file and system initialization parameters.

About this task

CICS uses a file referred to as the *STS configuration file* to control the behavior of the CICS STS. By default, CICS uses a file called `sts.xml` in the directory that is identified by the **JVMPROFILEDIR** SIT parameter. If that file does not exist, CICS uses the file specified by the JVM property **com.ibm.cics.sts.config** in the JVM server profile. For example: `-Dcom.ibm.cics.sts.config=/var/security/sts/sts-config.xml`. A sample file `sts-config.xml` is provided.

Procedure

1. Update the STS configuration file according to your requirements.
   a) If you want to update the issuer of the SAML token when you add attributes to the token, specify an *issuer*.
      i) Code an <issuer> element in your STS configuration file.
      ii) Within the <issuer> element, code a <format> element.
      iii) Within the <issuer> element, code a <uri> element.
   b) To allow for any time difference in system clocks on different computers, you can specify a tolerance value or *clock skew*.

      If you find that SAML tokens are regularly being rejected as expired, or as not yet valid, it might be caused by a discrepancy in clock time between the issuing and receiving systems. Setting a skew time can prevent SAML tokens from appearing to arrive before they are issued, or arriving already expired when they have, in reality, recently been issued. However, if you set too large a value, you might accept tokens that have genuinely expired.
      i) Code a <clock_skew> element in your STS configuration file. The value is set in milliseconds, so to set a time of 90 seconds, code `<clock_skew>90000</clock_skew>`.
   c) If you intend to re-sign a SAML token, specify the certificate label.
      i) Code a <signature> element in your STS configuration file. Set the `hash_algorithm` attribute to `sha-1` or `sha-2` according to the hash algorithm you want to use.

      Note: The hash algorithm `sha-3` is not supported.
      ii) Optional: Within the <signature> element, code a <certificate> element.
      iii) Optional: Within the <certificate> element, code a <label> element. Set it to the value of the RACF certificate label.
   d) If you intend to verify signatures or to re-sign SAML tokens, you can specify the keystore type.

i) Code a <keystore> element in the STS configuration file.

ii) Within the <keystore> element, code a <type> element. If you want to use cryptographic hardware to validate signatures, give it the value JCECCARACFKS. Otherwise, use the value JCERACFKS, which is the default value.

2. Optional: If you intend to verify signatures or to re-sign SAML tokens, update the `java.policy` file and set the **KEYRING** SIT parameter.

   a) Update the `java.policy` file to give the CICS user permission to alter the list of Java security providers.

   A sample file, `sts-java.policy`, is provided.

   i) To use the sample file, edit it to change `/&USSHOME/` to the appropriate value for your installation, and set the JVM property **`java.security.policy`** in the JVM server profile to specify its path and file name.

   ii) If you use an existing `java.policy` file, update it to include the following rule:

```
// All permissions granted to CICS codesource protection domain
grant codeBase "file:///&USSHOME///-" {
permission java.security.AllPermission;
};
```

   where *USSHOME* is the name and path of the root directory for CICS Transaction Server files on z/OS UNIX.

   b) Set the SIT parameter **KEYRING** to specify the RACF keyring that contains the set of certificates you want to use.

The STS configuration file

The STS configuration file specifies various aspects of the CICS Security Token Service (STS).

CICS uses a file called `sts.xml` in the directory that is identified by the **JVMPROFILEDIR** SIT parameter. If that file does not exist, CICS uses the file that is specified by the JVM property **`com.ibm.cics.sts.config`** in the JVM server profile. For example: `-Dcom.ibm.cics.sts.config=/var/security/sts/sts-config.xml`. A sample STS configuration file, `sts-config.xml`, is supplied for reference.

The XML schema file, `sts.xsd`, is supplied in the `/usr/lpp/cicsts/cicsts52/schemas/sts` directory.

The STS configuration file contains the following elements:

<keystore>
Defines the RACF keystore type. The possible values are JCERACFKS and JCECCARACFKS. The default value is JCERACFKS.

<issuer>
Defines the STS as an asserting party. This element contains the following elements:

<format>
Contains any string. There is no default value.

<uri>
Contains any string. The default value is `http://cics`.

<signature>
Specifies the hash algorithm and the certificate label. This element has the following attribute:

hash_algorithm
Possible values are `sha-1` and `sha-2`. The default value is `sha-2`.

This element contains the following element:

<certificate>
This element contains the following element:

<label>
The value of the RACF certificate label. The default value is CICSCERT.

<clock_skew>
The clock skew time, in milliseconds. The default value is 180000 ms (3 minutes).

A clock skew allows for any time difference in system clocks on different computers. It is applied to all timing conditions in a SAML token.

The following figure shows an example of an STS configuration file with all elements set:

```
<sts_configuration xmlns="http://www.ibm.com/xmlns/prod/cics/JVMSERVER/stsconfig">

    <keystore>
        <type>JCECCARACFKS</type>
    </keystore>

    <issuer>
        <format>urn:oasis:names:tc:SAML:2.0:nameid-format:entity</format>
        <uri>http://cics</uri>
    </issuer>

    <signature hash_algorithm="sha-1">
        <certificate>
            <label>CICSCERT</label>
        </certificate>
    </signature>

    <clock_skew>90000</clock_skew>

</sts_configuration>
```

Patterns for developing SAML-aware programs

SAML aware programs might conform to common patterns. One such pattern is an initial program that controls access to parts of the application. Another is logging information about the user.

Pattern: reusing a validated token

You might want to validate a SAML token and later in the same transaction call a web service from a requester program and use the same token. A validated SAML token is held in the DFHSAML-OUTTOKEN container. As this container is read-only, it cannot be moved between channels. To avoid having to reissue the validation request, and thus to improve performance, you can use the transaction channel, DFHTRANSACTION.

When you validate a SAML token from an incoming web service, code the **tran_channel="yes"** attribute in the <sts_authentication> element in the configuration file for your provider pipeline. This attribute specifies that the SAML assertions are copied from the output containers into containers in the DFHTRANSACTION channel.

To reuse the validated SAML token in a web service, code the **tran_channel="yes"** attribute in the <sts_authentication> element in the configuration file for the requester pipeline that is used by the web service.

Chapter 22. Developing SAML applications

You can develop applications to use the SAML facilities provided in CICS.

About this task

You might choose to develop applications to process SAML tokens in various ways. For example:

- Writing a program to extract a SAML token from an incoming message
- Writing an application to place a SAML token in an outbound request
- Writing an application to add attributes to a SAML token and re-sign the token
- Writing an application to create a SAML token

Developing a SAML-aware initial program

You can write a program to extract a SAML token and link to DFHSAML to process the token.

About this task

As an alternative to using the web services support (see Configuring a provider pipeline to use SAML tokens), you can write your own SAML-aware initial program as a security front end to validate SAML tokens. Such a program can be useful when you are using messages that use HTTP, IBM MQ , or any other protocol, rather than a SOAP message.

The SAML-aware initial program extracts the SAML token from the message. It then puts the token into a character container, DFHSAML-TOKEN, in a channel. The program then links to the program DFHSAML with the channel to validate the token and extract the containers. For details of the containers, see SAML support containers.

You can use either a user-defined channel or the transaction channel (DFHTRANSACTION). If you are using a user-defined channel, the containers are passed on LINK requests that explicitly pass that channel. If you are using the transaction channel, the containers are available throughout the transaction.

The application can use information in the containers, such as SAML attributes, by using **GET CONTAINER** commands.

For a specific example of how you might use this information, see “Pattern for developing a SAML-aware initial program” on page 451.

Pattern for developing a SAML-aware initial program

SAML aware programs might conform to common patterns. One such pattern is an initial program that controls access to parts of the application.

A typical pattern for an application is for the initial program to be SAML-aware. In this pattern, the program uses the information in the SAML assertion to make decisions before it runs appropriate parts of the application.

The information in the SAML assertion is in read-only containers. These containers are either in the channel that is passed to the program from the SOAP pipeline or, if the program is doing its own message handling and SAML validation, returned from the DFHSAML program.

