CICS Transaction Server for z/OS
Version 4 Release 2

External Interfaces Guide

CICS Transaction Server for z/OS
Version 4 Release 2

External Interfaces Guide

IBM

Contents

What this manual is about

This manual documents intended Programming Interfaces that allow the customer to write programs to obtain the services of Version 4 Release 2.

This manual describes how you can make the CICS® transaction processing services of CICS Transaction Server for z/OS® available to a variety of external users.

How to use this manual

Read Part 1, "Overview of CICS external interfaces," on page 1 for planning information, and for guidance about which other parts of the manual to consult.

What you need to know to understand this manual

This manual assumes that you are familiar with CICS, either as a system administrator or as a system or application programmer. Some parts of the manual assume additional knowledge about CICS and other products.

Notes on terminology

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

Changes in CICS Transaction Server for z/OS, Version 4 Release 2

For information about changes that have been made in this release, please refer to *What's New* in the information center, or the following publications:

- *CICS Transaction Server for z/OS What's New*
- *CICS Transaction Server for z/OS Upgrading from CICS TS Version 4.1*
- *CICS Transaction Server for z/OS Upgrading from CICS TS Version 3.2*
- *CICS Transaction Server for z/OS Upgrading from CICS TS Version 3.1*

Any technical changes that are made to the text after release are indicated by a vertical bar (|) to the left of each new or changed line of information.

Part 1. Overview of CICS external interfaces

CICS provides a number of interfaces which make transaction processing services available to a variety of external users.

Chapter 1. Interfaces to CICS transactions and programs

You can use many types of external requests to run transactions and program in CICS.

WebSphere® MQ users
WebSphere MQ users can use the CICS 3270 bridge to access CICS transactions. See Part 2, "Bridging to 3270 transactions," on page 15 and *CICS integration with WebSphere MQ.*

MVS™ applications
Applications running in MVS address spaces can use the External CICS Interface (EXCI) to access CICS programs. See Part 3, "External CICS Interface," on page 111.

ONC RPC clients
ONC RPC clients can use CICS ONC RPC support to access CICS programs. See Part 4, "CICS ONC RPC support," on page 253

DCE RPC clients
DCE RPC clients use the Application Support (AS) server to access CICS programs. See Part 5, "Using CICS as a DCE server," on page 341.

The following types of external requests are described in other books:

User socket applications
User socket applications can use the CICS Sockets feature of CICS Transaction Server. See *z/OS Communication Server: IP Configuration Guide.*

Web browsers
Web browsers can use a variety of methods:

    CICS Web support
    The CICS Web support is a CICS-provided facility for supporting Web browsers. See theConfiguring CICS web support components in the Internet Guide

    IBM WebSphere
    The IBM WebSphere Application Server for z/OS is an MVS application that supports Web browsers and routes their requests into CICS.

    CICS Transaction Gateway
    The CICS Transaction Gateway is a workstation application that can accept requests from Web browsers and route them into CICS. It uses the EXCI and ECI.

JVM applications
Java Virtual Machine applications can use a local gateway connection that uses the EXCI to pass requests to CICS. See *Java Applications in CICS.*

Java-enabled Web browsers
Java-enabled Web browsers can use applets that communicate with CICS. Writers of applets can use CICS-provided Java classes to construct external call interface (ECI) and external presentation interface (EPI) requests. The Web browsers communicate with Web servers, and with the CICS Transaction Gateway.

CICS client applications
CICS client applications use a distributed CICS client (either CICS Transaction Gateway or the CICS Universal Client) and the ECI or the EPI. See the *CICS Transaction Gateway: Programming Guide*.

CICS programs
Programs running in CICS Servers on any platform can use **EXEC CICS LINK** to call a CICS program, or can use transaction routing to send transaction requests to CICS Transaction Server. Programs running in CICS Transaction Server can use the CICS front end programming interface (FEPI) to start transactions in the same or another instance of CICS Transaction Server. See FEPI overview in the FEPI Guide.

Telnet clients
Telnet clients can use TN3270 to start transactions.

3270 users
Users of the IBM 3270 Display System can start transactions. This is the most familiar method of introducing work to CICS Transaction Server.

Figure 2 on page 5 shows the principal ways of using CICS transaction processing services from outside CICS.

Key to figure 2

TC	= Terminal Control
TR	= Transaction Routing
DPL	= Distributed Program Link
EXCI	= EXternal CICS Interface
ECI	= External Call Interfaces
EPI	= External Presentation Interface
CWP	= CICS WebServer Plugin

= Sources of external requests

= Targets of external requests

= CICS provided interfaces

= CICS components

= Other product components
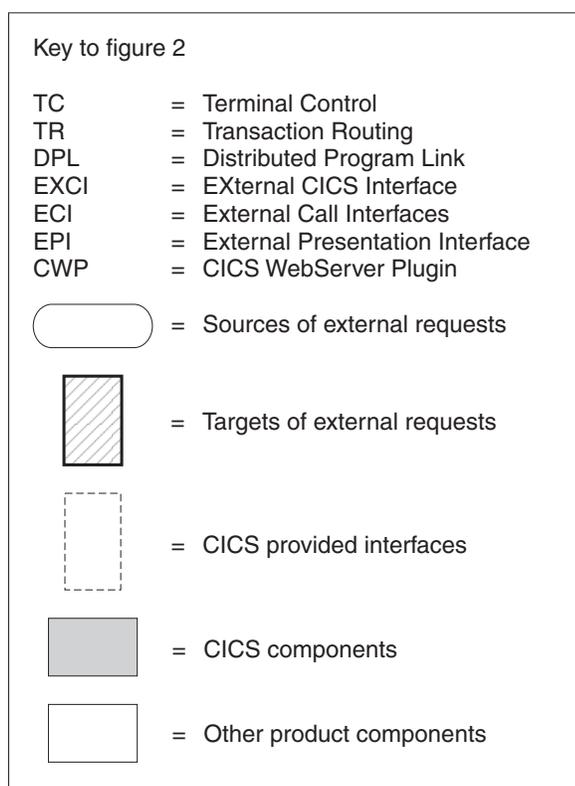
Figure 1. Key for External Interface diagram
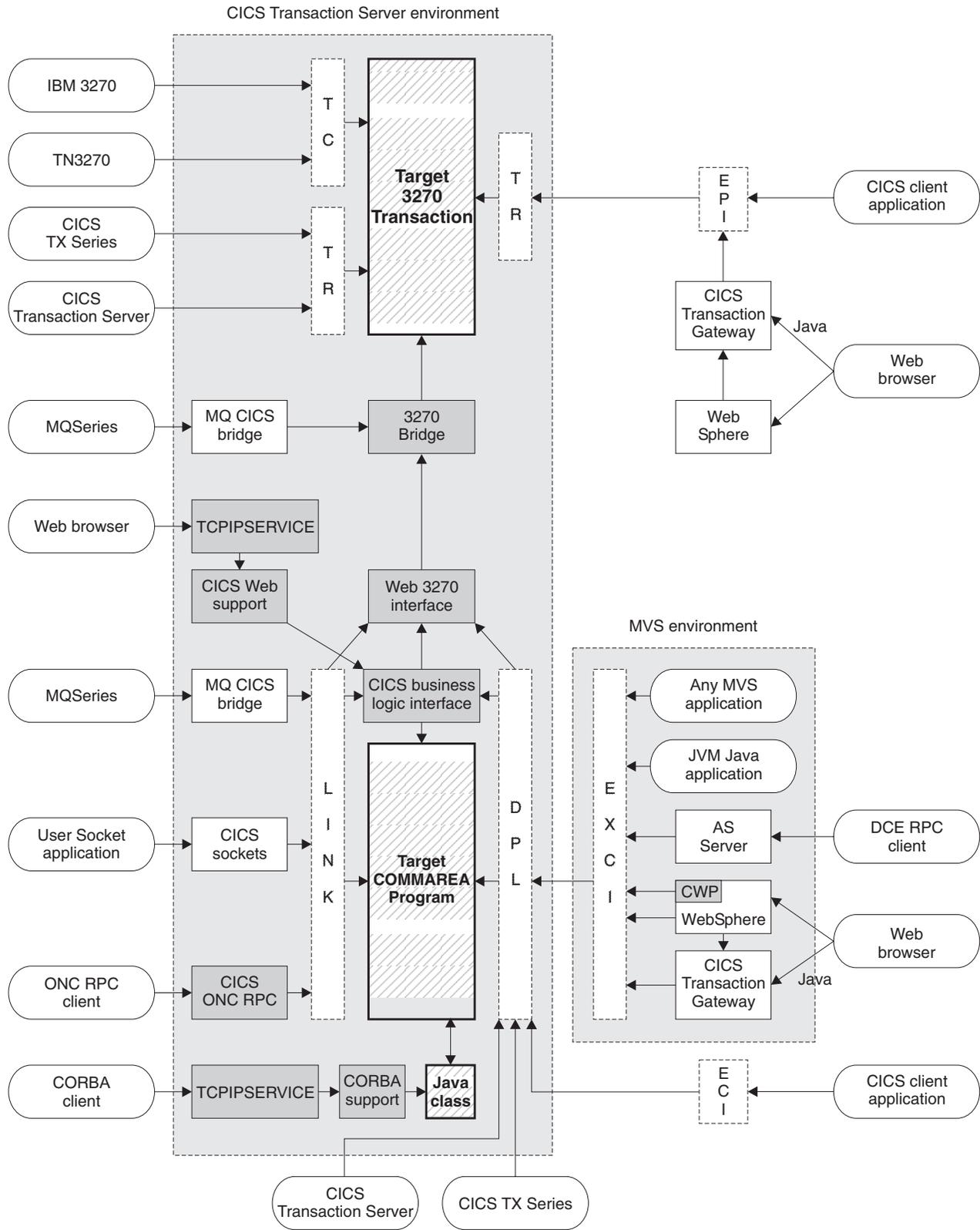
Figure 2. Client access to existing business logic

The client/server model

Client/server is a model of interaction in which a program sends a request to another program and awaits a response. The requesting program is called a client; the answering program is called a server. Although the client/server model can be used between programs in a single computer, the term typically refers to a network. In a network, the model provides a convenient way to interconnect programs that are distributed across different locations.

In CICS, a client is the source of an external request, and the server is the CICS program that services the request. A client can be a program on another platform that is connected to CICS over a network, or a program on another CICS region, connected with interregion communication (IRC).

CICS (or another product) provides a transport-specific listener (a long-running task) that starts another task (a facilitator such as an **alias** or a **mirror**), to process the incoming request. The facilitator uses CICS services to access the application.

The priorities of different alias transactions can be adjusted to determine the service that a client request receives. There must be enough free tasks to service the alias transactions as they are started by the listener. The CICS programs that service the client requests are subject to contention for resources in the CICS system, and to transmission delays if they are remote from the CICS system, or if they request the use of remote resources by function shipping or distributed program link.

The CICS server is independent of the application model (2/3-tier, 2/3 platforms). The listener/facilitator deals with the different transports used and sets the rules for which programming models are supported.

Distributed computing

Distributed computing involves the cooperation of two or more machines communicating over a network. The machines participating in the system can range from personal computers to super computers; the network can connect machines in one building or on different continents.

The main benefit of distributed computing is that it enables you to optimize your computing resources for both responsiveness and economy. For example, it enables you to:

- Share the cost of expensive resources, such as a typesetting and printing service, across many desktops. It also gives you the flexibility to change the desktop-to-server ratio, depending on the demand for the service.
- Allocate an application's presentation, business, and data logic appropriately. Often, the desktop is the best place to perform the presentation logic, as it is nearest to the end user and can provide highly responsive processing for such actions as drag and drop GUI interfaces.

  Conversely, you may feel that the best place for the database access logic is close to the actual storage device - that is, on an enterprise or departmental server. The most appropriate place for the business logic may be less clear, but there is much to be said for placing this too in the same node as the data logic, thus allowing a single desktop request to initiate a substantial piece of server work without intervening network traffic.

  Distributed computing enables you to make such trade-offs in a flexible way.

Along with the advantages of distributed computing come new challenges. Examples include keeping multiple copies of data consistent, keeping clocks in individual machines synchronized, and providing network-wide security. A system that provides distributed computing support must address these new issues.

CICS supports distributed computing and the client/server model by means of:

Internet Inter-Orb Protocol (IIOP)
> CORBA clients can access CICS Java servers using IIOP.

Distributed Computing Environment (DCE)
> The remote procedure call model implemented by the Open Software Foundation's DCE is supported in CICS.

Distributed program link (DPL)
> This is similar to a DCE remote procedure call. A CICS client program passes parameters to a remote CICS server program and waits for the server to send data in reply. Parameters and data are exchanged by means of a communications area.

The external CICS interface (EXCI)
> An MVS client program links to a CICS server program. Again, this is similar to a DCE RPC.

The external call interface (ECI)
> The ECI enables CICS Transaction Server for z/OS server programs to be called from client programs running on a variety of operating systems. For information about CICS Clients, see the *CICS Transaction Gateway: Programming Guide*.

Function shipping
> The parameters for a single CICS API request are intercepted by CICS code and sent from the client system to the server. The CICS mirror transaction in the server executes the request, and returns any reply data to the client program. This can be viewed as a specialized form of remote procedure call.

Asynchronous transaction processing
> A CICS client transaction uses the EXEC CICS START command to initiate another CICS transaction, and pass data to it. The START request can be intercepted by CICS code, and function shipped to a server system. The client transaction and started transactions execute independently. This is similar to a remote procedure call with no response data.

Distributed transaction processing
> A program in the client system establishes a conversation with a complementary program in the server, and exchanges messages. The programs may use the APPC protocols.

Transaction routing
> Terminals owned by one CICS system to run transactions owned by another.

The CICS family of products runs on a variety of operating systems, and provides a standard set of functions to enable members to communicate with each other. For information about the CICS family, see the *CICS Family: Interproduct Communication* manual.

Security support

CICS Transaction Server for z/OS supports: a single network signon, and authentication of the client system through bind-time security.

Single network signon is supported through the ATTACHSEC option of the DEFINE CONNECTION command.

RACF® or an equivalent security manager provides mechanisms similar to the DCE access control lists and login facility.

There is no CICS concept similar to the DCE Directory Service. In all the above scenarios the client environment must know which server CICS system to communicate with. This is normally done by specifying the name of the required remote CICS system in the definition of the relevant remote CICS resource, or in the client application program.

TCP/IP protocols

TCP/IP is a communication protocol used between physically separated computer systems. TCP/IP can be implemented on a wide variety of physical networks.

TCP/IP is a large family of protocols that is named after its two most important members, Transmission Control Protocol and Internet Protocol. Figure 3 shows the TCP/IP protocols used by CICS ONC RPC in terms of the layered Open Systems Interconnection (OSI) model. For CICS users, who may be more accustomed to SNA, the left side of Figure 3 shows the SNA layers that correspond very roughly to the OSI layers.

SNA		OSI	TCP/IP family	
Application	7	Application	RPC	
Presentation	6	Presentation	XDR	
Data flow	5	Session	(empty)	Sockets interface
Transmission	4	Transport	TCP or UDP	
Path control	3	Network	IP	
Data link	2	Data link	subnetwork	
Physical	1	Physical		
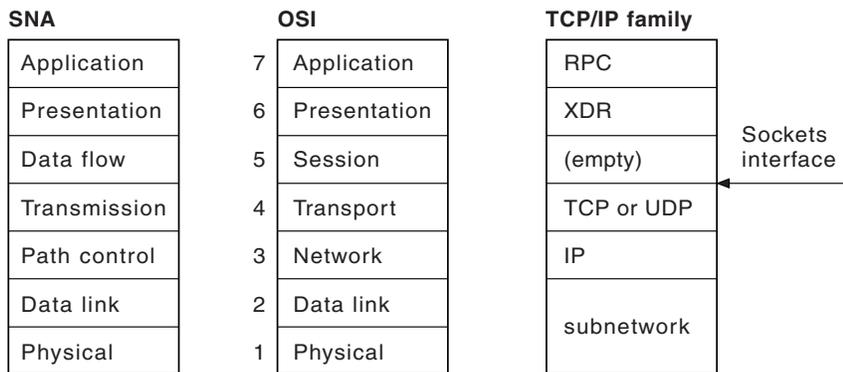
Figure 3. TCP/IP protocols compared to the OSI and SNA models

The protocols used by TCP/IP are shown in the right-hand box in Figure 3.

Internet Protocol (IP)
In terms of the OSI model, IP is a network-layer protocol. It provides a *connectionless* data transmission service, and supports both TCP and UDP. Data is transmitted link by link; an end-to-end connection is never set up during the call. The unit of data transmission is the *datagram*.

Transmission Control Protocol (TCP)
In terms of the OSI model, TCP is a transport-layer protocol. It provides a *connection-oriented* data transmission service between applications, that is, a connection is established before data transmission begins. TCP has more error checking that UDP.

User Datagram Protocol (UDP)
UDP is also a transport-layer protocol and is an alternative to TCP. It provides a connectionless data transmission service between applications. UDP has less error checking than TCP. If UDP users want to be able to

respond to errors, the communicating programs must establish their own protocol for error handling. With high-quality transmission networks, UDP errors are of little concern.

ONC RPC and XDR

XDR and ONC RPC correspond to the sixth and seventh OSI layers.

Sockets interface

The interface between the fourth and higher layers is the *sockets* interface. In some TCP/IP implementations, the sockets interface is the API that customers use to write their higher-level applications.

TCP/IP internet addresses and ports

TCP/IP provides for process-to-process communication, which means that calls need an addressing scheme that specifies both the physical host connection (Host A and Host B in Figure 4) and the software process or application (C, D, E, F, G, and H). The way this is done in TCP/IP is for calls to specify the host by an *internet address* and the process by a *port number*. You may find internet addresses also referred to elsewhere as internet protocol (IP) addresses or host IDs.
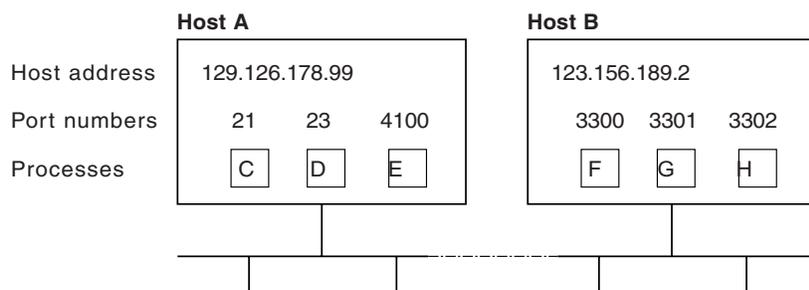
Figure 4. How applications are addressed

IP addresses

Each server or client on a TCP/IP internet is identified by a numeric IP (Internet Protocol) address. The two types of IP address are the IPv4 (IP version 4) address and the IPv6 (IP version 6) address.

IP addresses are managed and allocated to users by the Internet Assigned Numbers Authority (IANA) and its delegates. The internet address specifies both the network and the individual host. This specification varies with the size of the network.

IPv6 addresses

IPv6 addresses are 128-bit addresses, usually expressed in hexadecimal notation:

```
IP address in hexadecimal notation :   '00010022033344440000000abc0def0'x
Halfword 0:  0001 hexadecimal
Halfword 1:  0022 hexadecimal
Halfword 2:  0333 hexadecimal
Halfword 3:  4444 hexadecimal
Halfword 4:  0000 hexadecimal
Halfword 5:  0000 hexadecimal
Halfword 6:  abc0 hexadecimal
Halfword 7:  def0 hexadecimal
IP address in colon hexadecimal notation: 1:22:333:4444::abc0:def0

IP address in hexadecimal notation :   '000000000000000000000ffff01020304'x
Halfword 0:  0000 hexadecimal
```

```
Halfword 1:  0000 hexadecimal
Halfword 2:  0000 hexadecimal
Halfword 3:  0000 hexadecimal
Halfword 4:  0000 hexadecimal
Halfword 5:  ffff hexadecimal
Halfword 6:  0102 hexadecimal
Halfword 7:  0304 hexadecimal
IP address in colon hexadecimal notation: ::ffff:1.2.3.4 or ::ffff:0102:0304
```

The address consists of eight halfword fields. Zeros are treated in the following
ways in the address output:

- If a field contains leading zeros, they are ignored; for example, 0001 is
  represented as 1
- If one or more consecutive fields in the address contain the value 0000, these
  fields are expressed using the notation ::

  For example, 000000000000ffff is represented as ::ffff

  The :: substitution is used once only in an address, to avoid confusion in
  calculating how many fields were substituted.

IPv4 addresses

IPv4 addresses are 32-bit addresses, usually expressed in dotted decimal notation:

```
IP address in hexadecimal notation :   '817EB263'x
Byte 0:  81 hexadecimal = 129 decimal
Byte 1:  7E hexadecimal = 126 decimal
Byte 2:  B2 hexadecimal = 178 decimal
Byte 3:  63 hexadecimal =  99 decimal
IP address in dotted decimal notation: 129.126.178.99
```

In this example, 129.126 specifies the network and 178.99 specifies the host on
that network.

Port numbers (for servers)

An incoming connection request specifies the server that it wants by specifying the
server's port number.

For instance, in Figure 4 on page 9, a call requesting port number 21 on host A is
directed to process C.

Well-known ports identify servers that carry standard services such as the File
Transfer Protocol (FTP) or Telnet. The same service is always allocated the same
port number, so, for example, FTP is always 21 and Telnet always 23. Networks
generally reserve port numbers 1 through 255 for well-known ports.

Port numbers (for clients)

Client applications must also identify themselves with port numbers so that server
applications can distinguish different connection requests.

The method of allocating client port numbers must ensure that the numbers are
unique; such port numbers are termed *ephemeral port numbers*. For example, in
Figure 4 on page 9, process F is shown with port number 3300 on host B allocated.

ONC and DCE concepts

ONC (Open Network Computing) RPC (Remote Procedure Call) is an open source RPC framework developed by Sun Microsystems. DCE (Distributed Computing Environment) is an architecture defined by the Open Software Foundation (OSF). Both technologies support client-server applications in heterogeneous distributed environments.

DCE RPC is different from ONC RPC in many ways. For example, DCE RPC does not limit the number of parameters on the call, whereas an ONC RPC call is limited to one input and one output parameter (but these may be structures that contain many fields, including pointers to other data).
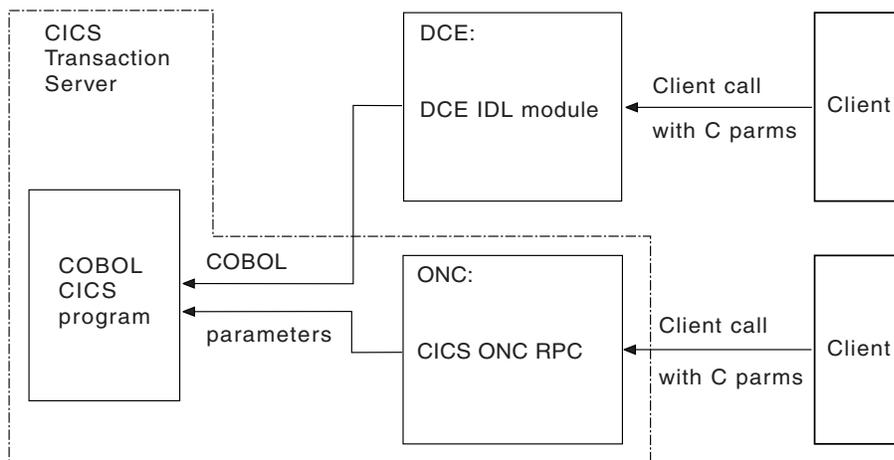
Figure 5. Remote procedures provided for DCE RPC and ONC RPC

Figure 5 shows how the two CICS RPC implementations provide the same function.

You provide a definition of the client's parameter list in the interface definition language (IDL) provided as a part of DCE RPC. The DCE IDL module maps the incoming parameters into a CICS communication area, so the communication area format is defined by the client's parameter list.

CICS ONC RPC CICS programs can be written in any CICS-supported programming language, and the conversion from client format to communication area is done by the **Decode** function of the converter. With ONC RPC you get more flexibility, but you have more work to do.

CICS programs that are used as servers for DCE RPC clients can also be used as servers for ONC RPC clients. You need to write a **Decode** function that converts the incoming data structure into the predefined communication area, and converts the incoming data from C types to COBOL types.

DCE

DCE provides a high-level, coherent environment for developing and running applications on a distributed system.

The DCE components fall into two categories: tools for developing distributed applications and services for running them. The tools, such as Remote Procedure Calls and Threads, assist in the development of an application. The services, like

the Directory Service, Security Service, and Time Service, provide support in a distributed system that is analogous to the support an operating system provides in a centralized system.

DCE includes management tools for administering all of the DCE services and many aspects of the distributed environment itself.

DCE is oriented towards heterogeneous rather than homogeneous systems. The DCE architecture allows for different operating systems and hardware platforms. Using DCE, a process running on one computer can interoperate with a process on a second computer, even when the two computers have different hardware or operating systems.

EXCI concepts

The external CICS interface makes CICS applications more easily accessible from non-CICS environments.

Programs running in MVS can issue an EXEC CICS LINK PROGRAM command to call a CICS application programs running in a CICS region. Alternatively, the MVS programs can use the CALL interface when it is more appropriate to do so.

The provision of this programming interface means that, for example, MVS programs can:
* Update resources with integrity while CICS is accessing them.
* Take CICS resources offline, and back online, at the start and end of an MVS job. For example, you can:
  – Open and close CICS files.
  – Enable and disable transactions in CICS (and so eliminate the need for a master terminal operator during system backup and recovery procedures).

The external CICS interface opens up a new way to implement client/server applications, where the client program in a non-CICS environment calls a server program running in the CICS address space. The external CICS interface benefits not only TSO and batch applications, but allows you to extend the use of CICS application programs in an open client/server environment.

Although the CICS external interface operates over CICS MRO links, the client program can run on non-MVS platforms, and pass requests to CICS over an open system interface (OSI) using the IBM z/OS Distributed Computing Environment Application Support MVS/ESA CICS feature (DCE AS/CICS). In this way the external CICS interface provides an open interface to a wide variety of other application platforms.

3270 bridge concepts

The 3270 bridge allows you to introduce new GUI front ends to access existing 3270-based CICS applications without modifying them.

This means that you can concentrate your efforts on the new user interfaces and avoid, or at least postpone, rewriting stable mainframe applications. You do not need to restructure your applications to separate the business logic from the presentation logic; the bridge effectively does this for you.

The same applications can be used both by 3270 terminals, and by the new client applications. This allows a phased migration of users from the 3270 applications to the new client applications. Applications written for 3270 terminals can be run on CICS systems without the z/OS Communications Server.

The bridge can process commands faster than existing front-end methods, such as FEPI and EPI, because the terminal emulation is part of the same CICS transaction. With the START BREXIT bridge mechanism, there is only a single unit of work. This means that the bridge can use a recoverable WebSphere MQ queue. This greatly simplifies recovery.

For BMS user transactions, there is no need to convert BMS data to 3270 format, because the client application receives the BMS Application Data Structure, rather than a 3270 datastream. This provides an easier method for the application programmer to interface with the user transaction compared to FEPI. A utility program (DFHBMSUP) is provided to re-create map source code from existing load modules, so that installations that do not have access to the original source code can still exploit the new ADS descriptor provided by the BMS macros.

The target transaction is unchanged, but because of the way it now executes in the bridge environment, there are some restrictions on what it can do. These restrictions are described in "Link3270 programming considerations" on page 20..

CICS provides two types of 3270 bridge mechanism:

The Link3270 mechanism
> This mechanism provides a simplified interface using LINK, ECI or EXCI. All messages have a fixed format and you are not required to provide any user-written supporting programs.

The START BREXIT mechanism
> This 3270 bridge mechanism requires a *bridge monitor* transaction to initiate the bridge environment by issuing a START BREXIT command, which specifies the target user transaction and also the name of a user-written *bridge exit*. The bridge exit is called to intercept 3270 requests and pass them in the form of messages to the client application. You can write your own bridge exit and also define your own message formats. Bridge exits are provided to support client applications using temporary storage, the web and Websphere MQ as transport mechanisms for requests, using sample message formats.

The START BREXIT mechanism is still supported, and the sample bridge exits are still provided, but it is better to use the simpler Link3270 mechanism, and migrate to it where possible. The START BREXIT is described in the CICS Transaction Server for OS/390®, Version 1 Release 3 *CICS External Interfaces Guide*, which can be found at http://www.ibm.com/software/ts/cics/library/books/os390/ .

The 3270 bridge and FEPI

To help you decide between the 3270 bridge technology and FEPI, the following table summarizes the major characteristics.

Table 1. Comparision between 3270 bridge technology and FEPI

START Bridge	Link3270 Bridge	FEPI
Enabling technology	Enabling technology	An application programming interface

Table 1. Comparision between 3270 bridge technology and FEPI (continued)

START Bridge	Link3270 Bridge	FEPI
Based on application data structure	Based on application data structure	Based on the 3270 data stream
Enables optimization due to integral knowledge of the target	Enables optimization due to integral knowledge of target	Easier to create generic driver (data structure is architected)
Efficient; no terminal control involved	Efficient, no terminal control involved	z/OS Communications Server managed connection between source and target
Single COMMAREA API and user replaceable program	COMMAREA API	Requires system programming and z/OS Communications Server skills
CICS specific: source and target must be in the same region	LINK, DPL, EXCI or ECI interface supported	Ideal for driving remote applications, not just CICS
Driven exit decides method of communication with the client	Client interface is LINK, DPL, EXCI or ECI	Can be freed from the workings of the target; terminal emulation
Knowledge of UOW	Standard CICS LINK coordination	No coordination
Ideal when the routing is done elsewhere	Supports workload balancing	Sysplex support requires three regions

Part 2. Bridging to 3270 transactions

Using the *3270 bridge* you can connect a client application to a 3270-based CICS transaction. In this configuration, the client application takes the place of the 3270 terminal and the terminal end-user.

Chapter 2. Introduction to the 3270 bridge

The 3270 bridge provides an interface so that you can run 3270-based CICS transactions without a 3270 terminal. The 3270 terminal and end-user are replaced by an application program, known as the *client application*.

Commands for the 3270 terminal in the CICS 3270 *user transaction* are intercepted by CICS and replaced by a messaging mechanism that provides a bridge between the client application and the CICS user transaction.

CICS provides two types of 3270 bridge mechanism:

The Link3270 mechanism
> This mechanism provides a simplified interface using LINK, ECI or EXCI. All messages have a fixed format and you are not required to provide any user-written supporting programs. This mechanism supports CICSPlex® SM load balancing; bridge facilities are shared between CICS regions on the CICSplex

The START BREXIT mechanism
> This 3270 bridge mechanism requires a bridge monitor transaction to initiate the bridge environment by issuing a **START BREXIT** command, which specifies the target user transaction and also the name of a user-written bridge exit. The bridge exit is called to intercept 3270 requests and pass them in the form of messages to the client application. You can write your own bridge exit and also define your own message formats. Bridge exits are provided to support client applications using Temporary Storage, the Web and WebSphere MQ as transport mechanisms for requests, using sample message formats. This mechanism is single region only: bridge facilities are local to the region.

The **START BREXIT** mechanism is supported and the sample bridge exits are still provided. However, consider migrating to use the simpler Link3270 mechanism where possible. **START BREXIT** is not documented in this release of CICS . To use it or modify existing implementations, refer to the CICS documentation that is published at http://publibfp.dhe.ibm.com/cgi-bin/bookmgr/BOOKS/dfhjatm6/CCONTENTS.

The Link3270 bridge mechanism

The client application uses the Link3270 bridge to run 3270 transactions by linking to the DFHL3270 program in the router region and passing a COMMAREA that identifies the transaction to be run and contains the data used by the user application.

The response contains the 3270 screen data reply. If the target application used BMS, this is presented in the form of an **application data structure (ADS)**, another name for the symbolic map that is generated by the BMS macros used to define the mapping of the 3270 screen.

The Link3270 bridge is called in the same way for all request mechanisms of the interface: EXEC CICS LINK, the EXternal CICS Interface (EXCI), and the CICS External Call Interface (ECI).

The following flow describing the Link3270 mechanism is shown also in Figure 6:

1. The client application creates a Link3270 request message.
2. The client application issues an appropriate link request (ECI, EXCI or LINK) to the CICS router program DFHL3270, passing the Link3270 message as a COMMAREA. Note that CICS takes care of the codepage conversion if necessary.
3. DFHL3270 dynamically routes the request to the bridge **driver** task, which may be in the same or another CICS region. Load balancing can be implemented in this step.
4. The driver starts the user application (the target 3270 transaction), running in a bridge environment.
5. The response Link3270 message is returned to the client as a COMMAREA.
6. The client application processes the outbound message .
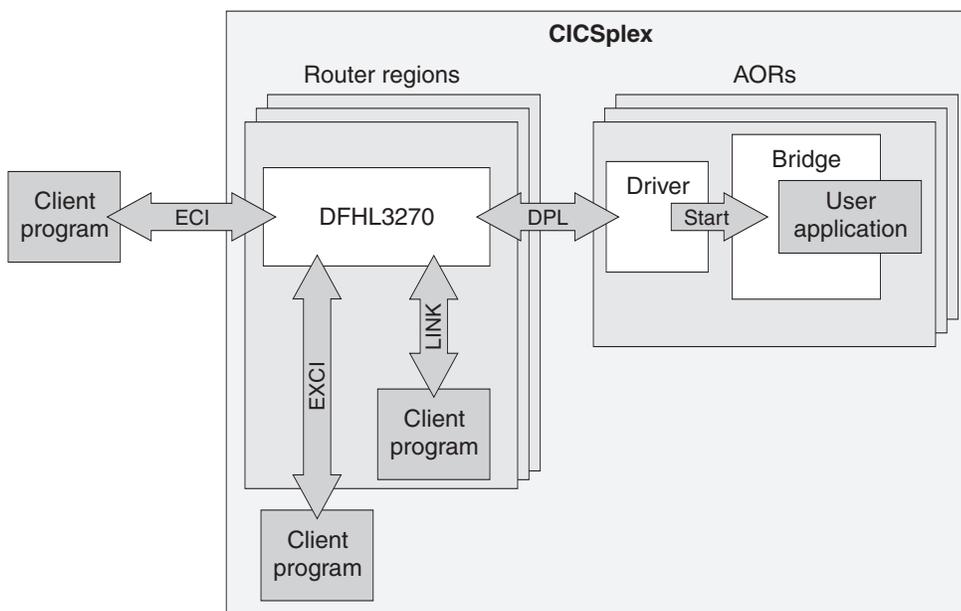
Figure 6. Link3270 request flow

The Link3270 mechanism supports non-conversational, pseudoconversational and conversational applications.