An example of the processing that the SAML-aware program does is obtaining information from the attribute containers. It looks for an attribute name container (for example, DFHSAML-ATTRN001) and an attribute value in that container (for example, DFHSAML-A001V001). This attribute might be used by the program to represent the role or authority of the user and so allow the application to select which parts of the application are available to the caller.

The application might have to pass on information to other programs in the application. Because the containers are read-only, information can be passed securely by passing the channel to the next program, or transaction that uses the CHANNEL interface, on the LINK, XCTL, RETURN, or START commands.

The application might also have to audit the request so that the transaction can be associated with the original SAML token, and hence the user. To make this association, write a customer logging program, which writes the validated SAML token (in container DFHSAML-OUTTOKEN) or selected containers to a journal.

Developing a program that uses a validated SAML token in an outbound request

Configure web services to add a verified SAML token to an outbound request.

About this task

You might want your application to use the SAML token in an outbound request. You can configure web services to add the SAML token to an outbound request. For instructions, see Configuring a requester pipeline to use SAML tokens.

For an application to insert a SAML token in a web service request SOAP message, the token must first be validated. Optionally, the application can add attributes to the token after validating it and before invoking the web service. For more information, see “Developing a program that creates or augments SAML tokens” on page 452.

The SAML token is stored in the read-only container DFHSAML-OUTTOKEN.

Developing a program that creates or augments SAML tokens

You can develop a CICS application or a web service application to create a SAML token or to add attributes to a SAML token and to re-sign it.

About this task

You can use CICS SAML support to add attributes to SAML tokens and re-sign the request with the certificate specified in the CICS STS configuration file. The SAML token might be received from an external sender or created from a template. The application that you develop can be either a CICS application or a web service application.

Attention: Create, augment, and re-sign tokens only on a region where all application code that participates in the augmenting of the token is trusted by other members of the federation.

Procedure

1. Add the attributes by creating the following containers in the same channel that was used to validate the original SAML token.

   Note: You must have validated the token before you can modify it. The validated token is contained in the DFHSAML-OUTTOKEN container.

   a) Put the attribute name into container DFHSAML-ATTRN *aaa* , where *aaa* are three uppercase alphanumeric characters.
   For example:

   ```
   EXEC CICS PUT CONTAINER('DFHSAML-ATTRNORG')
   CHANNEL('SAML-CHANNEL') FROM('title')
   ```

   b) Optional: Put the attribute name space into container DFHSAML-ATTRS *aaa* , where *aaa* are the same characters as you used for the attribute name container.

   This step is not required for SAML version 2.0.

c) Optional: Put the attribute friendly name into container DFHSAML-ATTRY *aaa* , where *aaa* are the same characters as you used for the attribute name container.
For example:

```
EXEC CICS PUT CONTAINER('DFHSAML-ATTRYORG')
CHANNEL('SAML-CHANNEL') FROM('eduPersonAffiliation')
```

d) Optional: Put the attribute format into container DFHSAML-ATTRF *aaa* , where *aaa* are the same characters as you used for the attribute name container.
For example:

```
EXEC CICS PUT CONTAINER('DFHSAML-ATTRNORG')
CHANNEL('SAML-CHANNEL')
FROM('urn:oasis:names:tc:SAML:2.0:attrname-format:uri')
```

e) Optional: Put one or more attribute values into containers DFHSAML-A *aaa* N *bbb* , where *aaa* are the same characters as you used for the attribute name container and *bbb* are three uppercase alphanumeric characters.
For example:

```
EXEC CICS PUT CONTAINER('DFHSAML-AORGV001')
CHANNEL('SAML-CHANNEL') FROM('staff')
EXEC CICS PUT CONTAINER('DFHSAML-AORGV002') CHANNEL('SAML-CHANNEL')
FROM('employee')
```

2. Create the token in either of the following ways:

- Put the SAML-ISSUE value in the DFHSAML-FUNCTION container and link to the linkable interface DFHSAML, which creates the new token. By default, the token is re-signed using the signature options that are specified in the STS configuration file. If no signature is required, the application can create a DFHSAML-SIGNED container with the SAML-IGNORED option specified before it calls DFHSAML. If an <issuer> is specified in the STS configuration file, its value is used in the new SAML token.
- Invoke a web service. If the requester pipeline associated with the web service is configured for SAML, it automatically adds attributes to the original token and creates a new one. By default, the pipeline re-signs the SAML token by using the signature options that are specified in the STS configuration file. If no signature is required, set the requester pipeline configuration option token_signature to no. If an <issuer> is specified in the STS configuration file, its value is used in the new SAML token. In addition to creating a new SAML token, the requester pipeline also inserts the SAML token in the outbound web service request.

Notices

This information was developed for products and services offered in the U.S.A. This material might be available from IBM in other languages. However, you may be required to own a copy of the product or product version in that language in order to access it.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property rights may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive, MD-NC119
Armonk, NY 10504-1785
United States of America

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan Ltd.
19-21, Nihonbashi-Hakozakicho, Chuo-ku
Tokyo 103-8510, Japan

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION “AS IS” WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE. Some jurisdictions do not allow disclaimer of express or implied warranties in certain transactions, therefore this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM websites are provided for convenience only and do not in any manner serve as an endorsement of those websites. The materials at those websites are not part of the materials for this IBM product and use of those websites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who want to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact

IBM Director of Licensing
IBM Corporation
North Castle Drive, MD-NC119 Armonk,
NY 10504-1785
United States of America

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Programming License Agreement, or any equivalent agreement between us.

The performance data discussed herein is presented as derived under specific operating conditions. Actual results may vary.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to actual people or business enterprises is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. The sample programs are provided "AS IS", without warranty of any kind. IBM shall not be liable for any damages arising out of your use of the sample programs.

Programming interface information

CICS supplies some documentation that can be considered to be Programming Interfaces, and some documentation that cannot be considered to be a Programming Interface.

Programming Interfaces that allow the customer to write programs to obtain the services of CICS Transaction Server for z/OS, Version 5 Release 6 are included in the following sections of the online product documentation:

- Developing applications
- Developing system programs
- CICS TS security
- Developing for external interfaces
- Application development reference
- Reference: system programming
- Reference: connectivity

Information that is NOT intended to be used as a Programming Interface of CICS Transaction Server for z/OS, Version 5 Release 6 , but that might be misconstrued as Programming Interfaces, is included in the following sections of the online product documentation:

- Troubleshooting and support
- CICS TS diagnostics reference

If you access the CICS documentation in manuals in PDF format, Programming Interfaces that allow the customer to write programs to obtain the services of CICS Transaction Server for z/OS, Version 5 Release 6 are included in the following manuals:

- Application Programming Guide and Application Programming Reference

- Business Transaction Services
- Customization Guide
- C++ OO Class Libraries
- Debugging Tools Interfaces Reference
- Distributed Transaction Programming Guide
- External Interfaces Guide
- Front End Programming Interface Guide
- IMS Database Control Guide
- Installation Guide
- Security Guide
- Supplied Transactions
- CICSPlex SM Managing Workloads
- CICSPlex SM Managing Resource Usage
- CICSPlex SM Application Programming Guide and Application Programming Reference
- Java Applications in CICS

If you access the CICS documentation in manuals in PDF format, information that is NOT intended to be used as a Programming Interface of CICS Transaction Server for z/OS, Version 5 Release 6 , but that might be misconstrued as Programming Interfaces, is included in the following manuals:

- Data Areas
- Diagnosis Reference
- Problem Determination Guide
- CICSPlex SM Problem Determination Guide

Trademarks

IBM, the IBM logo, and ibm.com® are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at 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.

Apache, Apache Axis2, Apache Maven, Apache Ivy, the Apache Software Foundation (ASF) logo, and the ASF feather logo are trademarks of Apache Software Foundation.

Gradle and the Gradlephant logo are registered trademark of Gradle, Inc. and its subsidiaries in the United States and/or other countries.