The bridge facility

The 3270 user application was designed to be used with a real 3270 terminal and the CICS commands that it uses assume that a real 3270 exists.

The 3270 bridge mechanism simulates the presence of a real 3270 by providing internal interfaces for a virtual 3270, known as the **bridge facility**.

This replaces the terminal resource definition that you would normally provide for a 3270 application. The bridge facility emulates a real terminal in the following EXEC CICS interfaces:

- ASSIGN
- Terminal control and some of the BMS API

- EIB
- INQUIRE TASK
- INQUIRE TERMINAL

You do not provide a resource definition for the bridge facility, but you can control some of the terminal properties used by providing a 3270 TERMINAL resource definition to be used as a template. The name of this TERMINAL definition, known as the **facilitylike**, is passed to the bridge on the Link3270 call.

See "Defining the bridge facility" on page 48 for further information about the bridge facility.

Lifetime of the bridge facility

When simple transactions are run in **single transaction mode**, the bridge facility is created dynamically by CICS and deleted at the end of the transaction.

In **session mode**, multiple transactions or pseudotransactions can be run using the same bridge facility. In this mode the client application can request creation and deletion of the bridge facility, and can also specify a **keeptime** in BRIH-FACILITYKEEPTIME in the Allocate function. See "Using Link3270 session mode" on page 37.

The maximum keeptime value can be limited by the `BRMAXKEEPTIME` system initialization parameter. Bridge facilities are deleted automatically if they are inactive for the keeptime interval.

For more information about specifying system initialization parameters, see CICS startup in the System Definition Guide.

The application data structure (ADS)

Application data structure (ADS) is another name for the symbolic map that is generated by the BMS macros used to define the mapping of the 3270 screen.

Application data structure (ADS) is another name for the symbolic map that is generated by the BMS macros used to define the mapping of the 3270 screen. For BMS programs, terminal data is passed between the client and the bridge in this format, giving the client application a simplified interface to the terminal data, without the need to understand 3270 data streams.

See "DFHBMSUP" on page 20 for guidance on creating the ADSD if you have no source.

The ADS descriptor (ADSD)

The ADS descriptor allows interpretation of the BMS application data structure (the symbolic map used by your application program for the data in SEND and RECEIVE MAP requests) - without requiring your client program to include the relevant DSECT or copybook at compile time.

The ADS descriptor contains a header with general information about the map, and a field descriptor for every field that appears in the ADS, corresponding to every named field in the map definition macro. It can be located in the mapset from an offset field in DFHMAPDS.

The ADS descriptor is available only if the map load module has been reassembled to include the descriptor, and CICS attempts to locate the descriptor only if the BRIH-ADSDESCRIPTOR indicator is set to BRIHADSD-YES in the Link3270 message header.

DFHBMSUP

If you are unable to reassemble the mapset because you do not have the source, you can use the DFHBMSUP utility to re-create source statements from your mapset load module.

For information about DFHBMSUP, see .

Link3270 programming considerations

The user transaction is unchanged, but because of the way it now executes in the bridge environment, there are some restrictions on what it can do, and some limitations on how it can use the bridge facility, because it is not a real terminal.

You can use the Load Module Scanner utility (described in Operations and utilities overview in the Operations and Utilities Guide), using the supplied table DFHEIDBR, to identify any CICS commands in your program that are not supported by the bridge.

Note: The bridge only supports valid documented CICS API interfaces. If either the application or vendor programs use undocumented interfaces, the results will be unpredictable.

If you are using CICS Service Flow Runtime there are additional restrictions described in Link3270 bridge restrictions.

Abend information
> The bridge facility name is not used as the TERMID in any diagnostic information produced as the result of an abend, except in a transaction dump.

ASSIGN
> If the user transaction issues ASSIGN NETNAME, the value returned is the NETNAME if there is one, or else the TERMID. The name is not visible outside the user transaction, and may contain '}' characters.
>
> You can only use ASSIGN to request information about BMS attributes such as MAPCOLUMN, MAPHEIGHT, MAPLINE, and MAPWIDTH if an ADS descriptor is present in the mapset. See "The ADS descriptor (ADSD)" on page 19.

BMS requests
> The Link3270 bridge supports the following BMS commands. If other BMS functions that require a principal facility are used, they cause the user transaction to abend ABR3.
>
> **RECEIVE commands**
> > - RECEIVE MAP TERMINAL
> > - RECEIVE MAP FROM
> > - RECEIVE MAP MAPPINGDEV
> >
> > **Note:** TERMINAL is implied if neither TERMINAL nor FROM is specified.
> >
> > **SEND commands**
> > - SEND MAP TERMINAL

- SEND TEXT TERMINAL
- SEND TEXT NOEDIT TERMINAL
- SEND TEXT MAPPED TERMINAL
- SEND CONTROL TERMINAL
- SEND MAP SET
- SEND TEXT SET
- SEND TEXT NOEDIT SET
- SEND TEXT MAPPED SET

Note:
1. TERMINAL is implied if none of TERMINAL, SET, or PAGING is specified.

Routing

Routing to real terminals from a transaction running on a bridge facility is supported, but it is not possible to route to a bridge facility, nor to specify a bridge facility as ERRTERM on ROUTE. If ERRTERM without a name is specified on a ROUTE request issued in a bridge environment, the INVERRTERM condition is raised.

PAGING is supported only under routing.

Partitions

Partition related commands and options are supported, but are treated in the same way as they would be for a real terminal that does not support partitions.

SEND PARTNSET

Supported, but the bridge exit is not invoked.

RECEIVE PARTN

Supported; the bridge exit is invoked with bridge exit area command fields set up for a terminal control RECEIVE.

INPARTN

Accepted but ignored; not passed to the bridge exit.

OUTPARTN

Accepted but ignored; not passed to the bridge exit.

ACTPARTN

Accepted but ignored; not passed to the bridge exit.

CICS-supplied transactions

CEDF, CEDX, CSFE, and CSGM cannot run as user transactions.

DB2® authorization check

Do not use the settings AUTHTYPE(TERM) or AUTHTYPE(OPID) in the DB2CONN definition, because these security checks fail in a bridge environment.

External security customization

TERMID, OPID, and TCTUA information is not passed in the DFHXSID parameter list.

Global User Exits

The following global user exits (GLUEs) are **not** driven because the bridge facility is not a real terminal.

XBMIN

    to intercept a RECEIVE MAP request.

XBMOUT

    to intercept a SEND MAP request.

XTCATT

    before a task attach.

XZCATT

    before a task attach (z/OS Communications Server SNA).

XZCIN

    after an input event (z/OS Communications Server SNA).

XZCOUT

    before an output event (z/OS Communications Server SNA).

XZCOUT1

    before a message is broken into RUs (z/OS Communications Server SNA).

The XALTENF and XICTENF exits can be driven if a request is made for a bridge facility. The 'terminal-not-found' condition is raised because the bridge facility is not a real terminal.

The standard user exit parameter list field UEPTERM that points to the TERMID are not set for exits invoked under a bridge task.

ISSUE PASS

ISSUE PASS is not supported and results in an INVREQ.

ISSUE PRINT

ISSUE PRINT is not supported and results in a no-op. A NORMAL condition is returned.

Monitoring

A 3270 bridge transaction identifier is present in monitoring records.

Remote DLI requests

No security check of the PSB against the terminal is done for function-shipped DLI requests.

Security Processing

When a bridge facility is created, it is signed on as a preset USERID terminal, with the client's USERID. As with other preset terminals, the SIGNON and SIGNOFF commands are not permitted, and INVREQ is raised.

The bridge facility is signed off when it is discarded. It remains signed on in session mode until a specific delete facility request is sent, or the keeptime interval expires.

START

The user transaction can issue **EXEC CICS START** requests for its own bridge facility. This allows existing menu-driven and pseudo-conversational applications that use this interface to work in a bridge environment. See "Pseudoconversational transactions" on page 39 for a description of START TERMID where TERMID specifies the bridge facility.

The time delay options, (INTERVAL, TIME, AFTER, AT, HOURS, MINUTES, SECONDS) are not normally used in the bridge environment, but the bridge mechanism uses them to put the STARTs for a particular bridge facility in time order, but the exact delays requested are not implemented. TIME and AT specifications are ignored completely.

Other options on the START command are partly supported :

TERMID

You can specify the name of your own bridge facility for this transaction, or for any real terminal.

USERID

USERID and TERMID are mutually exclusive. The CICS translator rejects START requests with both USERID and TERMID specified.

TRANSID

If the TRANSID cannot be defined as REMOTE, the TERMID will not be found if the request is shipped to a remote system.

SYSID

Routing of START requests is not possible in a bridge environment. This option is not supported, unless the value of the SYSID is the local SYSID. If you specify any other value, the request will be shipped and the TERMID will not be found on the remote system.

NOCHECK

This option only applies to shipped start requests and is ignored.

PROTECT

If you specify the PROTECT option on a START request for a bridge facility, and the starting task abends before taking a syncpoint, the START request is discarded. PROTECT normally delays the starting of the new task until a SYNCPOINT has occurred. This happens automatically for a task issuing a START for its own facility because the START cannot take effect until the starting task has terminated and freed up its bridge facility.

STARTed transactions

Some menu applications use START to initiate subsequent transactions.

You can specify BRIHSC-START in the BRIH-STARTCODE field of a single transaction mode request message, or in the first transaction of a session mode pseudoconversation, to return the correct response to ASSIGN STARTCODE and INQUIRE TASK STARTCODE commands issued by the user transaction.

User transactions that are initiated by START may issue one or more RETRIEVEs to obtain data passed on the START. When the bridge has passed all the data provided in the Link3270 request message, ENDDATA is returned to the user transaction.

Statistics

You cannot use `EXEC CICS COLLECT STATISTICS TERMINAL(xxxx)` where xxxx is a bridge facility.

Storage violation counts

No storage violation counts will be kept in a bridge facility.

TCTUA

The TCTUA is available to the user transaction using the EXEC CICS ADDRESS command. You can modify the contents of the TCTUA using the XFAINTU global user exit. See "Initializing the TCTUA" on page 51. Note that the TCTUA is NOT available to any programs in other CICS regions that are linked to by the user transaction using DPL.

Transaction restart

RESTART(NO) is forced for user transactions because CICS has no way of restoring the initial input message.

Transaction Routing

Transaction Routing is not directly supported, see "Transaction Routing considerations" for a technique you can use. The Link3270 bridge supports workload balancing with an affinity.

TWA

The TWA is available to the user transaction.

Transaction Routing considerations

Although the 3270 bridge does not directly support transaction routing, you can migrate applications using the following technique.

Add a **wrapper** program in the router region to drive initialization and termination routines as shown in the following table:

Client	Router wrapper	Bridged tran

Link to wrapper1	**wrapper1**
	ADDRESS COMMAREA(msg)
	brih-transaction =
	briht-allocate-facility
	LINK PROG(DFHL3270)
	  COMMAREA(alloc-msg)
	
	brih-transaction = **appl1**
	LINK PROG(DFHL3270)
	  COMMAREA(msg)
	
	brih-transaction = **term**
	LINK PROG(DFHL3270)
	  COMMAREA(dummy-msg)
	
	brih transaction =
	briht-delete-facility
	LINK PROG(DFHL3270)
	  COMMAREA(del-msg)
	
	READQ TS
	  INTO(appl-commarea)
	
	RETURN
	  COMMAREA(msg+appl-commarea)
		app1
		..BMS or 3270 commands..
		RETURN
		  COMMAREA(appl-commarea)
		
		term
		ADDRESS
		  COMMAREA(appl-commarea)
		WRITEQ TS
		  FROM(appl-commarea)

Client	Router wrapper	Bridged tran

Link to wrapper2	**wrapper2**
	ADDRESS
	COMMAREA(msg+appl-commarea)
	
	WRITEQ TS
	  FROM(appl-commarea)
	
	brih-transaction =
	briht-allocate-facility
	LINK PROG(DFHL3270)
	  COMMAREA(alloc-msg)
	
	brih-transaction=**init**
	LINK PROG(DFHL3270)
	  COMMAREA(dummy-msg)
	
	brih-transaction=**appl2**
	LINK PROG(DFHL3270)
	  COMMAREA(msg)
	
	brih-transaction =
	briht-delete-facility
	LINK PROG(DFHL3270)
	  COMMAREA(del-msg)
	
	RETURN
	  COMMAREA(msg)
		init
		READQ TS INTO(appl-commarea)
		RETURN
		  COMMAREA(appl-commarea)
		
		appl2
		ADDRESS
		  COMMAREA(appl-commarea)
		..BMS or 3270 commands..
		RETURN
		  COMMAREA(appl-commarea)

Note:

1. This solution could be varied according to the commarea size. If the msg+appl-commarea is greater than 32K, then rather than returning the appl-commarea to the listener, the init and term transansactions could write the commarea to a shared TS queue.

2. The same method can be used to initialize a TCTUA, large amounts of start data , or anything other parameters relating to the transaction environment.

Allocating a bridge facility name for a pseudoconversation when using the Link3270 bridge for transaction routing

In this example the application is controlled by a bridge client on the host.

About this task

This is described in "Select Link3270 client scenarios" on page 29.

Before running your client program:

1. Set the AIBRIDGE system initialization parameter to "yes" in the router region. This causes CICS to call the autoinstall user-replaceable program when a terminal ID has been allocated.

2. Ensure that your autoinstall user-replaceable program contains code to change the last character of the terminal ID in SELECTED-BRFAC-TERMID if it is set

to is set to "}". This character must be changed to a character that is unique to the system and can be an alphanumeric character or one of the following special characters: ¢@#./_$?!:|"=¬,;<>

If you are using NETNAME change it by copying SELECTED-BRFAC-TERMID to SELECTED-BRFAC-NETNAME.

Your client program should contain the following steps:

Procedure

1. Call the Link3270 bridge with an allocate-facility request. This bridge facility is referred to as the primary bridge facility in this example.
2. Set BRIH-FACILITYKEEPTIME to the time the application will take to run. If in doubt set it to the maximum value allowed. The maximum value is given in the description of BRMAXKEEPTIME in "Defining Link3270 system initialization parameters" on page 47. Set BRIH-FACILITYKEEPTIME to the time the application will take to run. If in doubt set it to the maximum value allowed. The maximum value is given in the description of BRMAXKEEPTIME in "Defining Link3270 system initialization parameters" on page 47.

   CICS calls the autoinstall user-replaceable program when the terminal ID for the bridge facility has been allocated.
3. When the transaction completes you may want to route to a different AOR:
   a. Keep the terminal ID and NETNAME which are returned from the Link3270 call in BRIH-TERMINAL and BRIH-NETNAME and do not delete the primary bridge facility.
   b. Allocate a new bridge facility using a Link3270 allocate-facility request. Before issuing this request, set BRIH-TERMINAL to the value of the primary bridge facility. Set BRIH-NETNAME also if you need NETNAME to be the same throughout. The facility allocated by this request is referred to as the secondary bridge facility in this example.
   c. When the autoinstall user-replaceable program is called for the new facility, SELECTED-BRFAC-TERMID is set to the value in BRIH-TERMINAL. Note that this name does not have "}" as the last character and the program will accept it.
   d. When changing to a third AOR, call Link3270 with a delete-facility request for the secondary bridge facility.
   e. Repeat steps 3a to 3d each time the target AOR changes.
4. When all transaction routing has finished, call Link3270 with a delete-facility request for the primary bridge facility.

Chapter 3. Using the Link3270 bridge

To run transactions using the Link3270 bridge, you must provide a client program that drives the Link3270 interface using LINK, EXCI LINK, or ECI requests. The message passed on each request determine the mode of operation, and the service to be performed.

To develop a client program to run an existing CICS 3270 transaction using the Link3270 bridge you need to:

- Establish the suitability of your applications for use with Link3270 .
- Design and write your client programs

CICS provides sample ECI, EXCI and LINK client programs to run the NACT sample transaction. You can use these as guidance in converting your own applications. See Chapter 7, "Using the Link3270 samples," on page 103 for more information about the Link3270 samples and NACT.

This chapter describes:

- "Establish Link3270 suitability"
- "Writing the Link3270 client" on page 28
- "Using Link3270 messages" on page 32
- "Using Link3270 single transaction mode" on page 36
- "Using Link3270 session mode" on page 37
- "Calling the Link3270 bridge" on page 42
- "Using data conversion with Link3270" on page 44

Establish Link3270 suitability

You need to establish that your applications are suitable for use with the Link3270 bridge.

This can be done in two ways:

1. Using the load module scanner to identify if the applications use any instructions that are not supported by the bridge. This involves the following steps:
   - Identify the programs used by the application
   - Run the load module scanner against the programs (see "Using the Load Module Scanner Utility" on page 28), using the bridge restrictions table DFHEIDBR. If there are any hits this indicates that there may be unsupported EXEC CICS commands.

     Note: note that the load module scanner can occasionally generate a false hit, so you will need to investigate the program to ensure that this has not occurred.
   - If there are unsupported commands you may be able to change them. If not, then the application is not supported by Link3270.
2. Using the 3270 Bridge Passthrough SupportPak (See "Using the 3270 Bridge Passthrough SupportPac" on page 28) to check if the application uses any unsupported interfaces.

The bridge is designed to support applications conforming to the documented CICS API specified in the Application programming reference overview in CICS Application Programming, subject to the restrictions described in "Link3270 programming considerations" on page 20. To confirm whether the application (and associated vendor products used on the system) conform to this, the application should be run under the Passthrough application. There may be various routes through the program which use different EXEC CICS API commands. Each of these routes should be tested using the Passthrough.

Using the Load Module Scanner Utility

The load module scanner is a batch utility that scans load modules for specified CICS API commands.

The commands to be reported upon are defined as a filter input file. A sample command filter list (DFHEIDBR) is provided to search for commands that are not supported in the 3270 bridge environment, and an output report identifies the commands and load module offsets, including EDF information if available. For more information about this utility, see Load module scanner (DFHEISUP) in the Operations and Utilities Guide.

Using the 3270 Bridge Passthrough SupportPac

The CA1E SupportPac is a support package providing the CICS 3270 Bridge Passthrough tool.

This allows you to run a CICS 3270 user transaction from a 3270 terminal in the normal way, but internally CICS uses the Link3270 bridge logic instead of real 3270 terminal support. This allows you to evaluate whether a CICS 3270 transaction is suitable to be driven using the 3270 bridge.

The Passthrough transactions also allow you to examine the 3270 data streams and log them for further analysis. You can then use this information to write the client program that will drive the CICS 3270 transaction instead of a real 3270 terminal.

The CA1E SupportPac can be obtained from the following address: http://www.software.ibm.com/ts/cics/txppacs.

Writing the Link3270 client

To design and write a client program to run an existing CICS 3270 transaction using the Link3270 bridge you need to:

1. select a suitable bridge scenario to decide where code needs to be written

2. analyze the application to understand the business data that flows between the 3270 and the application, so that you can replace it with messages

3. decide whether you can use the simplified single transaction mode interface or whether you need to use the full session mode interface

4. Write you client program using the selected scenario and transaction mode, using Link3270 messages to communicate with Link3270.

Link3270 has two modes of operation:

Single transaction mode
> This is a 'one-shot' type of request. A single transaction is run, and a single response message returned. The bridge facility is allocated automatically

by CICS and deleted at the end of the transaction. This mode is appropriate for inquiry type applications.

Session mode

This mode is appropriate for sequences of transactions where state data is maintained between transactions. In this mode, the client program can request:

- allocation of a bridge facility
- running of a transaction
- sending of continuation responses
- recovery from communication failure
- deletion of the bridge facility

Your client program manages the sequence of requests and the creation and deletion of the bridge facility. Note that this is different from the implementation of the START bridge, where the bridge facility is created dynamically.

Select Link3270 client scenarios

The following scenarios describe some common client environments.

They show how you can develop your client program to run in the most appropriate environment to make best use of existing skills and experience. These scenarios demonstrate tiered client applications that enable you to divide the logic to make best use of skills and experience. They use some common terms:

Business client

The business client is concerned only with the business data and its representation in the client end-user environment

bridge client

The bridge client builds the bridge messages and manages the communication with the bridge using the Link3270 interface. You can develop the more complex bridge client to run in CICS, using CICS commands, and the business client portion can run in any environment that allows communication with the bridge client. The bridge client can be designed to be reusable.

1. **Host CICS Client**

   In this scenario, shown in Figure 7 on page 30, the programmer has CICS skills and experience, so it is more appropriate to write the Link3270 interface code on CICS.

   You can separate the client logic into a business client, and a bridge client.

   The LINK and EXCI samples show how a client application can be separated in this way and how common logic can be shared in the bridge client. Note that a business client in another CICS region can use DPL to access the bridge client.
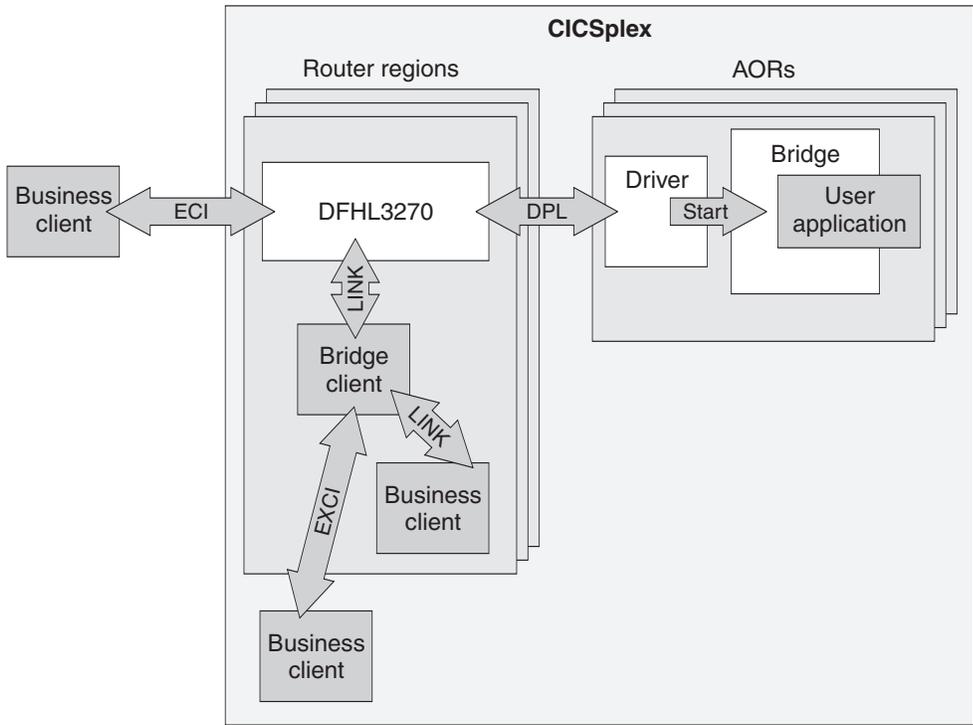
Figure 7. Link3270 host CICS client scenario

2. **CICS Workstation Client**

   In this scenario, shown in Figure 8, where a CICS product is installed on the workstation (such as CICS for Unix) then the client can be a CICS program using LINK to interface with Link3270, or with a host CICS bridge client. In the three tier model the writer of the bridge client needs to have CICS skills, but the business client programmer only needs skills on that platform.
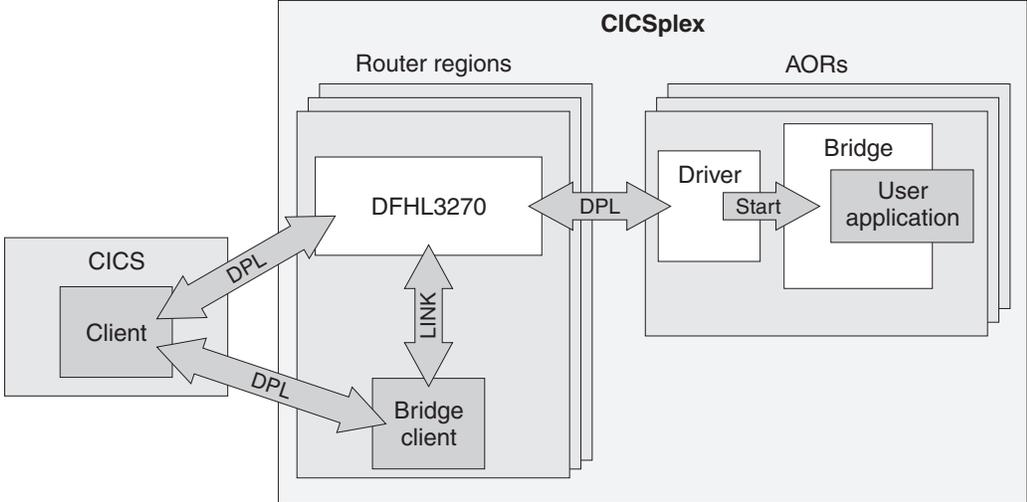
Figure 8. Link3270 CICS workstation client scenario

3. **Non - CICS Workstation Client**

   In this scenario, shown in Figure 9 on page 31, the programmer has workstation skills and limited CICS experience. For the two tier scenario, the

programmer must have some CICS experience to understand the messages (which involve EXEC CICS instructions).

The client program executes on a remote workstation, using ECI to drive the user application. A single client program is written, combining the business logic in the client environment and the interface to Link3270.
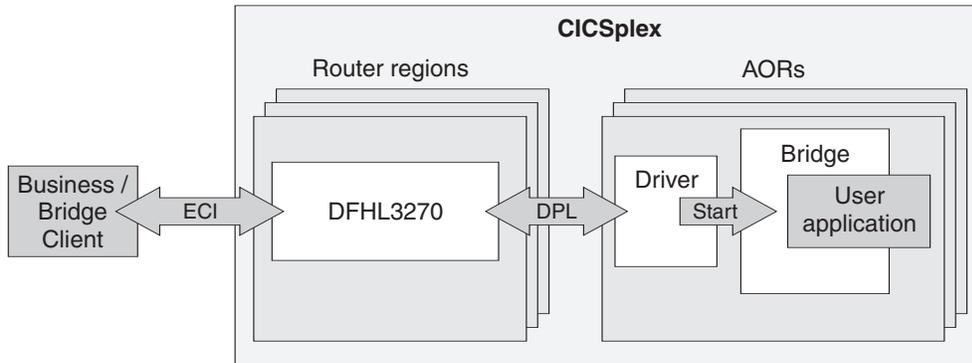
Figure 9. Link3270 non-CICS workstation client scenario

4. **3–tier Workstation client**

   In this scenario, shown in Figure 10, the workstation business client calls a bridge client in another environment, perhaps to utilize existing skills. For example, a Unix program could send a user-defined XML message to WebSphere on z/OS. A user-written bridge client application in WebSphere could then parse the XML message and convert it to a Link3270 message and use an EXCI LINK to call Link3270.
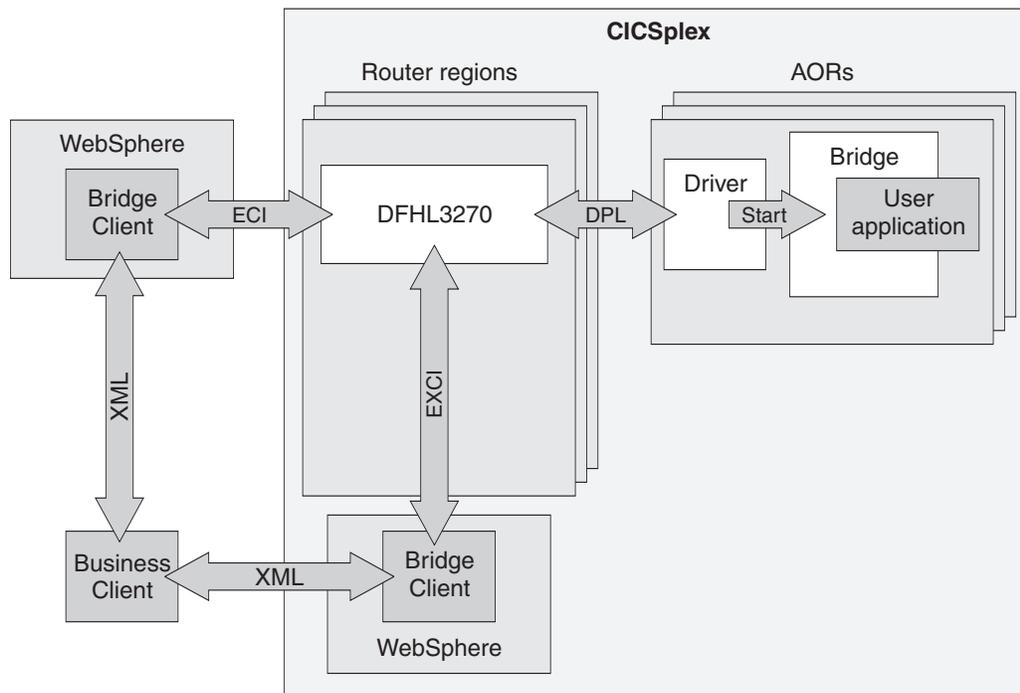
Figure 10. Link3270 3–tier client scenario

Analyze the 3270 application

You need to analyze the 3270 application programs that form your transaction in order to replace the 3270 input data with messages sent by your client program.

You can use the 3270 Bridge Passthrough SupportPak (See "Using the 3270 Bridge Passthrough SupportPac" on page 28) to drive your applications and log 3270 commands.

3270 programs fall mainly into the following types:

Minimum function BMS
This includes minimum function, SEND TEXT, and ACCUM. Link3270 supports these applications. Bridge clients can be written relatively easily by a CICS programmer.

BMS and little 3270 datastream
A typical example of this type is an application that issues RECEIVE of command line input, and issues a send MAP as output. Link3270 supports these applications. Client applications can be written relatively easily by a CICS programmer.

Mixed BMS and 3270 datastream
A typical example of this type is an application that issues RECEIVE and then RECEIVE MAP FROM. Link3270 supports these applications but the client programs are more difficult to write because MAP information has to be supplied in 3270 datastream format.

Pure 3270 datastream
Typically user-written 3270 datastream where the user has a well-understood fixed 3270 datastream structure. This is supported but the client program is more difficult to program than with BMS. However this is usually the format understood by the programmer of the target application.

Alternative map generators
3270 datastreams are generated dynamically. Link3270 supports these applications but the client programs are more difficult to write because MAP information has to be supplied in 3270 datastream format.

Full function BMS
Including ACCUM, PAGE and PARTITION support. These programs are not supported and will be detected by the load module scanner or the Passthrough tool.

Using Link3270 messages

To run transactions using the Link3270 Bridge, a client program: creates an inbound message, links to Link3270 with a COMMAREA containing the message, and interprets the result of the outbound message.

The inbound message
The inbound message is passed on the LINK, ECI or EXCI call as a COMMAREA. It contains the following data structures:

Bridge message header (BRIH)
A data structure containing parameters to be passed to the Link3270 bridge mechanism, such as the name of the user transaction; the facility-like template to be used when the bridge facility is created, and the termid to be assigned to the bridge facility.

Bridge message vectors (BRIV)s
Zero or more data structures containing data to be passed to the user transaction containing the data requested by the EXEC CICS command for 3270 terminal input.

For example, if the application issues an **EXEC CICS RECEIVE MAP**, the inbound message will have the following form:

Table 2. Message structure for the EXEC CICS RECEIVE MAP command

BRIH	BRIV-RM	ADS

Where ADS is the application data structure expected by the RECEIVE MAP command.

Sample BRIH and BRIV copybooks are supplied, primed with the default values, to simplify programming. You can include these in your program and then change only the specific fields relevant to the request.

The outbound message
The outbound message is passed in the COMMAREA on return from the LINK, ECI or EXCI call. It contains the following data:

Bridge message header (BRIH)
A data structure containing parameters returned by the Link3270 mechanism, such as return and response codes; the actual termid assigned to the bridge facility, and the length of the returned message.

Bridge message vectors (BRIV)s
Zero or more data structures containing the data supplied by the **EXEC CICS** command for a 3270 terminal output request, or requests for more data, to be passed to the client program.

For example, if the application issues several non-terminal EXEC CICS commands, an **EXEC CICS SEND MAP** and then an **EXEC CICS RETURN**, the outbound message will have the following form where ADS is the application data structure expected by the **SEND MAP** command.

Table 3. Message structure for the EXEC CICS SEND MAP command

BRIH	BRIV-SM	ADS

A more complicated example would be one where the application issues several non-terminal EXEC CICS commands, an **EXEC CICS SYNCPOINT**, an **EXEC CICS SEND CONTROL**, an **EXEC CICS SEND MAP**, and then an **EXEC CICS RETURN**. In this case, the outbound message has the following form:

Table 4. More complicated message structure

BRIH	BRIV-SP	BRIV-SC	BRIV-SM	ADS

Inbound BRIV vectors

One BRIV vector is required containing the data requested by every EXEC CICS command for 3270 terminal input issued by the user transaction.