Intel, Intel logo, Intel Inside, Intel Inside logo, Intel Centrino, Intel Centrino logo, Celeron, Intel Xeon, Intel SpeedStep, Itanium, and Pentium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

The registered trademark Linux® is used pursuant to a sublicense from the Linux Foundation, the exclusive licensee of Linus Torvalds, owner of the mark on a worldwide basis.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Red Hat®, and Hibernate® are trademarks or registered trademarks of Red Hat, Inc. or its subsidiaries in the United States and other countries.

Spring Boot is a trademark of Pivotal Software, Inc. in the U.S. and other countries.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Zowe™, the Zowe logo and the Open Mainframe Project™ are trademarks of The Linux Foundation.

Terms and conditions for product documentation

Permissions for the use of these publications are granted subject to the following terms and conditions.

Applicability
These terms and conditions are in addition to any terms of use for the IBM website.

Personal use
You may reproduce these publications for your personal, noncommercial use provided that all proprietary notices are preserved. You may not distribute, display or make derivative work of these publications, or any portion thereof, without the express consent of IBM.

Commercial use
You may reproduce, distribute and display these publications solely within your enterprise provided that all proprietary notices are preserved. You may not make derivative works of these publications, or reproduce, distribute or display these publications or any portion thereof outside your enterprise, without the express consent of IBM.

Rights
Except as expressly granted in this permission, no other permissions, licenses or rights are granted, either express or implied, to the publications or any information, data, software or other intellectual property contained therein.

IBM reserves the right to withdraw the permissions granted herein whenever, in its discretion, the use of the publications is detrimental to its interest or, as determined by IBM, the above instructions are not being properly followed.

You may not download, export or re-export this information except in full compliance with all applicable laws and regulations, including all United States export laws and regulations.

IBM MAKES NO GUARANTEE ABOUT THE CONTENT OF THESE PUBLICATIONS. THE PUBLICATIONS ARE PROVIDED "AS-IS" AND WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, AND FITNESS FOR A PARTICULAR PURPOSE.

IBM online privacy statement

IBM Software products, including software as a service solutions, (*Software Offerings*) may use cookies or other technologies to collect product usage information, to help improve the end user experience, to tailor interactions with the end user or for other purposes. In many cases no personally identifiable information is collected by the Software Offerings. Some of our Software Offerings can help enable you to collect personally identifiable information. If this Software Offering uses cookies to collect personally identifiable information, specific information about this offering's use of cookies is set forth below:

For the CICSPlex SM Web User Interface (main interface):
Depending upon the configurations deployed, this Software Offering may use session and persistent cookies that collect each user's user name and other personally identifiable information for purposes of session management, authentication, enhanced user usability, or other usage tracking or functional purposes. These cookies cannot be disabled.

For the CICSPlex SM Web User Interface (data interface):
Depending upon the configurations deployed, this Software Offering may use session cookies that collect each user's user name and other personally identifiable information for purposes of session management, authentication, or other usage tracking or functional purposes. These cookies cannot be disabled.

For the CICSPlex SM Web User Interface ("hello world" page):
Depending upon the configurations deployed, this Software Offering may use session cookies that collect no personally identifiable information. These cookies cannot be disabled.

For CICS Explorer:

Depending upon the configurations deployed, this Software Offering may use session and persistent preferences that collect each user’s user name and password, for purposes of session management, authentication, and single sign-on configuration. These preferences cannot be disabled, although storing a user's password on disk in encrypted form can only be enabled by the user's explicit action to check a check box during sign-on.

If the configurations deployed for this Software Offering provide you, as customer, the ability to collect personally identifiable information from end users via cookies and other technologies, you should seek your own legal advice about any laws applicable to such data collection, including any requirements for notice and consent.

For more information about the use of various technologies, including cookies, for these purposes, see IBM Privacy Policy and IBM Online Privacy Statement, the section entitled *Cookies, Web Beacons and Other Technologies* and the IBM Software Products and Software-as-a-Service Privacy Statement.

Index

Special Characters

-ESM interface
  overview 212
  RACROUTE macros 212
* 17
** (double asterisk)
  in data set profile names 17
% 17

A

access authorization levels 83, 93
access control
  HFS files 91
  z/OS UNIX files
  90
access lists
  avoiding with UACC(READ) 71
  conditional, for transaction profiles 71
  PERMIT command to create 24
accounting 328
ACICSPCT general resource class 84, 87
activating RACF classes 25
activating security parameters 207
activating user-defined RACF classes 140
activities
  security of 425, 426
ADDMEM operand 319
ADDUSER command
  defining the userid for CICS to RACF 32
administration
  security 425
Advanced Encryption Standard
  encryption algorithm 294
algorithm 410, 412
algorithms
  encryption 294
ALLOCATE_PIPE command
  security check failure 337
APPC PEM (password expiration management)
  ATTACH security fields 252
  benefits 243
  buffer size 252
  CICS activity 247
  data from CICS to PEM client 254
  EBCDIC for userids and passwords 252
  information on passwords 243
  overview of processing 247
  permitted userid and password length 252
  processing done by CICS PEM server 247
  processing required by PEM client 247
  PROFILE option 252
  sample configuration 244
  setting up the PEM client 252
  sign-on data sent to CICS PEM server 253
  sign-on input data sent by PEM client 253
APPC PEM (password expiration management) *(continued)*
  sign-on request, formatting errors 257
  sign-on status 243
  unsuccessful sign-on with PV 248
  using with persistent verification (PV) 247
APPCLU general resource class
  defining profiles 225
APPL general resource class
  controlling access to CICS region 43
application
  security 176, 202, 217
application program security
  access authorization levels 89
  defining resource classes 88
  MCICSPPT general resource class 88
  NCICSPPT general resource class 88
  QUERY SECURITY command 5, 131
applying a security policy 379
AT-TLS
  Application Transparent Transport Layer Security 392, 396, 397
  PROTOCOL(HTTP) 392, 396, 397
  SSL(ATTLSAWARE) 392, 396, 397
  SSL(NO) 392, 396, 397
AT-TLS Basic 392
AT-TLS Controlling 392
Atom collection
  security 403
Atom feed
  security 403
attach-time security 426
ATTACHSEC attribute 272
ATTACHSEC operand
  IDENTIFY parameter 231
  LOCAL parameter 231
  MIXIDPE parameter 231
  PERSISTENT parameter 231
  VERIFY parameter 231
ATTLS 396, 397
ATTLS Aware 396, 397
ATTLS Basic 396, 397
ATTLS Controlling 396, 397
ATTLSAWARE 392
auditing
  bind security failure 227
  requested by CICS on authorization requests 80
  second request to RACF to write log data 72
  SMF type 80 log records 72
AUTHENTICATE(AUTOREGISTER) 397
AUTHENTICATE(CERTIFICATE) 397
authentication 383–385
authentication, RPC 339
AUTHID
  surrogate security 116
authorization failures
  CICS resources 79
  command security 131

authorization failures *(continued)*
error messages 72
ICH408I, RACF message 72
authorization IDs, primary 327
authorization IDs, secondary 329
authorizing CICS region userid as surrogate user 45
authorizing CICS users to RACF 66
authorizing SYS1.PARMLIB libraries
data sets 167
AUTHTYPE security 320
AUTHTYPE values for security
authorization ID 323
group ID 323
sign ID from DB2CONN 323
terminal ID 323
transaction ID 323
user ID 323
autoinstall models 63

B

backup while open (BWO) 39
basic authentication 383–385
batch access to CSD, restricting 62
batch call interface 277
BCICSPCT general resource class 84, 87
bind-time security
for IPCONN 261
for IPIC connections 261
introduction 221
MRO links 269
overview 221
BINDSECURITY option 226
BMS commands 78
BUILD ATTACH command 273
building a keyring 300
BWO (backup while open) 39