The following commands are supported:
- CONVERSE
- RECEIVE
- RECEIVE MAP

Note: If the application issues CONVERSE, and there is an inbound converse vector to satisfy this request, then the output from the converse is used to build an output SEND vector.

When the user transaction issues the command, the bridge mechanism searches the inbound message for the first BRIV that matches the command type. For RECEIVE MAP commands it attempts to match the MAPSET and MAP if these have been supplied by the client. RECEIVE MAP vectors are processed in order, and those that do not match the current command are discarded until a match is found. Blank names in the vector match any command.

Where there are several input vectors of different types, the order is not important.

For 'conversational' transactions (see "Conversational transactions" on page 39) when the client is asked for further input, the previous inbound message vectors (except RETRIEVE vectors) are discarded when a new inbound message is received. Note that RETRIEVE vectors can only flow in the first message of the first transaction in a session. See "Using Link3270 session mode" on page 37 for an explanation of the session programming mode.

Outbound BRIV vectors

One BRIV vector is created containing the data supplied by every **EXEC CICS** command for 3270 terminal output issued by the user transaction. This passes to the client all the information and data relating to the command.

The following commands are supported:
- ISSUE ERASEUP
- SEND
- SEND MAP
- SEND TEXT
- SEND CONTROL
- SYNCPOINT
- SEND PAGE
- PURGE MESSAGE

Note: SEND PAGE and PURGE MESSAGE are only available for the Link3270 bridge with extended support. See"Link3270 bridge basic and extended support" for an explanation of the differences between Link3270 bridge basic and extended support.

For 'conversational' transactions (see "Conversational transactions" on page 39), the last BRIV vector can represent an **EXEC CICS** command that requests more data. This vector is only created if the previous input message did not contain a BRIV to satisfy all the CICS commands. The following commands are supported:
- CONVERSE request
- RECEIVE request
- RECEIVE MAP request
- RETRIEVE request

Link3270 bridge basic and extended support

There are two levels of support for the Link3270 bridge.

- Link3270 bridge with basic support provides the same as the support as that provided in CICS Transaction Server for z/OS, Version 2 Release 2. If you want to continue using basic support, you do not need to take any action. Basic support is provided automatically.
- Link3270 bridge with extended support provides support for the ACCUM option on EXEC CICS SEND TEXT, EXEC CICS SEND MAP, and EXEC CICS SEND CONTROL, in addition to the basic support. To support the ACCUM option, there are two extra outbound vectors, SEND PAGE and PURGE MESSAGE. If you want to take advantage of extended support, you must recompile any CICS Transaction Server for z/OS, Version 2 Release 2 Link3270 programs using the extended copybooks (listed in Table 6), instead of the basic copybooks.

Copybooks and default vectors

To simplify the task of constructing and analyzing Link3270 messages, CICS provides copybooks and header files containing BRIH and BRIV structures. Sample BRIH and input BRIV structures already primed with default values are also supplied, so all you need to do is copy them into your COMMAREA and modify relevant fields.

Default structures and message copybooks

The following default structures are supplied in all supported languages:
- BRIH-DEFAULT
- BRIV-CONVERSE-DEFAULT
- BRIV-RECEIVE-DEFAULT
- BRIV-RECEIVE-MAP-DEFAULT
- BRIV-RETRIEVE-DEFAULT

You will find the copybooks and headers in the files listed in the tables below. Table 5 shows the basic copybooks. Table 6 shows the extended copybooks.

Table 5. Link3270 message copybooks for basic support

structure	COBOL	C	PLI	Assembler
BRIH	DFHBRIHO	DFHBRIHH	DFHBRIHL	DFHBRIHD
Inbound BRIVs	DFHBRIIO	DFHBRIHH	DFHBRIHL	DFHBRIHD
Outbound BRIVs	DFHBRIOO	DFHBRIHH	DFHBRIHL	DFHBRIHD
Defaults and constants	DFHBRICO	DFHBRICH	DFHBRICL	DFHBRICD

Table 6. Link3270 message copybooks for extended support

structure	COBOL	C	PLI	Assembler
BRIH	DFHBR2HO	DFHBR2HH	DFHBR2HL	DFHBR2HD
Inbound BRIVs	DFHBR2IO	DFHBR2HH	DFHBR2HL	DFHBR2HD
Outbound BRIVs	DFHBR2OO	DFHBR2HH	DFHBR2HL	DFHBR2HD
Defaults and constants	DFHBR2CO	DFHBR2CH	DFHBR2CL	DFHBR2CD

Sample client programs are supplied to illustrate the use of the copybooks and defaults. You will find a description of these in "Running the sample client programs" on page 105.

Using Link3270 single transaction mode

Single transaction mode allows a client to run a single transaction.

The bridge facility is automatically created, then deleted at the end of the transaction. This mechanism is more efficient and easier to program than session mode, if only one transaction is being run. It is particularly suited to inquiry transactions.

To run in single transaction mode, your client program must supply the name of the user transaction in the BRIH_TRANSACTIONID field of the bridge message header (BRIH). See "Link3270 message header (BRIH)" on page 62 for a description of the BRIH. The following parameters may also be optionally defined:
* BRIH-DATALENGTH (if BRIV vectors are appended to the message)
* BRIH-FACILITYLIKE
* BRIH-TERMINAL
* BRIH-NETNAME
* BRIH-ADSDESCRIPTOR
* BRIH-ATTENTIONID
* BRIH-STARTCODE
* BRIH-CURSORPOSITION

Your client program must also create BRIV vectors for any input commands. If you add data to a BRIV, you must also update the BRIH-DATALENGTH field. See "Updating data length fields".

To use single transaction mode your application must satisfy the following restrictions, otherwise session mode should be used:
* Only one input and one output message are allowed. To run conversational transactions, you must provide all the input data in sequential BRIV structures in the input message. The BRIH-CONVERSATIONALTASK and BRIH-GETWAITINTERVAL parameters in the BRIH are ignored.
* The COMMAREA should be large enough to receive the output message. If it is not, the message is truncated at the last complete vector, and the rest of the message is discarded. BRIH-REMAININGDATALENGTH is set to a non zero value to indicate there has been truncation.
* If a communications link breaks, you can not obtain the output using a resend message request. For this reason it is recommended that single transaction mode is mainly used for inquiry type transactions.

Updating data length fields

If you add data to a BRIV you must update the following fields to include the length of the data:
* The BRIV data length field
* The BRIV header vector length field

If you add a BRIV to a message, you must add the value in the BRIV header vector length field to BRIH-DATALENGTH.

Using Link3270 session mode

Session mode allows a client to run a number of transactions using the same bridge facility.

It is more efficient than running each of these transaction in single transaction mode. Session mode supports the following operations:
- Allocating a bridge facility
- Running transactions
- Deleting a bridge facility
- Delivering large messages
- Recovery in the event of communications failure

Note: The USERID must be the same for the whole session and must be specified in every Link3270 request.

How to create a message

Before sending each message you must perform these steps.

Procedure
1. Move the default bridge message header (BRIH) into a message area.
2. For all messages other than the allocate, set BRIH_FACILITY to the value returned on the allocate.
3. Modify other parameters of the message as required, as described in the following sections.

Allocating a bridge facility

To allocate a bridge facility, your client program must set the value of the BRIH_TRANSACTIONID field of the bridge message header (BRIH) to BRIHT-ALLOCATE-FACILITY.

See "Link3270 message header (BRIH)" on page 62 for a description of the BRIH. The following parameters may also be optionally defined:
- BRIH-FACILITYKEEPTIME
- BRIH-FACILITYLIKE
- BRIH-TERMINAL
- BRIH-NETNAME

Other fields in the message are ignored.

For example, to allocate a bridge facility using the supplied default BRIH and constants:

```
Working-Storage Section
...
copy dfhbrico.
...
Linkage Section
    01 msg-area
        copy DFHBRIHO.
...
Procedure Division.
...
move brih-default to msg-area.
set briht-allocate-facility to true.
```

```
EXEC CICS LINK PROGRAM('DFHL3270) COMMAREA(msg-area)
               LENGTH(length of brih) DATALENGTH(len)
               END-EXEC
...

:
```

Note: The BRIH-FACILITYLIKE value supplied by your client program is not
validated until the first application transaction is run. It is only when the first
application transaction is processed that the AOR region is determined and the
facilitylike value can be validated within the selected AOR.

Running transactions

To run transactions in session mode, your client program must supply the name of
the user transaction in the BRIH_TRANSACTIONID field of the bridge message
header (BRIH), and set BRIH-FACILITY to the value returned by the allocate
request.

The following parameters may also be optionally defined:
- BRIH-DATALENGTH
- BRIH-CONVERSATIONALTASK
- BRIH-GETWAITINTERVAL
- BRIH-ADSDESCRIPTOR
- BRIH-ATTENTIONID
- BRIH-STARTCODE
- BRIH-CURSORPOSITION

Other fields in the BRIH are ignored.

For example, to run transaction NACT using the supplied default BRIH and
constants:
```
Working-Storage Section
.
.
copy dfhbrico.
.
Linkage Section.
01 msg-area.
   copy DFHBRIHO.
   03 msg-vectors pic x(2000).
.
Procedure Division
.
    move brih-default  to msg-area
    move 'NACT' to brih-transactionid
    move facility to brih-facility
    move brih-datalength to len
    EXEC CICS LINK PROGRAM('DFHL3270') COMMAREA(msg-area)
                LENGTH(length of msg-area) DATALENGTH(len)
                END-EXEC
.
.
```

Your client program must also create BRIV vectors for any input commands. For
example:
```
        move briv-receive-map-default to briv-in.
        move 'DFH0MNA '            to briv-rm-mapset.
        move 'ACCTMNU '            to briv-rm-map.
```

```
          move '422'                  to briv-rm-cposn.
          move length of acctmnui     to briv-rm-data-len.
          set  address of acctmnui    to address of briv-rm-data.
          move low-values             to acctmnui.
          add  briv-rm-data-len       to briv-input-vector-length.
          add  briv-input-vector-length to brih-datalength.
```

Note: When adding a BRIV always remember to increment the
BRIH-DATALENGTH

Conversational transactions

A traditionally conversation transaction, making multiple interactions with a
terminal, can be run under the Link3270 bridge as a simple 'non-conversational'
transaction by providing all the terminal input in multiple BRIV vectors in the
Link3270 request message.

Here, the term 'conversational' refers to transactions where there are multiple flows
between the client and the user transaction. To enable this conversational
interaction, you must set BRIH-CONVERSATIONALTASK to BRIHCT-YES.

If the user transaction encounters a CONVERSE, RECEIVE or RECEIVE MAP and
the Link3270 mechanism has not received a BRIV to satisfy the request, and the
BRIH allows conversations (BRIH-CONVERSATIONALTASK is set to
BRIHCT-YES), a message is returned to the client requesting further data. The
value of BRIH-TASKENDSTATUS is set to the value BRIHTES-CONVERSATION,
and a request BRIV is the last vector in the message.

The client then responds by sending a further input message containing the
required 3270 input data. The client initializes the message to the default BRIH and
sets the value of the BRIH-TRANSACTIONID field to BRIHT-CONTINUE-
CONVERSATION and BRIH-FACILITY to the value returned on the allocate
request. The following parameters may also be optionally defined:

- BRIH-DATALENGTH (if BRIV vectors are appended to the message)
- BRIH-CONVERSATIONALTASK
- BRIH-GETWAITINTERVAL
- BRIH-CANCELCODE

Other fields in the BRIH are ignored.

The client program may also need to create BRIV vectors if appropriate, and it
must reply within the time specified in BRIH-GETWAITINTERVAL

Note: If BRIH-CONVERSATIONALTASK is set to BRIHCT-NO, the bridge will
abend the user transaction if it issues an input command for which no vector has
been supplied.

Pseudoconversational transactions

A pseudoconversation normally involves a series of transactions, each initiated by
the previous transaction, which may also pass some data. The name of the next
transaction to be run can be defined by the user transaction in different ways.

1. **EXEC CICS RETURN TRANSID**
2. **EXEC CICS RETURN TRANSID IMMEDIATE**
3. **EXEC CICS START TRANSID TERMID**
4. **EXEC CICS SET TERMINAL/NETNAME NEXTTRANSID**
5. Terminal data

Note: Transactions initiated by START TERMID are not necessarily
pseudoconversational. Here we are considering only those transactions initiated by
a START to the principal facility (the bridge facility) where the STARTING and
STARTED applications are associated in a pseudoconversation. In this case, START
TERMID must specify the bridge facility.

Commands 1-4 all cause the bridge mechanism to set the next transaction identifier
in the BRIH-NEXTTRANSACTIONID field to be returned to the client in the next
response message.

The client responds by sending a run request for the next transaction, with
BRIH-TRANSACTIONID set to the value from BRIH-NEXTTRANSACTIONID and
BRIH-FACILITY set to the value returned on the allocate request. The following
parameters may also be optionally defined:
- BRIH-DATALENGTH (if BRIV vectors are appended to the message)
- BRIH-CONVERSATIONALTASK
- BRIH-GETWAITINTERVAL (if conversational)
- BRIH-ADSDESCRIPTOR
- BRIH-ATTENTIONID
- BRIH-CURSORPOSITION

Other fields in the BRIH are ignored.

Note: The same bridge facility must be used by all transactions in the
pseudoconversation.

Deleting a bridge facility

When all session activity is complete, the client can delete the bridge facility.

To do this, your client program must set the value of the BRIH_TRANSACTIONID
field of the BRIH to BRIHT-DELETE-FACILITY, and set BRIH-FACILITY to the
value returned by the allocate request. Other fields in the message are ignored.

For example, to delete a bridge facility using the supplied default BRIH and
constants:

```
Working-Storage Section
.
.
copy dfhbrico
.
Linkage Section.
01 msg-area.
 copy DFHBRIHO.
.
Procedure Division
.
    move brih-default  to msg-area
    set briht-delete-facility to true
    move facility to brih-facility
    move brih-datalength to len
    EXEC CICS LINK PROGRAM('DFHL3270') COMMAREA(msg-area)
                   LENGTH(length of brih) DATALENGTH(len)
                   END-EXEC
.
.
```

If the bridge facility is not explicitly deleted, it is scheduled for deletion automatically by CICS if it is unused for the time specified in the BRIH-FACILITYKEEPTIME field, or in the BRMAXKEEPTIME system initialization parameter. The smaller interval is used.

Delivering large messages

If the output message from the user transaction is larger than the size of the COMMAREA passed on the request, the bridge mechanism returns a BRIH and as many complete BRIV vectors as will fit into the returned COMMAREA.

If it is not possible to fit the whole of the outbound message into the COMMAREA, the field BRIH-REMAININGDATALENGTH is set to a non zero value. The client can then issue one or more requests to obtain the rest of the data. To do this, your client program must set the value of the BRIH-TRANSACTIONID field to BRIHT-GET-MORE-MESSAGE, and set BRIH-FACILITY to the value returned by the allocate request. Other fields in the message are ignored.

This is so that CICS can return error information. Clients should follow CICS recommendations regarding COMMAREA lengths described in LENGTH options in CICS commands, in the *CICS Application Programming Reference*.

Recovery from connection failure

If the communications connection fails before a response message is received, the client can reconnect to the same router and request that the message be sent again.

To do this, your client program must set the value of the BRIH-TRANSACTIONID field to BRIHT-RESEND-MESSAGE, and set BRIH-FACILITY to the value returned by the allocate request. Other fields in the message are ignored.

If successful, the outbound Link3270 bridge message contains as much of the message as can be fitted into the COMMAREA. If either the router or the AOR CICS region has failed, the message returned indicates that the facilitytoken is unknown.

If unsuccessful, the output is the BRIHT-RESEND-MESSAGE message with an appropriate BRIH-RETURNCODE.

Note:

1. A resend request must be sent before the interval specified in BRIH-FACILITYKEEPTIME on the allocate request has expired. Otherwise, both the bridge facility and the outstanding message are deleted.
2. You can use the field BRIH-SEQNO to check whether the previous request has worked.

Validity of Link3270 requests

At any time, the bridge facility is considered to be in a specific *state*.

Some requests are only valid is the facility is in an appropriate state. If the request is not valid, BRIH-RETURNCODE is set to the value indicated in Table 7 on page 42.

Possible states are:
• Not Allocated

- Allocated
- Conversational
- Transaction Ended

The following table will help you to decide when a request is valid, and what the resulting state will be. If a request is invalid, the state does not change:

Table 7. Validity of Link3270 requests

Request	Not Allocated	Allocated	Conversational	Transaction ended
Allocate Facility	valid ->Allocated	note[8]	note[8]	note[8]
Run Transaction	valid[2]	valid[4]	invalid[3]	valid[4]
Continue Conversation	invalid[1]	invalid[5]	valid[4]	invalid[5]
Get More Message	invalid[1]	invalid[6]	valid/invalid[7]	valid/invalid[7]
Resend Message	invalid[1]	invalid[6]	valid ->Conversational	valid ->Transaction ended
Delete Facility	invalid[1]	valid ->Not Allocated	invalid[3]	valid ->Not Allocated

Note:
1. BRIH-RETURNCODE set to BRIHRC-INVALID-FACILITYTOKEN.
2. This is defined as single transaction mode.
3. BRIH-RETURNCODE set to BRIHRC-FACILITYTOKEN-IN-USE.
4. The resulting stated depends on whether the transaction issues further requests for which there is no BRIV. Possible new states are Conversational or Transaction-ended
5. BRIH-RETURNCODE set to BRIHRC-TRANSACTION-NOT-RUNNING.
6. BRIH-RETURNCODE set to BRIHRC_NO-DATA.
7. The resulting state depends on whether there is more data to send (indicated by BRIH-REMAININGDATALENGTH).
8. This state is not relevant, as Allocate always creates a new facility.

Expiry of facilitytoken
If the facilitytoken expires due to inactivity, any subsequent requests are invalid. BRIH-RETURNCODE is set to BRIHRC-INVALID-FACILITYTOKEN and the resulting state is Not Allocated. Conversational requests may result in loss of data.

Calling the Link3270 bridge

The Link3270 bridge supports these external request mechanisms.
1. **EXEC CICS LINK**. This includes both local link and DPL.
2. The External CICS Interface (EXCI). This includes both the EXCI call interface and the EXEC CICS interface.
3. The External Call Interface (ECI).

Calling Link3270 using LINK

The interface is the standard **EXEC CICS LINK** interface.

```
EXEC CICS LINK PROGRAM('DFHL3270')
          COMMAREA(Link3270_message)
          DATALENGTH(inbound_message_length)
          LENGTH(outbound_message_length)
```

- PROGRAM must specify DFHL3270.
- The COMMAREA must contain a structured Link3270 message, as described in Chapter 5, "Link3270 message formats," on page 61.

If you are using DPL:

- SYSID may be specified. If there are multiple router regions, all calls must be issued to the same region where the allocate-facility call was sent.
- SYNCONRETURN can be used, but is not required. If it is not used, a mirror task remains in the router for the duration of the session.
- TRANSID can be used
- INPUTMSG and INPUTMSGLEN are ignored.

The bridge header (BRIH) indicates whether the transaction ran successfully or not. See Chapter 6, "Link3270 diagnostics," on page 97 for a full description of the return codes from the Link3270 call.

For a full description of the **LINK** command, see LINK in CICS Application Programming .

Calling Link3270 using EXCI

Either form of the EXCI interface can be used to run the bridge.

The **EXEC CICS** interface is recommended for the single transaction mode. The call interface is recommended for the session mode. See Chapter 3, "Using the Link3270 bridge," on page 27 for a description of single transaction and session modes. See "The EXCI programming interfaces" on page 114 for information about using the EXCI interface.

Calling Link3270 using ECI

The interface is the standard ECI interface, passing the ECIPARMS parameter list.

This should contain the following specific fields:

parameter	value
eci_call_type	synchronous or asynchronous
eci_program_name	DFHL3270
eci_userid	Userid for security validation. The user transaction runs with this userid
eci_password	Password or Passticket for security validation
eci_tpn	User transaction name
eci_commarea	Address of the Link3270 message
eci_commarea_length	Length of the Link3270 message

The other fields are set according to normal ECI programming. See *CICS Family: Client/Server Programming* for more information about the using the ECI interface.

The return code from the ECI call indicates whether the request was accepted by CICS. A return code of ECI_NO_ERROR does not imply that the transaction ran successfully. It implies that the transmission of the message was successful. The client application should look in the returned bridge header (BRIH) for the return code and abend code. See Chapter 6, "Link3270 diagnostics," on page 97 for a full description of the return codes from the Link3270 call.

Multiple Router regions

If there are multiple router regions, all calls must be issued to the same region where the allocate-facility call was sent.

Using data conversion with Link3270

If the codepage of your client program is different from the codepage used by the CICS user program, your messages need to be converted.

The BRIH, all BRIV vector headers and RETRIEVE data can be converted using the CICS conversion program DFHCCNV. See the *CICS Intercommunication Guide* for information about the data conversion process.

DFHCCNV uses the DFHCNV table to determine the required conversions. You need to supply entries in this table for each resource that requires conversion. See the *CICS Intercommunication Guide* .

Converting BRIH and BRIV header data

If you are using codepages other than the defaults, that is, other than 437 for the client and 037 for the server, then you must ensure that the correct codepage conversion is applied to DFHBRCTD.

You can do this by coding `SRVERCP` and `CLINTCP` parameters on the `DFHCNV TYPE=INITIAL` statement. The `COPY DFHBRCTD` statement should follow `DFHCNV TYPE=INITIAL`.

If the application programs run using the same codepages as those specified on the `TYPE=INITIAL` statement, then the `TYPE=ENTRY` statements do not need to specify the codepages.

If an application program, or programs, need to use different codepages, then the new values must be specified on the appropriate `TYPE=ENTRY` statements.

In your DFHCNV, you should include:

```
COPY DFHBRCTD
```

Build DFHCNV as described in the *CICS Intercommunication Guide*. This will convert the BRIH and BRIV vector headers using the codepages described in your conversion template.

DFHCNV example for Link3270

This shows an example of a DFHCNV for a CICS region using codepage 939 for the server programs and codepage 943 for most of the clients.

The COPY DFHBRCTD statement follows the DFHCNV TYPE=INITIAL statement, so it will use 939 and 943. This is used in the conversion of BRIHs.

PROG1 uses the same codepages, but PROG2 uses client codepage 932 instead of 939. In order to achieve this, the TYPE=ENTRY statement for PROG2 contains overrides for the client and server codepages.

```
DFHCNV    TITLE 'EXAMPLE DFHCNV CONVERSION TABLE'
*
          DFHCNV TYPE=INITIAL,SRVERCP=939,CLINTCP=943
*
          COPY DFHBRCTD
*
          DFHCNV TYPE=ENTRY,RTYPE=PC,RNAME=PROG1
          DFHCNV TYPE=SELECT,OPTION=DEFAULT
          DFHCNV TYPE=FIELD,OFFSET=0,DATATYP=CHARACTER,DATALEN=10018,   *
               LAST=YES
*
          DFHCNV TYPE=ENTRY,RTYPE=PC,RNAME=PROG2,                       *
               SRVERCP=939,CLINTCP=932
          DFHCNV TYPE=SELECT,OPTION=DEFAULT
          DFHCNV TYPE=FIELD,OFFSET=0,DATATYP=CHARACTER,DATALEN=500,     *
               LAST=YES
*
          DFHCNV TYPE=FINAL
          END
```

Figure 11. DFHCNV example for Link3270

Converting RETRIEVE data

RETRIEVE data is converted using the conversion template for the user transaction. You should provide a DFHCNV entry with RTYPE=IC RNAME=*user tranid*.

Converting user data

User data following each BRIV (including the ADSD, but excluding RETRIEVE data) is converted in the AOR using the GCHARS and GCODES values defined for the facility-like terminal that was used as a template to build the bridge facility.

You should define these values appropriately for your client program, and specify the facility-like name in your Link2370 call if you are not using the default. The client codepage must be specified in value 2 of CGCSGID in the AOR as well as in the DFHCNV conversion table in the router region.

The codepage conversion for user data is based on the codepage conversion for EPI emulation.

CGCSGID values 1 and 2 can be retrieved by an application program using **EXEC CICS ASSIGN GCHARS** and **EXEC CICS ASSIGN GCODES** respectively.

Note: SEND TEXT NOEDIT data is always converted as if it were in 3270 data stream format.

Chapter 4. Managing the Link3270 bridge environment

Before you can use the Link3270 bridge you must define a DFHBRNSF file.

Defining a DFHBRNSF file is described in "Defining the bridge facility name" on page 49.

Optionally, you can also:
- Allocate specific names to bridge facilities using the autoinstall user replaceable program. See "Defining the bridge facility" on page 48 .
- Control the keeptime for bridge facilities. See "Defining Link3270 system initialization parameters" .
- Use the dynamic transaction routing program to enable load balancing. See "Using Link3270 bridge load routing" on page 57

This chapter describes the CICS interfaces provided to help you set up and control a Link3270 bridge environment. It describes:
- "Defining Link3270 system initialization parameters"
- "Defining the bridge facility" on page 48
- "Managing Link3270 bridge resources" on page 54
- "Using Link3270 bridge load routing" on page 57

Defining Link3270 system initialization parameters

These system initialization parameters affect the operation of the Link3270 bridge.

AIBRIDGE={AUTO|YES}
Code this parameter to specify whether the autoinstall URM is called when bridge facilities are created and deleted.

> **AUTO**
> The autoinstall URM is not called.

> **YES**
> The URM is called for all new bridge facilities and can change the termid and netname, or can reject the request

BRMAXKEEPTIME={86400|*timeout* }
Bridge facilities are deleted after they have been unused for a given length of time. The timeout value can be specified by the client when it sends in a request to create a bridge facility in session mode. This parameter specifies the maximum timeout value that a client can specify (in seconds). If the client specifies a larger value than the BRMAXKEEPTIME value in the AOR, then CICS will change this parameter in the link parameter list (but does not return the reduced value to the client).

> Use this parameter to ensure that CICS does not run short of bridge facilities caused by clients reserving bridge facilities for too long a period.

> **86400**
> The default maximum timeout value (24 hours) that a client can specify to retain an unused bridge facility before it is deleted.

> **timeout**
> The maximum timeout value that a client can specify (in seconds),

before an unused bridge facility is deleted. The value specified must be in the range 0 to 86400. A value of 0 means that bridge facilities are never kept at the end of a transaction. Therefore CICS will not be able to run pseudoconversational transactions. This may be useful if the region is only used for inquiry transactions. The default value is 24 hours (86400 seconds).

You can also specify the **SPCTR** and **STNTR** parameters to request standard and special tracing for the bridge (BR) and partner (PT) domains:

BR domain tracing
- Exception tracing for disasters and significant user errors.
- Level 1 normal domain tracing.
- Level 2 additional information. Messages and vectors trace only up to 4000 bytes.

PT domain tracing
- Exception tracing for disasters and significant user errors.
- Level 1 normal domain tracing.
- Level 2 additional information.

See CICS startup in the System Definition Guide for further information.

Defining the bridge facility

The bridge facility is an emulated 3270 terminal. It is a virtual terminal, created by Link3270 when it receives a single transaction mode request, or a session mode request to allocate a bridge facility.

You do not provide a TERMINAL resource definition for the bridge facility, but you can control the terminal properties used by providing a 3270 TERMINAL resource definition to be used as a template. This TERMINAL definition, is known as the **facilitylike**.

Defining the facilitylike

The facilitylike value is the name of a real terminal resource definition that is used as a template for some of the properties of the bridge facility.

The name of the facilitylike definition to be used can be passed to CICS in one of three ways (the first non-blank value found is used):
- From the BRIH-FACILITYLIKE parameter in the Link3270 call.
- From the PROFILE resource definition for the user transaction.
- The default is CBRF, a definition supplied by CICS to support the bridge.

.

Once the bridge facility has been defined, its facilitylike template cannot be changed. Therefore, if the bridge facility is reused in session mode, CICS ignores the facilitylike value passed in subsequent calls.

Note: If you are running in a CICS system started with the VTAM=NO system initialization parameter, the resource definition specified by facilitylike must be defined as REMOTE. A default definition of CBRF, defined as REMOTE, is provided in the group DFHTERM.

Note: VTAM® is now the z/OS Communications Server.

For information about the PROFILE resource definition, see PROFILE resources in the Resource Definition Guide.

Defining the bridge facility name

When CICS creates a bridge facility, it creates both an eight-byte token to identify it (the **facilitytoken**) and a four-character terminal identifier, which is used as both TERMID and NETNAME.

The facilitytoken is returned on the Link3270 allocate call and must be supplied by you on all subsequent session mode calls (See "Using Link3270 session mode" on page 37).

For bridge facilities created by the Link320 bridge, the token and name are unique across the CICSplex, and the TERMID and NETNAME are of the form AAA}. Naming occurs in the routing region, at the time of processing an "allocate" command in session mode, or the internal allocate step in single-transaction mode. See Chapter 3, "Using the Link3270 bridge," on page 27 for information about session and single-transaction modes.

Link3270 bridge facility namespace allocation information is recorded in a shared file, DFHBRNSF, to ensure uniqueness of the names. The router regions that share file DFHBRNSF and their associated AOR's form the bridge CICSplex. Multi-bridge CICSplexes can be set up within a larger CICSplex , each router region within a bridge CICSplex sharing the same DFHBRNSF file. The AOR regions of a bridge CICSplex can only be associated with router regions on one bridge CICSplex.

To improve performance, the Link3270 bridge namespace is split into allocation ranges, so that a 'chunk' of names is allocated to each router region, and the DFHBRNSF file is only accessed when a namespace range is allocated or released. Names within the allocated chunks can be reused when keeptimes expire, and chunks may also be reused in other regions, so you may see the same names appearing in different regions, but they are only active in one region at any given time.

Message DFHBR0505 is issued when 90% of the DFHBRNSF names have been allocated and is issued for each percentage point increase in the names being allocated. Message DFHBR0506 is issued for each percentage point reduction in names allocated until below 90%. When no more names are available, message DFHBR0507 is issued, and client application new allocation (or one shot) requests receive a return code of BRIHRC_NO_FREE_NAME.

DFHBRNSF file types

The bridge facility namespace allocation file (DFHBRNSF) can be a local user data table, a local VSAM file, a coupling facility data table (CFDT), a remote VSAM file or a VSAM RLS file.

If only one router region is used a user maintained data table or local VSAM version of DFHBRNSF is recommended.

If multi-router regions are used, the DFHBRNSF file can be defined as a local VSAM file in a remote file owning region (FOR) and as a remote VSAM file in the router regions; as a VSAM RLS in all the router regions, or as a coupling facility data table in all the router regions.

If the user maintained data table version of DFHBRNSF is used, the VSAM data set specified in the CICS file definition must be empty; no records should be loaded from the file to the data table. The data table should not be closed when the router region is running, because all bridge facility namespace data will be lost and the next request to allocate or release a range of bridge facilities will fail. For this reason, a user maintained data table is not recommended for a production environment.

Defining DFHBRNSF

For VSAM files and data tables you will need to use IDCAMS to create file DFHBRNSF.

Figure 12 shows a some sample IDCAMS statements that you can modify to create the DFHBRNSF file. The cluster name and volume values should be changed to comply with your own standards.

```
//DEFDS     JOB  accounting info,name,MSGCLASS=A
//TDINTRA   EXEC PGM=IDCAMS
//SYSPRINT  DD   SYSOUT=A
//SYSIN     DD   *
      DEFINE CLUSTER(NAME(CICSTS22.CICS.DFHBRNSF)-
               INDEXED-
               TRK(1 1)-
               RECORDSIZE(384 384)-
               KEYS(13 20)-
               FREESPACE(0 50)-
               SHAREOPTIONS(2 3)-
               LOG(NONE)-
               VOLUME(DISK01)-
               CISZ(512)) -
            DATA  (NAME(CICSTS22.CICS.DFHBRNSF.DATA)-
             CISZ(512)) -
           INDEX (NAME(CICSTS22.CICS.DFHBRNSF.INDEX)-
             CISZ(512))
/*
//
```

Figure 12. Sample IDCAMS job to create the DFHBRNSF file

See the *CICS System Definition Guide* for guidance on creating coupling facility data tables.

File DFHBRNSF contains two control records plus 1 record for each router region that accesses the file. The maximum number of records that can be written to DFHBRNSF is 731 (this includes the 2 control records).

You need to define file DFHBRNSF to CICS in the Link3270 router regions. Resource definitions have been provided for all versions of the file. You should include the appropriate group in your startup grouplist, or copy chosen definitions into a group in the grouplist. You will need to edit the definitions to match your IDCAMS statements. Change the DSN field to match the cluster name used with IDCAMS to create the file, unless the CFDT version of the file is to be used. If the CFDT definition is being used, change the CFDTPOOL value to the name of the pool containing the table defined by the file definition. The table below shows the groups provided that contain the DFHBRNSF definitions.

Table 8. Supplied resource definitions for DFHBRNSF

Group	Type
DFHBRVR	VSAM RLS
DFHBRVSL	Local VSM, non-RLS
DFHBRVSR	Remote VSAM, non-RLS
DFHBRCF	Coupling facility data table
DFHBRUT	User maintained data table

Note:

1. If DFHBRNSF becomes unavailable, only Link3270 requests that do not cause an allocation or release of a bridge facility namespace range will still complete successfully.
2. If DFHBRNSF file has to be redefined for any reason, all router regions that access the file should be shut down before the file is redefined, and restarted after the file has been redefined.

DFHBRNSF at CICS termination

During the normal termination of a CICS router region, new Link3270 requests are rejected and existing Link3270 facilities are released. When all facilities have been released, the DFHBRNSF name ranges associated with the router region are freed.

It is possible that the freeing of the DFHBRNSF name ranges will fail. For example, if file DFHBRNSF is remote, the shutdown transaction CESD breaks the connection to the file owning region before the name ranges have been freed. If this happens an error message will be issued. The name ranges are only freed when the CICS region is restarted and a Link3270 transaction has run in that region.