C

cataloged procedures
authorizing CICS as a started task 31
CCICSCMD general resource class 130, 138
CCRL
running from a START command 314
running from a terminal 313
CDT (class descriptor table)
IBM-supplied default classes 26
resource length 138
setting up installation-defined classes 26, 139
CEDA LOCK command 62
CEDA transaction 62
CEDF 94
CEDF transaction 94, 130
CEDX 94
CEDX transaction 94
CEMT, main terminal transaction
and CRTE 234, 236, 266, 275
considerations for command security 131
system programming commands 118
certificate 287, 293
certificate authorities 294
certificate authority 287, 293
certificate revocation list 312
certificates, creating new 302
CESL transaction 55
CESN transaction 55
CFDT server authorization 282
CFRM policy 282
CICS as an HTTP client
security 384
SSL 392
CICS as an HTTP server
security 383, 385, 387
CICS business transaction services
administration
security 425
security
attach-time 426
command-level 426
resource-level 425
CICS command security
VCICSCMD general resource class 319
CICS commands and resources
creating security profiles with RACF
controlling access to resources 176
CICS JOB statement, PASSWORD parameter 32
CICS JOB statement, USER parameter 32
CICS load libraries, protecting 30
CICS region
access to 43
access to APPL class profiles 44
remote 44
userid as security token 46
CICS region userid
in started jobs 31
CICS resource security
XCICSDB2 general resource class 318
CICS resources
protecting 19
CICS security 316
CICS security, controlling 205
CICS segment 11
CICS SIT parameters
security-related 207
CICS source libraries, protecting 30
CICS system definition file (CSD), restricting batch access to 62
CICS system initialization parameters
SEC 340
XCMD 340
XPPT 340
XUSER 340
CICS user restart program, PLTPI 73
CICS users 2
CICS web support
security 383
CICS-RACF security interface
how ESM exit programs access CICS-related information 159
installation data parameter list 160
RACF user exit parameter list 159
system authorization facility (SAF) 159, 212
CICS-supplied RACF dynamic parse validation routines 29
CICS-supplied transactions security 142
CICSPlex SM
authorizing

CICSPlex SM *(continued)*
authorizing *(continued)*
libraries 167
procedures 170
protecting
with another ESM 212
with RACF 163
resource names 176
CICSPlex SM definitions, protecting
adding SAF resource classes 176
CICSPlex SM resource classes
controlling access to 176
CICSPlex SM security profiles
creating 167
cipher suite specification file 294
cipher suites
identifying which are used 308
restricting with CIPHERS attribute 294, 308
supported by z/OS and CICS 294
CIPHERS attribute
CORBASERVER resource 294, 308
TCPIPSERVICE resource 294, 308
URIMAP resource 294, 308
CKBM security 427
CKCN security 427
CKDL security 427
CKDP security 427
CKQC transaction
security 427
CKRS security 427
CKRT security 427
CKSD security 427
CKSQ security 427
CKTI transaction
security 427
class descriptor table (CDT) 138
class, resource 10
class, resource grouping 10
classification of data and users 26
CLAUTH (class authority) attribute
in CICS-related general resource classes 9
in user's profile 9
installation-defined classes 26, 140
client certificate 383, 384
client program
MRO logon and bind-time security 337
ClientAuthType 392, 396
clock skew
changing 447
setting 447
clock_skew 447
CLOUD.APPLICATION security profile 202, 217
CLOUD.DEF security profile 202, 217
CLOUD.PLATFORM security profile 202, 217
CLS4 transaction
XTRANID X'06F3F0F1' 252
CMDSEC 340
CMDSEC operand of DEFINE TRANSACTION 266
CMDSEC system initialization parameter 130
CMDSEC, command security parameter 130
COMAUTHID
surrogate security 116
command authorization
Db2 330
command security
authorization failures 131
CCICSCMD general resource class 130
CEMT considerations 131
CICS resources subject to 120
for IPIC connections 266
for ISC over TCP/IP connections 266
QUERY SECURITY command 131
specifying 129
VCICSCMD general resource class 129
XCMD parameter 49, 77
XUSER parameter 62
command-level security 426
conditional access lists 71
conditional access processing 59
configuration files
Security Token Service 448
configuring 443
configuring RACF 413
configuring the pipeline 417
connection authorization 324
connection exit routine 325
connection security 429
CONSOLE general resource class
description 60
coupling facility data table pool 281
coupling facility data tables security 281
COVA 172
COVC 172
COVE 172
COVG 172
COVP 172
COVU 172
CPLT transaction 114
CPSMOBJ general resource class 20
CPSMXMP general resource class 20
CRL (certificate revocation list) 312
CRTE, routing transaction 234, 236, 266, 275
CSCS transient data destination 61
CSD (CICS system definition file), restricting batch access to 62
CSD definitions, locking 63
CSMI (CICS-supplied mirror transaction)
authorizing the link user ID 338
security 338
custom security handler 420
customizing the CICS-RACF interface
determining userid of CICS region 150
ESMEXITS parameter 48
installation data parameter list 160
RACF user exit parameter list 159

D

Data Encryption Standard
encryption algorithm 294
data for default user 65
data set
encryption 334
data set profile 10
data set profiles
enhanced generic naming 17
SETROPTS EGN command 17
data set security

data set security *(continued)*
access to CICS data sets 36
access to user data sets 39
APPLID parameter 37
CICS installation requirements 29
CICS system 30
MVS library lookaside (LLA) facility 38
data tables
bind security 280
CONNECT security checks 280
coupling facility 281
file security 280
security checking 279
server authorization security check 279
date subfields, format 255
Db2 security
accounting 328
authorization to execute a plan 331
establishing authorization IDs 327, 329
primary authorization ID 324
secondary authorization ID 324
security mechanisms 317
DB2ENTRY resource classes 23
DCICSDCT general resource class
defining profiles 80
default user 113
default user, CICS
defining 34
DFLTUSER parameter 48
specifying on SIT 48
DEFINE CONNECTION
ATTACHSEC attribute 272
ATTACHSEC operand 230, 240
BINDSECURITY operand 226
SECURITYNAME option 226
DEFINE TRANSACTION
CMDSEC operand 266
RESSEC operand 235, 241, 266
defined activity userid 425
defined process userid 425
defining profiles
APPCLU general resource class 225
defining to RACF
groups 66
users 66
users, example 67
definitions, protecting
controlling access to resources 176
delegation of RACF administrative responsibility 9
DELMEM operand 59
deploying 443
DES
encryption algorithm 294
DES (data encryption standard) 293
DES authentication 339
developing
applications
Kerberos 437
DFH$RACF 27, 52
DFHAPPL FACILITY class profiles, defining 338
DFHEXCI surrogate profile 117
DFHHTML 173
DFHINSTL surrogate profile 117
DFHIRP (interregion communication program)
DFHIRP (interregion communication program) *(continued)*
security checks performed by 337
DFHSNxxxx messages 61
DFHSTART surrogate profile 117
DFHWBPW, password expiry management program 383, 385
DFHXCIS 277
DFHXCOPT, EXCI options table 116
DFLTUSER parameter
definition 15
obtaining user data 65
DFLTUSER SIT parameter
creating a security environment 167
DFLTUSER, system initialization parameter 48
diagnostic messages 396
DIGTCERT general resource class 20
discrete profile 10
distributed program link (DPL)
with LU6.2 237
with MRO 277
document template security
defining resource classes 73
RCICSRES resource class 73
WCICSRES grouping class 73
XRES parameter 73, 77, 391
document templates
security 387, 388
domain 396
DSN3SATH sample 325
DSN3SSGN sample 329
dynamic parse validation routines 29

E

EBCDIC, for PEM userids and passwords 252
ECICSDCT general resource class
defining profiles 80
enabling a security policy 379
encryption
data sets 334
public key 292
encryption algorithms 294
ENCRYPTION parameter
and SP800-131A 309
enhanced generic naming
data set profile names 17
SETROPTS EGN command 17
ESDSs, VSAM, access to 39
ESM (external security manager)
EBCDIC for userids and passwords 252
invoking another
overview of interface 212
RACROUTE macros 212
PassTickets 6, 54
sample configuration 244
sign-on data from CICS to PEM client 254
user profile 255
using RACF
controlling CICS security 205
creating profiles 167, 176
ESMEXITS, system initialization parameter 48
evaluation sequence, security 208
example tasks
security 213
examples

examples *(continued)*
attach-time security 426
RACF commands 425, 426
surrogate security checking 425
EXCI security 277
EXEC CICS commands
QUERY SECURITY 5
QUERY SECURITY command 131
EXEC CICS LINK command
DFHAPPL profile definition 338
security checking 338
EXEC CICS QUERY SECURITY 340
EXEC CICS SET TRQUEUE ATIUSERID 115
EXEC CICS START USERID 340
EXEC CICS VERIFY PASSWORD 340
exits
ESM, accessing CICS-related information 159
RACF user exit parameter list 159
explicit sign-on 55
external call interface 277
external CICS interface (EXCI)
CALL interface
DFHAPPL profile definition 338
security 337
External CICS interface (EXCI) and surrogate checking 116
external security manager (ESM)
invoking another
overview of interface 212
RACROUTE macros 212
using RACF
controlling CICS security 205
creating profiles 167, 176
EXTRACT CERTIFICATE command 383
EYUCOVE 173
EYUCOVI 173
EYULOG 173
EYUWREP 173
EYUWUI 173

F

FACILITY class profiles, defining 338
FACILITY general resource class 21
FCICSFCT general resource class 82
FEPIRESOURCE resource name 118
FIELD general resource class 14, 20
file resource security checking 283
file security
access authorization levels 83
data set profiles 17
defining resource classes 82
FCICSFCT general resource class 82
generic data set profiles 17
HCICSFCT general resource class 82
XFCT parameter 49, 77, 83
files processed by CICS 82
FIPS compliance 309
flows, examples 248
FMH (function management header)
attach 253
attach FMH5 and data
FMH5 attach header 252
user data following 252
function shipping
function shipping *(continued)*
mirror transaction 236, 241, 276
RESSEC operand of DEFINE TRANSACTION 235, 241

G

GCICSTRN general resource class 47, 69, 85
GCPSMOBJ resource group class 20
GDS (generalized data stream)
GDS LL length
variables to pass data
general resource class
APPCLU
defining profiles 225
general resource classes
ACICSPCT 84, 87
APPCLU 20
APPL 20
BCICSPCT 84, 87
CCICSCMD 130
CONSOLE 20
CPSMOBJ 20
CPSMXMP 20
DIGTCERT 20
FACILITY 21, 268
FIELD 14, 20
for protecting CICS resources 19
for protecting resources 20
for protecting system resources 20
JCICSJCT 83
JESSPOOL 21, 46
KCICSJCT 83
LOGSTRM 21
MCICSPPT 88
NCICSPPT 88
OPERCMDS 21, 64
PCICSPSB 93
PROGRAM 21
PROPCNTL 21, 44
PTKTDATA 21
QCICSPSB 93
SCICSTST 89
STARTED 21
SUBSYSNM 21
SURROGAT 21, 45
TERMINAL 21
UCICSTST 89
user-defined 139
VCICSCMD 129
VTAMAPPL 21, 44
general resource profile 10
general resource profiles
refreshing 207
generating and using RACF PassTickets 6
generic connection
note about lack of security checks 337
generic profile 10
generic profiles
SETROPTS command 25
SETROPTS GENERIC 10, 17, 27, 58
generic resource names (z/OS Communications Server)
z/OS Communications Server generic resource 43
generic resource profiles 94, 95

global security parameters 207
goodnight transaction 57
GRANT command 330
group identifier 53
group profile 10
group profiles 16
GROUP special command
  SEARCH command warning 25
group-SPECIAL attribute 8
grouping class, resource 10
GTERMINAL definition 58
GTERMINL definition 58
GTERMINL resource group class 21

H

handshakerole=serverwithclientauth 397
HCICSFCT general resource class 82
HFS file security
  XHFS parameter 77, 91
HTTP client requests
  security 384

I

IBM-supplied classes
  example for files 51
  example for PSBs 51
  example for transactions 51
ICH408I, RACF message 72
ICHERCDE macro 9, 139
ICHRFR01 (RACF router table) 140
ICHRFRTB macro 140
ICHRIN03 started procedures table 170
ICHRIN03, RACF started task table 31
ICHRRCDE (installation-defined class descriptor table) 139
ICHRTX00, MVS router exit 150
IDENTIFY parameter, ATTACHSEC operand 231
identifying remote users 231, 265, 273
in-storage profiles
  and XCMD resource class 130
  GTERMINL profiles 58
  reducing need for 20
  refreshing 16
installation-defined classes
  example for files 52
  example for PSBs 52
  example for transactions 52
  example for user-defined resource 139
intersystem communication (ISC) security
  coding ATTACHSEC 230, 272
intrapartition transient data resources 115
invoking the trust client 420
IPCONNs
  link security 263
IPIC
  bind-time security 261
IPIC connections
  command security 266
  how security is implemented 261
  resource security 266
  transaction security 266
  user security 264
IPIC security
  CRTE 234, 236, 266, 275
  introduction 6
IRR.DIGTCERT.ADD profile 304
ISC over SNA security
  APPC (LU6.2) session security 6
ISC over TCP/IP connections
  command security 266
  resource security 266
  transaction security 266

J

Java security manager 379
java.security.policy 379
JCICSJCT general resource class 83
JES spool protection 46
JESSPOOL general resource class 21, 46
job submission, surrogate 45
journal security
  access authorization levels 83, 86
  defining resource classes 83
  XJCT parameter 49, 77, 83
journals and log streams
  journal access authorization levels 83

K

KCICSJCT general resource class 83
Kerberos
  configuring web services 436
  developing applications 437
key rings 300
keyring
  building 300
KEYRING 392, 396

L

labels, RACF security 26
language segment
  system defaults 15
  user profile 15
LDAP server
  configuring 312
levels, RACF security 26
libraries, CICSPlex SM
  protecting with RACF 163
link security
  for IPCONNs 263
  introduction 222
load libraries, protecting 30
LOCAL parameter, ATTACHSEC operand 231
LOCK command, CEDA 62
log records
  SMF type 80 72, 80
log streams
  authorizing access to 35
  LOGSTRM general resource class 35
logging security events
  QUERY SECURITY 141
  RACF audit messages in SMF 80
  requested by CICS on authorization requests 80

logging security events *(continued)*
sign-on and sign-off activity 61
logon security 337
LOGSTRM general resource class 21
LU6.1 links 239
LU6.1 security 239
LU6.2 (APPC) session security
introduction 6, 224
XAPPC parameter 49, 50, 77, 225
XDB2 parameter 77
LU6.2 security
CRTE 234, 236, 266, 275

M

MAC (message authentication code) 292
maps
security considerations 174
marking a certificate untrusted 305
MCICSPPT general resource class 88
MD5
encryption algorithm 294
members, group
ADDMEM operand to add 59
DELMEM operand to remove 59
message authentication code (MAC) 292
Message Digest
encryption algorithm 294
message handler
invoking trust client 420
messages
authorization failures 72
destination of ICH408I message 72
DFHSNxxxx 61
ICH408I, RACF 72
MINTLSLEVEL/ENCRYPTION 392, 396
mirror transactions
for DPL on LU6.2 237
for DPL on MRO 277
function shipping 236, 241, 276
mixed case
password 67
MIXIDPE parameter, ATTACHSEC operand 231
MQCONN
security 428
MQMONITOR
security 428, 429
MRO (multiregion operation) security
CRTE 234, 236, 266, 275
MRO logon and connect 270
multiple volumes 39
multiregion operation (MRO)
logon and bind time security 337
MVS
library lookaside (LLA) facility 38
password and RACF authorization checking 29
program properties table (PPT) 29

N

National Language Support 15, 65
NATLANG and non-terminal transactions 68
NCICSPPT general resource class 88
NETNAME terminal definition 58
NIST conformance 309
non-terminal security
transactions not associated with terminals 5
Null authentication 339

O

OIDCARD (operator identification card) 5
OPCLASS 12
operator, CICS terminal
example of defining to RACF 67
obtaining data for 65
operator, terminal
data at sign on 66
data for default user 65
OPERCMDS general resource class 21
OPIDENT 12
OPPRTY 12
OSGi security 379