If the CICS termination is immediate, the Link3270 request is rejected with return code BRIH-CICS-TERMINATION, but the Link3270 facilities and DFHBRNSF name ranges associated with the facilities are not released. The name ranges are only freed when the CICS region is restarted and a Link3270 transaction has run in that region.

Defining a specific bridge facility name

If the name or netname of the 3270 terminal is important to the logic of the 3270 application, you can supply a specific name in the BRIH-TERMINAL or BRIH-NETNAME parameter on the Link3270 call and also optionally request that the autoinstall user replaceable module (URM) is called when the bridge facility is allocated.

The autoinstall URM is called if you specify the system initialization parameter AIBRIDGE=YES at CICS startup, or use SET AUTOINSTALL to activate this option at a later time.

The AUTOINSTALL URM can accept, reject or modify the supplied or generated terminal name and netname. See the *CICS Customization Guide* for information about the autoinstall URM.

Initializing the TCTUA

The bridge facility can have a TCTUA (Terminal Control Table User Area), which can be accessed by **EXEC CICS ADDRESS TCTUA** in the normal way.

The TCTUA is initialized to nulls when the bridge facility is created. A global user exit (GLUE) called XFAINTU is called when a bridge facility is created and discarded. XFAINTU is passed the address of the TCTUA, so you can use this exit to initialize the TCTUA. For information about the XFAINTU global user exit, see Global user exit points (by function) in the Customization Guide.

Accessing bridge facility properties

The user transaction can retrieve information about its principal facility (the bridge facility) from the EIB or by using INQUIRE and ASSIGN commands, in exactly the same way that it does when running normally, where the principal facility is a real 3270.

For example, the TERMID can be obtained from EIBTERMID or from an ASSIGN FACILITY, INQUIRE TASK FACILITY or INQUIRE NETNAME command, and the NETNAME can be obtained with ASSIGN NETNAME or INQUIRE TERMINAL.

You can use the INQUIRE BRFACILITY command to obtain information about any bridge facility, identified by its facilitytoken, but all other INQUIRE commands return only information about the bridge facility that is the principal facility of the transaction issuing the command. To other transactions, a transaction running in a bridged environment appears to be a non-terminal transaction, and an INQUIRE TERMID against a bridge facility TERMID issued by another transaction will result in TERMIDERR. INQUIRE NETNAME and INQUIRE TASK behave similarly.

Bridge facilities do not appear in response to INQUIRE TERMINAL browses.

All keywords of ASSIGN and INQUIRE are supported and return the values that have been set for the bridge facility from the FACILITYLIKE terminal definition, or that have been set during the execution of the transaction.

Some keywords return values fixed by CICS for the bridge environment. These are:

Table 9. INQUIRE TERMINAL values

Keyword	Returned value
ACQSTATUS	ACQUIRED
ACCESSMETHOD	VTAM
CORRELID	blanks
EXITTRACING	NOTAPPLIC
LINKSYSTEM	blanks
MODENAME	blanks
REMOTENAME	blanks
REMOTESYSTEM	blanks

Table 9. INQUIRE TERMINAL values (continued)

Keyword	Returned value
REMOTESYSNET	blanks
SERVSTATUS	INSERVICE
TCAMCONTROL	X'FF'
TERMSTATUS	ACQUIRED
TTISTATUS	YES
ZCPTRACING	NOZCPTRACE

Note: VTAM is now the z/OS Communications Server.

Table 10. INQUIRE TASK values

Keyword	Returned value
FACILITY	the bridge facility name
FACILITYTYPE	TERM or TASK
STARTCODE	S,SD,TO,TP

QUERY

The keywords listed represent terminal attributes that can be set by the 3270 Query function at logon time for a real device.

ALTSCRNHT	ALTSCRNWD	APLKYBDST	APLTEXTST
BACKTRANSST	COLORST	EXTENDEDDSST	GCHARS
GCODES	HILIGHTST	MSRCONTROLST	OUTLINEST
PARTITIONSST	PROGSYMBOLST	SOSIST	VALIDATIONST

If the real FACILITYLIKE terminal is logged on when the bridge facility is created, the QUERY will have been performed and the values returned will apply to the bridge facility.

If the real FACILITYLIKE terminal is not logged on at the time that the bridge facility is created, the QUERY will not have been performed and the bridge facility will be created using values from the FACILITYLIKE resource definition.

SET TERMINAL/NETNAME

The following table shows the effect of each of the SET TERMINAL/NETNAME keywords when issued by a user transaction for its bridge facility. Unless otherwise specified, the response is DFHRESP(NORMAL).

KEYWORD	EFFECT
ACQSTATUS	Ignored.
ALTPRINTER	Value is SET, and is returned on INQUIRE, but is never used by CICS.
ALTPRTCOPYST	Value is SET, and is returned on INQUIRE, but is never used by CICS.
ATISTATUS	Works as for normal 3270.
CANCEL	Ignored
CREATESESS	Ignored.
DISCREQST	Value is SET, and is returned on INQUIRE, but is never used by CICS.
EXITTRACING	Ignored.
FORCE	Ignored.
MAPNAME	Works as for normal 3270.
MAPSETNAME	Works as for normal 3270.
NEXTTRANSID	Works as for normal 3270.
OBFORMATST	Works as for normal 3270.
PAGESTATUS	Ignored.
PRINTER	Value is SET, and is returned on INQUIRE, but is never used by CICS.
PRTCOPYST	Value is SET, and is returned on INQUIRE, but is never used by CICS.
PURGE	Ignored.
PURGETYPE	Ignored.
RELREQST	Value is SET, and is returned on INQUIRE, but is never used by CICS.
SERVSTATUS	Works as for normal 3270.
TCAMCONTROL	Returns INVREQ, as for normal 3270.
TERMPRIORITY	Value is SET, and is returned on INQUIRE, but is never used by CICS.
TERMSTATUS	Ignored.
TRACING	Value is SET, and is returned on INQUIRE, but is never used by CICS.
TTISTATUS	Ignored.
UCTRANST	Works as for normal 3270.
ZCPTRACING	Ignored.

Managing Link3270 bridge resources

You can use these commands and interfaces to obtain information about the
Link3270 bridge environment and the bridge facility.
- INQUIRE/SET AUTOINSTALL
- INQUIRE/SET BRFACILITY
- INQUIRE TERMINAL/NETNAME
- INQUIRE TASK
- INQUIRE/SET TRACETYPE

- INQUIRE TRANSACTION
- CEMT
- The exit programming interface (XPI)

See System commands in CICS System Programming Reference for details of INQUIRE and SET commands, and CEMT - master terminal in CICS Supplied Transactions for information about the CEMT command.

Note: The BRIDGE option of the ASSIGN command returns the name of the transaction that issued the Link3270 command. It returns blanks if the transaction is not run in a bridge environment.

INQUIRE/SET AUTOINSTALL with the Link3270 bridge

You can use the AIBRIDGE option of the **INQUIRE AUTOINSTALL** command and **CEMT INQUIRE AUTOINSTALL** to indicate whether the autoinstall URM is called when bridge facilities are allocated. You can change this setting using the **SET AUTOINSTALL** command and **CEMT SET AUTOINSTALL**.

Related reference

INQUIRE AUTOINSTALL in CICS System Programming Reference

CEMT INQUIRE AUTOINSTALL in CICS Supplied Transactions

SET AUTOINSTALL in CICS System Programming Reference

CEMT SET AUTOINSTALL in CICS Supplied Transactions

INQUIRE/SET BRFACILITY with the Link3270 bridge

You can use the **INQUIRE BRFACILITY** command and **CEMT INQUIRE BRFACILITY** to return information about a bridge facility.
- The terminal name associated with the bridge facility (TERMID)
- The netname associated with the bridge facility
- The name of the user transaction running with this bridge facility
- The number of the task running the user transaction
- The applid of the AOR
- The sysid of the AOR
- The applid of the router region
- The sysid of the router region
- An indicator showing whether the Link3270 or START BREXIT bridge mechanism is being used
- The length of time this bridge facility will be kept if unused
- The current status of the bridge facility

You can change the status setting of the bridge facility to RELEASED using the **SET BRFACILITY** command and **CEMT SET BRFACILITY**.

Related reference

INQUIRE BRFACILITY in CICS System Programming Reference

CEMT INQUIRE BRFACILITY in CICS Supplied Transactions

SET BRFACILITY in CICS System Programming Reference

CEMT SET BRFACILITY in CICS Supplied Transactions

INQUIRE TASK with the Link3270 bridge

You can use the BRFACILITY option of the **EXEC CICS INQUIRE TASK** or **CEMT INQUIRE TASK** command to return an 8-byte field containing the facility token for the bridge facility in use by the task.

Note: The BRIDGE and IDENTIFIER options return information about the START BREXIT bridge mechanism, and are not used with Link3270.

Related reference

➡ INQUIRE TASK in CICS System Programming Reference

➡ CEMT INQUIRE TASK in CICS Supplied Transactions

INQUIRE/SET TRACETYPE with the Link3270 bridge

You can use the **INQUIRE TRACETYPE** command to indicate whether tracing is enabled for the bridge (BR) and partner (PT) domains. You can change this setting using the **SET TRACETYPE** command .

Related reference

➡ INQUIRE TRACETYPE in CICS System Programming Reference

➡ SET TRACETYPE in CICS System Programming Reference

INQUIRE TRANSACTION with the Link3270 bridge

You can use the FACILITYLIKE option of the **INQUIRE TRANSACTION** command and **CEMT INQUIRE TRANSACTION** to return the 4-character name of the terminal defined by the FACILITYLIKE parameter of the PROFILE associated with the named transaction resource definition.

If FACILITYLIKE is not defined, blanks are returned.

Note: The BREXIT option returns information about the START BREXIT bridge mechanism, and is not used with Link3270.

Related reference

➡ INQUIRE TRANSACTION in CICS System Programming Reference

➡ CEMT INQUIRE TRANSACTION in CICS Supplied Transactions

XPI commands for the Link3270 bridge

You can use the INQUIRE_CONTEXT function of the DFHBRIQX call to return the following information for the Link3270 bridge.

- The name of the bridge exit program used by a task running the START BREXIT bridge mechanism.
- The bridge facilitytoken associated with a user transaction running in a bridge environment.
- The address of the bridge facility. This has the same format as a TCTTE and can be mapped using the DSECT DFHTCTTE.
- The name of the bridge monitor transaction used to start a user transaction using the START BREXIT bridge mechanism.
- A token that contains the address of the bridge exit area used by a task running the START BREXIT bridge mechanism.
- The type of environment in which the transaction is running. This can be :

NORMAL
A transaction that is not running in a bridge environment.

BRIDGE
A user transaction that was started using a bridge.

BREXIT
A bridge exit program.

See the *CICS Customization Guide* for full details of the INQUIRE-CONTEXT interface.

Using Link3270 bridge load routing

The Link3270 bridge mechanism extends the dynamic routing capability of base CICS to support dynamic routing of 3270 bridge transactions.

Figure 13 on page 58 shows that the DPL request from the Link3270 bridge program to a remote user transaction can be sent to any of a number of application owning regions (AORs) where the user transaction is enabled.

The dynamic transaction routing user replaceable program (URM) is called when Link3270 bridge program DFHL3270 needs to identify the AOR to which an eligible transaction should be routed. New parameters on the interface allow the dynamic transaction routing program to identify Link3270 bridge requests and obtain the names of the target transaction and the bridge facility token. You can write your own transaction routing program to exploit these new parameters, use the CICS supplied dynamic routing program (DFHDYP), or use workload routing services provided by CICSPlex System Manager.
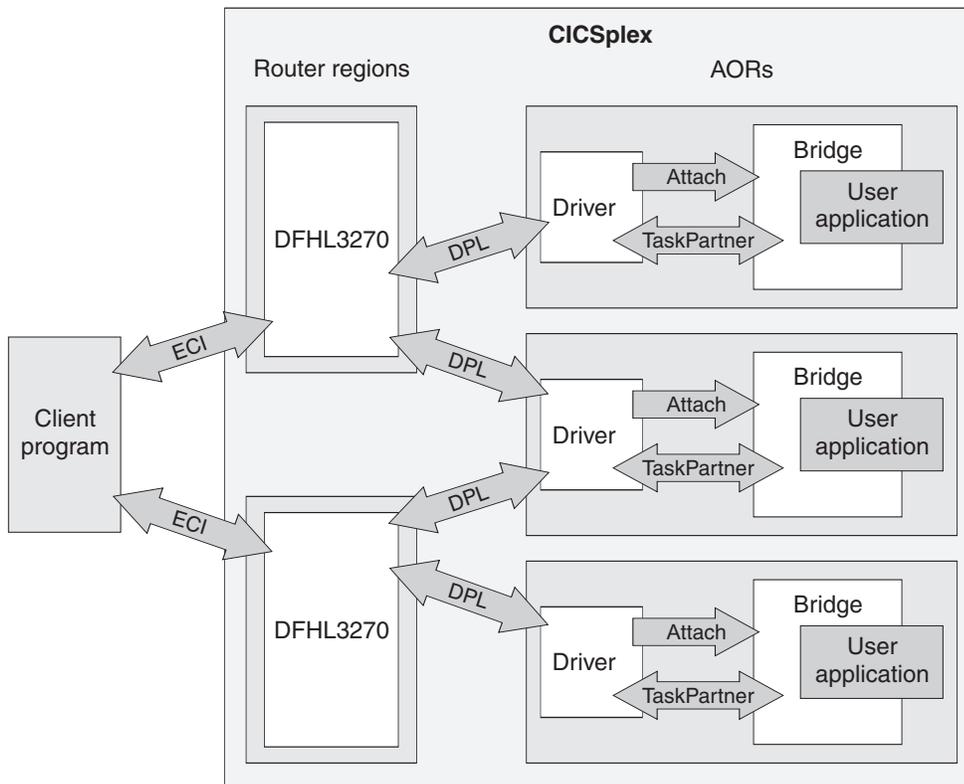
Figure 13. Link3270 load balancing

Using the dynamic transaction routing program with Link3270

When the dynamic transaction routing user replaceable program is called in the Link3270 bridge environment, these input parameters are set.

DYRTYPE
> 8

DYRBRTK
> bridge facilitytoken

DYRTRAN
> name of user transaction as known in the router region

When a client links to the bridge routing program (DFHL3270) with the first application transaction for that facility and that transaction is defined as dynamic, the dynamic transaction routing program is called to determine if the request should be routed (using DPL) to another server region. The dynamic transaction routing program is passed the transaction id in the message, and determines the SYSID of the region (AOR) where the user transaction will be started.

The dynamic transaction routing program is only called if the transaction is defined as DYNAMIC(YES).

Once a bridge transaction has been routed successfully to an AOR, all transactions executing with the same FACILITYTOKEN are routed to the same AOR. This affinity continues until the bridge facility is deleted in the AOR.

In session mode, subsequent transactions that are defined as dynamic will cause a notify call to the dynamic transaction routing program, informing the routing program that the transaction request is being routed to a specific region.

See the *CICS Customization Guide* for information about using a dynamic routing program.

Chapter 5. Link3270 message formats

This topic contains Product sensitive Programming Interface and Associated Guidance Information.

Link3270 message components

Link3270 messages contain the following components:
- Link3270 message header (BRIH)
- Inbound Link3270 vectors
- Outbound Link3270 vectors
- The application data structure (ADS)

Copybook names and descriptions

To help simplify the programming of clients, CICS provides copybooks and header files defining the BRIH and BRIV data structures in Assembler, COBOL, PLI and C. Defaults are provided for each inbound vector. These vectors are used to initialize the input message. There are two versions of the copybooks. When the basic copybooks are used, which are listed in Link3270 message copybooks for basic supportCopybooks and default vectors, the current version is set to indicate basic support. The extended copybooks, which are listed in Link3270 message copybooks for extended support, provide extended support, and when they are used the current version is set to indicate extended support.

The copybook names for the basic copybooks are:

DFHBRIHx
> BRIH and inbound and outbound BRIVs for C, PLI, and Assembler

DFHBRIIx
> Inbound BRIVs for COBOL

DFHBRIOx
> Outbound BRIVs for COBOL

DFHBRICx
> Constants and default values

and the names for the extended copybooks are:

DFHBR2Hx
> BRIH and inbound and outbound BRIVs for C, PLI, and Assembler

DFHBR2Ix
> Inbound BRIVs for COBOL

DFHBR2Ox
> Outbound BRIVs for COBOL

DFHBR2Cx
> Constants and default values

where x is the language suffix:

D   Assembler

H	C			
L	PLI			
O	COBOL			

Field names

Field names are shown in this documentation with "-" (dash) separators, as used in earlier versions of COBOL supported by CICS. Other languages use _ (underscore) separators.

Constants

Constants are provided for all enumerated values (input and output). For COBOL, these are provided as level 88 in the copybook.

Link3270 message header (BRIH)

The Link3270 bridge message header prefixes all input and output messages.

describes the values of fields used for input. describes the values of fields used for output.