P

parameter
authorizing access to CICS region 43
protecting CICS data sets 37
parameters
security
activating 207
checking 207
global 207
PassTicket 341
PassTickets, for signon security 6, 54
password
mixed case 67
password expiry management program DFHWBPW 383, 385
password phrase 55
password phrases
mixed case 67
passwords
8 characters 252
in ESM user profile 255
information provided by APPC PEM 243
updating 243
passwords and password phrases 55
patterns
SAML programs 451
Patterns
SAML programs 449
PCICSPSB general resource class 93
PEM requester
conversation type 252
definition 244
format of user data 252
sign-on completion status values returned by CICS 256
PEM server, CICS
data exceeding maximum buffer size 252
EBCDIC for userids and passwords 252
error status returned 247
format of date and time subfields 255
PROFILE option 252
synclevel 0 252
PERMIT command

PERMIT command *(continued)*
WHEN operand 59
PERSISTENT parameter, ATTACHSEC operand 231
persistent sessions
XRFSOFF operand 12
persistent sessions restart
remaining signed on 12
sign-off 12
persistent verification (PV)
ATTACHSEC-PERSISTENT 248
CONNECTION 247
sign-on successful, example flow 248
sign-on unsuccessful, with PV 248
signed on 248
signed-on-from list 247
signed-on-to list 247
successful sign-on flow 248
unsuccessful sign-on 248
when implementing LU6.2 security 230
PIP (program initialization parameter) data 252
pipeline
provider
configuring for SAML tokens 444
requester
configuring for SAML 446
PKCS (public key cryptography standard) 293
platform
security 176, 202, 217
PLT
post-initialization processing 114
PLT programs 73
PLTPI 73
PLTPISEC, system initialization parameter 48
PLTPIUSR system initialization parameter 48, 114
PLTSD 73
POSIT numbers
installation-defined general resource classes 18, 26
post-initialization processing, surrogate security 114
PREFIX attribute definition 89
prefixing
with SECPRFX 47
preset security sessions 63
preset terminal NATLANG 68
preset terminal security
autoinstall models 63
CEDA LOCK command 62
CEDA transaction 62
controlling definition and installation 62
other considerations 63
restricting batch access to CSD 62
starting tasks at terminals 85
SURROGAT transaction 62
terminal routing 236
transactions not associated with a terminal 72
using MVS system console as CICS terminal 64
problem determination
ATTACH security fields 252
data exceeds maximum buffer size 252
determining userid of CICS region 150
error messages for authorization failures 72
GDS FREE command received 252
ICH408I, RACF message 72
new password ID 252
password not in EBCDIC 252
problem determination *(continued)*
PIP data optional 252
PROFILE option 252
reasons for sign-on failure 247
response to incorrect data format 257
sign-on failure 247
sign-on request formatting errors 257
synclevel 252
transaction ID 252
userid and password of more than 8 characters 252
userid not in EBCDIC 252
process
security of 425, 426
PRODCFT1 282
profile 10
profile, data set 10
profile, discrete 10
profile, general resource 10
profile, generic 10
profile, resource group 10
profile, user 2, 10
profiles
ACICSPCT general resource class 84, 87
BCICSPCT general resource class 84, 87
CCICSCMD general resource class 130, 138
data set 17
DCICSDCT general resource class 80
defining
APPCLU general resource class 225
ECICSDCT general resource class 80
enhanced generic naming 17
FCICSFCT general resource class 82
GCICSTRN general resource class 47, 69, 85
generic 25
generic data set 17
HCICSFCT general resource class 82
JCICSJCT general resource class 83
JESSPOOL 46
KCICSJCT general resource class 83
MCICSPPT general resource class 88
NCICSPPT general resource class 88
PCICSPSB general resource class 93
PROPCNTL 44
QCICSPSB general resource class 93
RALTER command to change 24
RCICSRES resource class 73
RDEFINE command to create 24
RDELETE command to delete 24
refreshing in main storage 18
resource and WARNING option 80
resources, defining generic 94, 95
SCICSTST general resource class 89
SETROPTS command 25, 58
SETROPTS EGN command 17
SURROGAT general resource class 45, 117
TCICSTRN general resource class 47, 69, 85
terminal (PoE), defining 58
transaction and conditional access lists 71
transaction, defining to RACF 71
UCICSTST general resource class 89
USER parameter on CICS JOB statement 32
VCICSCMD general resource class 129
VTAMAPPL 44
WCICSRES grouping class 73

profiles for transient data queues 81
profiles, group 10
PROGRAM general resource class 21
program initialization parameter (PIP) data 252
program properties table (PPT), MVS 29
program security
  XPPT parameter 49, 77, 88
propagation of userid, controlling 44
PROPCNTL general resource class
  defining profiles 44
protecting
  CICS resources 19
  resources 20
  system resources 20
provider pipeline
  configuring
    for SAML tokens 444
proxy authentication 384
PSB security
  access authorization levels 93
  defining resource classes 93
  PCICSPSB general resource class 93
  QCICSPSB general resource class 93
  XPSB parameter 49, 77, 93
PSBCHK parameter 93, 132
PSBCHK, system initialization parameter 48
pthreads 297
PTKTDATA general resource class 21
public key encryption 292, 293
PVDELAY system initialization parameter 232

Q

QCICSPSB general resource class 93
QUERY SECURITY command
  and resource classes 131
  and transaction routing 131
  description 131
  effect of SEC parameter 131
  effect of SECPRFX parameter 131
  how the command works 131
  logging 141
  RESCLASS 138
  RESTYPE 133
  RESTYPE, values returned 137
  specifying user-defined resources 139

R

RACDCERT command 300, 303
RACF
  example commands 425, 426
  external security manager 315
  profile 10
RACF (resource access control facility)
  activating the CICS classes 18
  administration 8
  authorizing CICS users 66
  CICS default user 15
  CICS installation requirements 29
  CICS segment 11
  class descriptor table, ICHRRCDE 139
  data set profiles 17
RACF (resource access control facility) *(continued)*
  defining default CICS userid 34
  defining port of entry profiles 58
  defining your own resource class names 26
  FIELD general resource class 14
  general resource profiles 18
  generic data set profiles 17
  generic resource profiles 94, 95
  group profile 16
  group profiles 16
  language segment 15
  overriding SETROPTS TERMINAL 59
  RACF segment 11
  refreshing resource profiles in main storage 18
  router table, ICHRFR01 140
  security labels 26
  security levels 26
  terminal profiles 58
  undefined terminals 59
  user profiles 11
  with multiple MVS images 29
RACF (Resource Access Control Facility)
  controlling access to resources 176
  defining transactions 170, 171
  exempting items from security checking 206
RACF class
  DSNR 325
RACF commands
  ADDGROUP, example 17
  ADDUSER, example for default CICS userid 34
  CONNECT, example 17
  DELMEM operand 59
  example of ALTUSER command 9
  example of CONNECT command (group-SPECIAL) 10
  PERMIT 24
  RALTER 24, 59
  RDEFINE 24
  RDELETE 24
  REMOVE,example 17
  SEARCH—warning 25
  SETROPTS 17, 25
RACF default resource profiles
  VCICSCMD general resource class 319
RACF definitions for surrogate user checking 117
RACF list of groups option 329
RACF PassTickets 6
RACF profiles
  refreshing 207
RACF Secured Sign-on 341
RACF SPECIAL authority 304
RACFVARS profiles 117
RACLIST 140
RACROUTE macros, for security 212
RALTER command 24
RC2
  encryption algoritm 294
RC4
  encryption algoritm 294
RCICSRES 26
RCICSRES resource class 73
RDELETE command 24
RDO
  restricting use of transaction 62

refreshing RACF profiles 207
remote operators 229, 272
remote user sign-off 231, 265, 273
remote users 229, 272
requester pipeline
  configuring
    for SAML 446
resource access control facility (RACF) 337
Resource Access Control Facility (RACF)
  controlling access to resources 176
  defining transactions 170, 171
  exempting items from security checking 206
Resource and command check cross-reference 96
resource checker (CICS ONC RPC)
  writing 342
resource class
  APPCLU
    defining profiles 225
resource classes, CICSPlex SM
  controlling access to 176
resource definition
  LU6.2 (APPC) session security 226
  resource security 235, 241
  SECURITYNAME option 226
  transaction security 235, 241, 266, 274
  user security in link definitions 230, 272
resource definition parameters
  CMDSEC 130
  RESSEC 78, 94
resource group
  DELMEM operand to remove 59
resource group classes
  GCPSMOBJ 20
  GTERMINL 21
resource group profile 10
resource grouping class 10
resource names, CICSPlex SM 176
resource profiles
  RALTER command to change 24
  RDEFINE command to create 24
  RDELETE command to delete 24
resource security
  access authorization levels, files 83
  access authorization levels, z/OS UNIX files 93
  ACICSPCT general resource class 84, 87
  activating the CICS classes 18
  application programs 88
  auditing 80
  authorization failures 79
  BCICSPCT general resource class 84, 87
  CCICSCMD general resource class 138
  CICS SIT parameters 48
  DCICSDCT general resource class 80
  defining generic profiles 94, 95
  defining profiles for TD queues 81
  defining your own resource class names 26
  document templates 73, 391
  ECICSDCT general resource class 80
  FCICSFCT general resource class 82
  FIELD general resource class 14
  files 82
  for IPIC connections 266
  for ISC over TCP/IP connections 266
  GCICSTRN general resource class 47, 69, 85
resource security *(continued)*
  general checking by CICS and RACF 76
  general resource profiles 18
  HCICSFCT general resource class 82
  HFS files 91
  JCICSJCT general resource class 83
  journals and log streams 83
  KCICSJCT general resource class 83
  level of access required 95
  logging RACF audit messages to SMF 80
  MCICSPPT general resource class 88
  NCICSPPT general resource class 88
  PCICSPSB general resource class 93
  profiles and WARNING option 80
  program specification blocks 93
  QCICSPSB general resource class 93
  QUERY SECURITY command 5, 131
  QUERY SECURITY RESCLASS 138
  RCICSRES resource class 73
  refreshing profiles in main storage 18
  resource definition 235, 241
  RESSEC system initialization parameter 79
  RESSEC transaction resource security parameter 78
  SCICSTST general resource class 89
  TCICSTRN general resource class 47, 69, 85
  temporary storage 89
  transaction routing 235, 275
  transient data queues 80
  UCICSTST general resource class 89
  WCICSRES grouping class 73
  XAPPC parameter 77
  XCMD parameter 77
  XDB2 parameter 77
  XDCT parameter 77
  XFCT parameter 77, 83
  XHFS parameter 49, 77, 90, 91
  XJCT parameter 77, 83
  XPCT parameter 77, 84, 87
  XPPT parameter 77, 88
  XPSB parameter 77, 93
  XRES parameter 49, 73, 77, 391
  XTST parameter 78, 89
  z/OS UNIX files 90
resource-level security 425
resources
  protecting 20
RESSEC 340
RESSEC operand of DEFINE TRANSACTION 235, 241, 266
RESSEC resource security parameter 78
RESSEC, system initialization parameter 48
Rivest
  encryption algoritm 294
routing transaction, CRTE 234, 236, 266, 275
RRCDTE sample job 318

S

SAF (system authorization facility)
  and MVS router 159, 212
  to route requests to RACF 1
SAML
  configuring CICS 443
  developing applications 451
SAML tokens

SAML tokens *(continued)*
expired 447
modifying 447
not yet valid 447
re-signing 447
signing 447
SAML-aware programs
augmenting tokens 452
outbound request 452
sample connection exit routine (DSN3SATH) 325
sample sign-on exit routine (DSN3SSGN) 329
SASS (single address space) 324
SCICSTST general resource class 89
scoping sign-on definition 55
SEC system initialization parameter 340
SEC, system initialization parameter 46
SECPRFX, system initialization parameter 47
Secure Hash
encryption algorithm 294
Secured Sign-on 341
securing transactions and resources 223
security
access to WUI resources 174
alias transaction 388
application-generated responses 388
authentication 383, 384, 387
AUTHTYPE 320
basic authentication 383–385
CICS system 388
command security 317
COVA 172
COVC 172
COVE 172
COVG 172
COVP 172
COVU 172
DB2TRAN resource security 318
defining RACF profiles 318
DFHHTML 173
document templates 387, 388
EYUCOVE 173
EYUCOVI 173
EYULOG 173
EYUWREP 173
EYUWUI 173
identification 383, 384
inbound ports 387
non-terminal 72
of activities
attach-time 426
defined activity userid 425
resource-level 425
of BTS commands 426
of processes
attach-time 426
defined process userid 425
resource-level 425
password expiry management 385
password phrase expiry management 385
port number
security 387
proxy authentication 384
RACF 315
RACF class, DSNR 325
security *(continued)*
resource level 387
resource security 317
SAML 443
SASS 324
sign-on 6, 54
SSL 392
surrogate user checking 320
the CICSESM interface 159
using PassTickets 6, 54
z/OS UNIX files 387, 388
z/OS UNIX System Services 387, 388
security categories 26
security checking
controlling CICS 205
ESM interface 212
evaluation sequence 208
exempting items 206
parameters 207
with another ESM 212
with RACF 163
security classification of data and users 26
security for web services 405
security handler
writing your own 420
security labels 26
security levels 26
security manager
applying a security policy 379
enabling a security policy 379
security profiles
refreshing 207
security profiles, RACF
creating 167
views protected by 179
security rebuild 16, 228
security tasks, example 213
security token extensions
configuring provider pipeline 444
configuring requester pipeline 446
security token of JES spool files 46
Security Token Service
trust client interface 409
Security Token Service configuration file 448
security, DBCTL
PSB authorization checking by CICS 334
security, surrogate 320
SECURITYPREFIXID 282
segment
CICS 11
data for terminal user 14
LANGUAGE 15
RACF 11
server program
security considerations 337
session key 224
session security 224
session segment 225
SETROPTS command
CLASSACT option 140
generic data set profiles 17
GENERIC option 140
generic terminal profiles 58
generic user profiles 27

SETROPTS command *(continued)*
RACLIST option 140
REFRESH option 140
SETROPTS GENERICOWNER command 10
SHA
encryption algorithm 294
shared data tables
bind security 280
CONNECT security checks 280
file security 280
security checking 279
server authorization security check 279
sign-off
after persistent sessions restart 12
after XRF takeover 12
logging activity 61
process 56
sign-off process 56
sign-on
after persistent sessions restart 12
after XRF takeover 12
logging activity 61
unsuccessful, example flow 248
user data for terminal user 66
sign-on exit routine 329
sign-on requester transaction
input data required by CICS PEM server 253
SNA service transaction program name 253
sign-on security 6, 54
signature verification
SAML 447
signon requester transaction
ATTACH security fields 252
data exceeds maximum buffer size 252
EBCDIC for userids and passwords 252
new password ID 252
permitted userid and password length 252
PIP data optional 252
PROFILE option 252
synclevel 0 252
X'06F3F0F1', transaction ID 252
simulated CICS security 205
single address space (SASS) 324
SIOCTTLSCTL ioctl 392, 396
SIT parameters, CICS
security-related 207
skew time
changing 447
setting 447
SMF (System Management Facility) 8, 61
SNA LU
SNA ACB access 44
SNA service transaction program name for sign-on transaction program 253
SNSCOPE sign-on operand 48
SOAP message
encrypting 411
signing 410
source libraries, protecting 30
SP800-131A 309
specific connection
MRO logon security checks 337
spool files, security token 46
SPOOLOPEN commands 46
SQL
dynamic 332
static 332
SSL
client certificate authentication 383, 384
for CICS as an HTTP client 392
SSL connection improvements 297
SSL pool 297
SSL(ATTLSAWARE) 397
start transaction
started transactions 114
STARTED general resource class 21
started jobs
defining CICS region userid 31
started task
and RACF userid 11
authorizing CICS procedures 31
started transaction security 84, 87
STS configuration file 448
sts-config.xml 448
sts.xml 448
sts.xsd 448
SUBSYSNM general resource class 21
successful signon
new password 257
PEM client to CICS PEM server 248
response to correct sign-on data 257
response to incorrect data format 257
successful sign-on 248
successful sign-on with PV 248
unsuccessful sign-on 248
unsuccessful sign-on with PV 248
SURROGAT general resource class 21, 45, 62, 117
SURROGAT transaction 62
surrogate authority, querying a user's 141
surrogate job submission
to JES internal reader 45
surrogate security checking 425
surrogate terminal 236
surrogate user
authorizing CICS region userid as 45
surrogate user security
checking 113
post-initialization processing 114
RACF definitions 117
RACF definitions examples 117
SURROGCHK parameter 116
symmetric encryption 293
system data set
authorizing access to 36
generic profiles needed 36
levels of access to 36
protecting 30
system initialization parameters, CICS
CMDSEC 48, 130
DFLTUSER 48
ESMEXITS 48
PLTPISEC 48
PLTPIUSR 48
prefixing CICS resource names 47
PSBCHK 48, 93, 132
resource security 48
RESSEC 48
SEC 46

system initialization parameters, CICS *(continued)*
SEC with QUERY SECURITY 131
SECPRFX 47
SECPRFX with QUERY SECURITY 131
SNSCOPE 48
XAPPC 49, 50, 77, 225
XCMD 49, 77, 129
XDB2 49, 77
XDCT 49, 77, 80
XFCT 49, 77, 83
XHFS 49, 50, 77, 90, 91
XJCT 49, 77, 83
Xname parameters 131
XPCT 49, 77, 84, 87
XPPT 49, 77, 88
XPSB 49, 77, 93
XRES 49, 73, 77, 391
XTRAN 70
XTST 49, 78, 89
XUSER 49, 50
System Management Facility (SMF) 8, 61
system resources
protecting 20
system security
CICS installation requirements 29
system-SPECIAL attribute 8
systems network architecture (SNA) session security 221

T

tasks, example
security 213
TCICSTRN general resource class 47, 69, 85
TCP/IP connections
bind-time security 261
TCPIPSERVICE 397
TCPIPSERVICE resource definition
security 387
SSL
URIMAP resource definition 387
temporary storage
access authorization levels 90
authorizing access to named counter servers 41
authorizing access to the named counter pools 41
authorizing access to the TS pools 39
authorizing access to TS servers 40
defining resource classes 89
SCICSTST general resource class 89
UCICSTST general resource class 89
terminal
emulators 437
TERMINAL definition 58
TERMINAL general resource class 21
terminal security
autoinstall models 63
CEDA LOCK command 62
controlling access 57
example of defining users to RACF 67
identifying users 53
obtaining CICS-related data for a user 65
overriding SETROPTS TERMINAL 59
preset 5, 61
sign-on 55
terminal profiles 58
terminal security *(continued)*
terminals in TCT 63
undefined terminals 59
universal access authority 59
user 5
using MVS system console as CICS terminal 64
XTRAN 49
terminal user security 5
terminals defined in TCT 63
Terminals, defining individual profiles 58
time subfields, format 255
TIMEOUT 12
TLS12 309
trace points 396
transaction attach security
CICS parameters controlling 69
processing when SEC=YES and XTRAN=YES 70
transaction initiation 234, 241, 266, 274
transaction routing and QUERY SECURITY 131
transaction security
access authorization levels 86
conditional access lists 71
CRTE 234, 236, 266, 275
defining profiles to RACF 71
for IPIC connections 266
for ISC over TCP/IP connections 266
resource definition 235, 241, 266, 274
started transactions 69, 84, 87
transaction-attach security 69
transactions started without terminals 85
XJCT parameter 84, 87
XPCT parameter 49, 77
XPCT-checked transactions 84, 87
XTRAN system initialization parameter 70
transactions
in a CMAS
defining to RACF 170
in a MAS
defining to RACF 171
Kerberos 437
transient data
access authorization levels 81
CICS-required transient data queue resource definitions 82
security considerations 80
transient data trigger-level transactions 115
trigger level transactions
default security for 35, 115
specifying security for 72, 115
Triple DES
encryption algorithm 294
trust client
interface 409
invoking 420
TSO command
refreshing using TSO command 18
TSO commands and security processing 140

U

UACC 71
UCICSTST general resource class 89
universal access 71
UNIX authentication 339

untrusted certificate, marking 305
URIMAP definitions
  surrogate security 116
URP_DISASTER response
  to resource checker 343
URP_EXCEPTION response
  to resource checker 343
URP_INVALID response
  to resource checker 343
URP_OK response
  to resource checker 343
user data
  attach FMH5 and data
  format 252
  GDS LL length
  SFL1 and SFL2 lengths
  TP LL length
user exits
  ICHRTX00 MVS router exit 150
  RACF parameter lists 159
user ID
  security 387, 388
  security checking with CRTE 234, 236, 266, 275
USER parameter on CICS JOB statement 32
user profile 2, 10, 255
user profiles
  in RACF 11
  refreshing 207
  with ESM 255
user security
  CICS default user 15
  for IPIC connections 264
  transaction routing 235, 275
  user profiles 11
user-defined classes 139
userid
  ADDUSER to add default CICS userid 34
  controlling propagation of 44
  default 48
  defining CICS default user 34
  defining for CICS 32
  DFLTUSER parameter 48
  non-terminal started transaction 85
  of CICS region as security token 46
  surrogate job submission 45
userid of non-terminal started transaction 85
userid on Db2 AUTHID and COMAUTHID parameters 116
userid on URIMAP resource definitions 116
userid passed as parameter on EXCI calls 116
userids used in CICS 2
users, CICS 2
USRDELAY, system initialization parameter 273

V

validate 443
validating 443
VCICSCMD general resource class 129, 319
VERIFY parameter, ATTACHSEC operand 231
verifying
  signatures 447
verifying remote users 232
views
  security considerations 174
views protected by security profiles 179
VSAM data sets, and BWO 39
VSAM data sets, and Dynamic Volume Count 39
VSAM ESDSs, access to 39
VTAMAPPL general resource class
  defining profiles 44

W

WARNING option 80
WCICSRES 26
WCICSRES grouping class 73
web services
  configuring
    Kerberos 436
Web Services Security (WSS) 405, 413, 417
WHEN operand of PERMIT
  WHEN operand 59

X

XAPPC, system initialization parameter 49, 50, 77, 225
XCICSDB2 general resource class 318
XCICSDB2 member class 318
XCMD system initialization parameter 340
XCMD, system initialization parameter 49, 77, 129
XDB2, system initialization parameter 49, 77
XDCT, system initialization parameter
  considerations for triggered transactions 82
XFCT, system initialization parameter 49, 77, 83
XHFS, system initialization parameter 49, 50, 77, 90, 91
XJCT, system initialization parameter 49, 77, 83
XPCT-checked transaction security 84, 87
XPCT, system initialization parameter 49, 77, 84, 87
XPPT system initialization parameter 340
XPPT, system initialization parameter 49, 77, 88
XPSB, system initialization parameter 49, 77, 93
XRES, system initialization parameter 49, 73, 77, 391
XRF (extended recovery facility)
  FORCE operand 12
  NOFORCE operand 12
  remaining signed on after takeover 12
  sign-off after takeover 12
  XRFSOFF operand 12
XTRAN, system initialization parameter 49
XTST, system initialization parameter 49, 78, 89
XUSER system initialization parameter 340
XUSER, system initialization parameter
  resource security
    XUSER parameter 78
  system initialization parameters, CICS
    XUSER 78

Z

z/OS Communications Server
  generic resource names 43
z/OS Connect
  Security 421
z/OS UNIX file security
  access authorization levels 93
  XHFS parameter 90
z/OS UNIX files

z/OS UNIX files *(continued)*
security 387, 388
z/OS UNIX Systems Services
security 387, 388
ZCICSDB2 grouping class 318