Table 11. The BRIH message header on input

Offset Hex	Type	Len	Name	Default
(0)	STRUCTURE	180	BRIH	
(0)	CHARACTER	4	BRIH-STRUCID	BRIH-STRUC-ID
(4)	FULLWORD	4	BRIH-VERSION	BRIH-CURRENT-VERSION
(8)	FULLWORD	4	BRIH-STRUCLENGTH	BRIH-CURRENT-LENGTH
(C)	n/a	36	reserved	
(30)	FULLWORD	4	BRIH-GETWAITINTERVAL	BRIHGWI-MAXWAIT
(34)	n/a	4	reserved	
(38)	FULLWORD	4	BRIH-DATALENGTH	BRIH-CURRENT-LENGTH
(3C)	FULLWORD	4	BRIH-FACILITYKEEPTIME	BRIHKT-DEFAULT
(40)	FULLWORD	4	BRIH-ADSDESCRIPTOR	BRIHADSD-YES
(44)	FULLWORD	4	BRIH-CONVERSATIONALTASK	BRIHCT-NO
(48)	n/a	4	reserved	
(4C)	CHARACTER	8	BRIH-FACILITY	BRIHFACT-NEW

Table 11. The BRIH message header on input (continued)

Offset Hex	Type	Len	Name	Default
(54)	n/a	40	reserved	
(7C)	CHARACTER	4	BRIH-TRANSACTIONID	
(80)	CHARACTER	4	BRIH-FACILITYLIKE	BRIHFACL-DEFAULT
(84)	CHARACTER	4	BRIH-ATTENTIONID	DFHENTER
(88)	CHARACTER	4	BRIH-STARTCODE	BRIHSC-TERMINPUT
(8C)	CHARACTER	4	BRIH-CANCELCODE	blanks
(90)	n/a	4	reserved	
(94)	CHARACTER	8	BRIH-NETNAME	BRIHNN-DEFAULT
(9C)	CHARACTER	4	BRIH-TERMINAL	BRIHTN-DEFAULT
(A0)	n/a	4	reserved	
(A4)	FULLWORD	4	BRIH-CURSORPOSITION	BRIHCP-DEFAULT
(A8)	n/a	12	reserved	

Inbound BRIH message header

The fields that are used in an input message are listed. You can supply values in these fields; other fields are ignored on input. A BRIH structure primed with input default values (BRIH-DEFAULT) is supplied in the DFHBRICx copybooks. If a default value is not specified, the field is initialized to nulls.

Fields are valid on all calls, except where indicated. See also "Using Link3270 single transaction mode" on page 36 and "Using Link3270 session mode" on page 37.

BRIH-STRUCID
> The identifier for the header structure. You must set this to BRIH-STRUC-ID, which is the default.

BRIH-VERSION
> The version number for the header structure. You must set this to BRIH-CURRENT-VERSION, which is the default. Refer to "Link3270 bridge basic and extended support" on page 34 for a description of the different levels of support for the Link3270 bridge.

BRIH-STRUCLENGTH
> The length of the header structure. You must set this to BRIH-CURRENT-LENGTH, which is the default.

BRIH-DATALENGTH
> The length of the input message, including the BRIH. The default is
> BRIH-CURRENT-LENGTH.

BRIH-TRANSACTIONID
> The transaction identifier of the user transaction, as defined in the routing
> region. In session mode, this can also specify the following request values:

BRIHT-ALLOCATE-FACILITY
> Allocate a new bridge facility

BRIHT-DELETE-FACILITY
> Delete an existing bridge facility

BRIHT-CONTINUE-CONVERSATION
> Reply to a conversational request message

BRIHT-GET-MORE-MESSAGE
> Obtain the remainder (or next section) of the Link3270 message. This
> applies if the COMMAREA in the original request was too small to
> accommodate the output message.

BRIHT-RESEND-MESSAGE
> Resend the previous saved Link3270 message. This is used if the
> communications connection is broken in executing the previous request.
> See "Recovery from connection failure" on page 41 for further information.
>
> **Note:** A message is not saved if an error occurs before the message starts
> the user transaction. The BRIH-SEQNO can be used to determine whether
> a message returned by the RESEND-MESSAGE command is from the last
> Link3270 request issued, or the previous request.

BRIH-FACILITY
> The facilitytoken of the bridge facility. For single transaction mode this must be
> set to BRIHFACT-NEW. For session mode, on allocation, this is set to
> BRIHFACT-NEW. For subsequent requests, this must be set to the value
> returned on the allocate.
>
> The default is BRIHFACT-NEW.

BRIH-FACILITYLIKE
> *(Single transaction mode and allocation of a bridge facility in session mode)*
>
> The name of an installed terminal that is to be used as a model for the bridge
> facility. If no value is supplied in single-transaction mode, and a
> FACILITYLIKE value has been specified in the PROFILE definition of the user
> transaction, this value is used. Otherwise, or if no value is specified in session
> mode, a CICS-supplied definition, CBRF, is used.
>
> The default BRIHFACL-DEFAULT means that no value is specified.

BRIH-NETNAME
> *(Single transaction mode and allocation of a bridge facility in session mode)*
>
> The NETNAME to be assigned to the bridge facility.
>
> The default value, BRIHNN-DEFAULT, causes CICS to generate a name. The
> name is subject to change or rejection by the autoinstall URM whether
> specified by the user or generated by CICS. The name, as modified, is returned
> in this field in the response from the Link3270 bridge.

If you are specifying your own BRIH-NETNAME, the valid character set is the same as that for the NETNAME attribute of the CICS TERMINAL definition. See TERMINAL resources in the Resource Definition Guide.

BRIH-TERMINAL
(Single transaction mode and allocation of a bridge facility in session mode)

The TERMID to be assigned to the bridge facility.

The default value, BRIHTN-DEFAULT, causes CICS to generate a name. The name is subject to change or rejection by the autoinstall URM whether specified by the user or generated by CICS. The name, as modified, is returned in this field in the response from the Link3270 bridge.

If you are specifying your own BRIH-TERMINAL, the valid character set is the same as that for the TERMINAL attribute of the CICS TERMINAL definition. See TERMINAL resources in the Resource Definition Guide.

If you plan to specify your own BRIH-TERMINAL and to allow BRIH-NETNAME to default to this, you must use the BRIH-NETNAME character set, which is more restricted.

BRIH-FACILITYKEEPTIME
(Allocation of a bridge facility in session mode)

The length of time that the bridge facility is kept after the user transaction has ended (in seconds). The value used is the smaller of this value, and the value specified in the router region's SIT parameter BRMAXKEEPTIME.

The default is BRIHKT-DEFAULT.

BRIH-CONVERSATIONALTASK
(Run transaction in session mode)

An indicator specifying what the Link3270 bridge should do if the user transaction issues an input command for which no input vector has been provided. Possible values are:

BRIHCT-YES
The Link3270 bridge suspends the transaction and adds a request vector to the end of the output message. The client is expected to send a CONTINUE-CONVERSATION message containing the requested vector.

BRIHCT-NO
The Link3270 bridge abends the user transaction.

The default is BRIHCT-NO.

BRIH-GETWAITINTERVAL
(Run transaction in session mode)

The maximum wait interval for message input (in milliseconds). The value used is the smaller of the BRIH-GETWAITINTERVAL and the RTIMEOUT value for the transaction.

This value is used only when BRIH-CONVERSATIONALTASK is BRIHCT-YES

The default is BRIHGWI-MAXWAIT.

BRIH-CANCELCODE
(Session mode - continue conversation only)

The abend code with which the Link3270 bridge is to terminate a user transaction. This value is meaningful only in CONTINUE-CONVERSATION

messages. If it is non-blank, Link3270 l abends the suspended user transaction with an abend code of BRIH-CANCELCODE. It should be used only when the client wants to terminate the transaction rather than supply the requested vector.

BRIH–ADSDESCRIPTOR
(Single transaction mode and run transaction in session mode)

An indicator specifying whether ADS descriptors are sent on outbound SEND MAP and RECEIVE MAP messages. Possible values are:

BRIHADSD-YES
ADS descriptors are sent.

BRIHADSD-NO
ADS descriptors are not sent.

The default is BRIHADSD-YES.

BRIH–ATTENTIONID
(Single transaction mode and run transaction in session mode)

The initial value of the AID key (EIBAID) when the user transaction is started. This is a 1-byte value, left justified. EIBAID is reset after each RECEIVE, RECEIVE, or CONVERSE command from the AID value in the input vector

The default is DFHENTER (The enter key).

BRIH–STARTCODE
(Single transaction mode and first run transaction in a session)

An indicator available to the first transaction in a session to show the type of start that the request is emulating. The value generated depends on whether there is a RETRIEVE vector present in the input. Possible values are:

BRIHSC-START
START command

BRIHSC-TERMINPUT
Terminal input

The default is BRIHSC-TERMINPUT.

BRIH–CURSORPOSITION
(Single transaction mode and run transaction in session mode)

The initial cursor position, EIBCPOSN, when the transaction is started. EIBCPOSN is reset from the value in the input vector after every RECEIVE, RECEIVE MAP or CONVERSE command.

The default is BRIHCP-DEFAULT, the top left of the screen.

Outbound BRIH message header

Table 12. The output BRIH message header

Offset Hex	Type	Len	Name
(0)	STRUCTURE	180	BRIH
(0)	CHARACTER	4	BRIH-STRUCID
(4)	FULLWORD	4	BRIH-VERSION

Table 12. The output BRIH message header (continued)

Offset Hex	Type	Len	Name
(8)	FULLWORD	4	BRIH-STRUCLENGTH
(C)	n/a	20	reserved
(20)	BINARY	4	BRIH-RETURNCODE
(24)	BINARY	4	BRIH-COMPCODE
(28)	BINARY	4	BRIH-REASON
(2C)	n/a	8	reserved
(34)	BINARY	4	BRIH-REMAININGDATALENGTH
(38)	FULLWORD	4	BRIH-DATALENGTH
(3C)	n/a	12	reserved
(48)	FULLWORD	4	BRIH-TASKENDSTATUS
(4C)	CHARACTER	8	BRIH-FACILITY
(54)	CHARACTER	4	BRIH-FUNCTION
(58)	CHARACTER	4	BRIH-ABENDCODE
(5C)	**CHARACTER**	**4**	**BRIH-SYSID**[1]
(60)	n/a	28	reserved
(7C)	CHARACTER	4	BRIH-TRANSACTIONID
(80)	n/a	16	reserved
(90)	CHARACTER	4	BRIH-NEXTTRANSACTIONID
(94)	CHARACTER	8	BRIH-NETNAME
(9C)	CHARACTER	4	BRIH-TERMINAL
(A0)	FULLWORD	8	BRIH-NEXTTRANIDSOURCE
(A8)	FULLWORD	4	BRIH-ERROROFFSET
(AC)	FULLWORD	4	BRIH-SEQNO
(B0)	n/a	4	reserved

Note:

1. BRIH-SYSID is available only for the Link3270 bridge with extended support.

The following fields are returned in an output message. Other fields are not relevant.

BRIH-RETURNCODE
Return code from the Link3270 interface. See "BRIH-RETURNCODE values" on page 98 for a list of possible return codes, and their associated BRIH-COMPCODE and BRIH-REASON values.

BRIH-COMPCODE
Additional error information. See "BRIH-RETURNCODE values" on page 98 for a list of possible return codes, and their associated BRIH-COMPCODE and BRIH-REASON values.

BRIH-REASON
Additional error information. See "BRIH-RETURNCODE values" on page 98 for a list of possible return codes, and their associated BRIH-COMPCODE and BRIH-REASON values.

BRIH-REMAININGDATALENGTH
(Session mode)

The length of the remaining message if the COMMAREA is too small to return the complete outbound message. The remaining message is prefixed by another BRIH (included in the length). If there is no more data , this field is set to zero. See "Delivering large messages" on page 41 for information about processing large messages.

BRIH-DATALENGTH
The length of the output message, including the BRIH.

BRIH-TASKENDSTATUS
The status of the user transaction. Possible values are:

> **BRIHTES-CONVERSATION**
> The user transaction has issued an input command for which no vector has been supplied, and BRIH-CONVERSATIONALTASK was specified in the inbound BRIH header.

> **BRIHTES-ENDTASK**
> The user transaction has ended (or abended).

BRIH-FACILITY
This value identifies the session. It is set on return from an allocate request and must be supplied on every subsequent request in the session. On return from a delete-facility request or a run request in single-transaction mode, it is reset to BRIHFACT-NEW.

BRIH-FUNCTION
Additional error information returned for some return codes. See "BRIH-RETURNCODE values" on page 98 for details.

BRIH-ABENDCODE
The abend code returned if the transaction abends. If the transaction completed successfully, this is set to BRIHAC-NONE.

Transaction abends are indicated by the return code BRIHAC-APPLICATION-ABEND. See "BRIH-RETURNCODE values" on page 98 for details.

BRIH-SYSID
The region in which the transaction ran. This is the system ID of the AOR as it is known by the routing region. If the transaction ran in the routing region, this field is set to blanks. This field is available only for the Link3270 bridge

with extended support. See "Link3270 bridge basic and extended support" on page 34 for a description of the different levels of support for the Link3270 bridge.

BRIH-TRANSACTIONID
> BRIH-TRANSACTIONID is both an input and an output field. Normally the output value is the same as the input value. The exceptions to this are:
> 1. When the request is for message recovery and the input BRIH-TRANSACTIONID is set to BRIHT-RESEND-MESSAGE. See "Recovery from connection failure" on page 41 for further information.
> 2. When the router region resource definition of the transaction is an alias of the definition in the AOR, the transaction id in the AOR is returned.

BRIH-NEXTTRANSACTIONID
> The name of the next transaction returned by the user transaction (usually by EXEC CICS RETURN TRANSID). If there is no next transaction, this field is set to blanks.

BRIH-NETNAME
> *(Allocation only)*
>
> The NETNAME assigned to the bridge facility.

BRIH-TERMINAL
> *(Allocation only)*
>
> The TERMID assigned to the bridge facility.

BRIH-NEXTTRANIDSOURCE
> The source of the next transaction id. Possible values are:
>
> **BRIHNTS-NORMAL**
>> Created by the TRANSID option of an EXEC CICS RETURN command, or by SET TERMINAL NEXTTRANSID.
>
> **BRIHNTS-IMMEDIATE**
>> Created by the TRANSID option of an EXEC CICS RETURN IMMEDIATE command.
>
> **BRIHNTS-STARTED**
>> Created by the TRANSID option of an EXEC CICS START command.

BRIH-ERROROFFSET
> The offset from the start of the message to the location of the invalid data for message validation errors.

BRIH-SEQNO
> *(Session mode only)*
>
> A sequence number returned on every message. The sequence number is set to 0 on an allocate facility request and incremented on subsequent requests. The exceptions to this are:
> 1. A successful BRIHT-RESEND-MESSAGE request returns the previous message and its sequence number.
> 2. If BRIHRC-INVALID-FACILITY-TOKEN is returned, the sequence number is undefined.

Inbound Link3270 vectors

Inbound Link3270 bridge vectors all have a common header.

Supported inbound vector types

Table "Link3270 inbound vector header" shows the common header. One BRIV vector is required to satisfy each input CICS command issued by the user transaction. The following inbound vector types are supported:

- "Link3270 INPUT CONVERSE vector"
- "Link3270 RECEIVE vector" on page 72
- "Link3270 RECEIVE MAP vector" on page 73
- "Link3270 RETRIEVE vector" on page 75

Link3270 inbound vector header

This header precedes all the vectors (both inbound and outbound) in the message.

Offset Hex	Type	Len	Name
(0)	STRUCTURE	16	BRIV-INPUT-VECTOR-HEADER
(0)	FULLWORD	4	BRIV-INPUT-VECTOR-LENGTH
(4)	CHARACTER	4	BRIV-INPUT-VECTOR-DESCRIPTOR
(8)	CHARACTER	4	BRIV-INPUT-VECTOR-TYPE
(C)	n/a	4	reserved

BRIV-INPUT-VECTOR-LENGTH
The length of the vector. This is rounded up to the next multiple of 4, to facilitate full word alignment of subsequent vectors in the message. The default is the length of the default BRIV with no data.

BRIV-INPUT-VECTOR-DESCRIPTOR
An indicator to define the CICS command associated with this vector. Valid values are:

BRIVDSC-CONVERSE (0406)
CONVERSE

BRIVDSC-RECEIVE (0402)
RECEIVE

BRIVDSC-RECEIVE- MAP (1802)
RECEIVE MAP

BRIVDSC-RETRIEVE (100A)
RETRIEVE

BRIV-INPUT-VECTOR-TYPE
This must be set to BRIVVT-INBOUND. This is the default.

Link3270 INPUT CONVERSE vector

This vector is used to supply data to an **EXEC CICS CONVERSE** command.

See CONVERSE (3270 logical) in CICS Application Programming for details of the command options.

The default vector is BRIV-CONVERSE-DEFAULT

Offset Hex	Type	Len	Name
(0)	STRUCTURE	36	BRIV-CONVERSE
(0)	STRUCTURE	16	INPUT header
(10)	CHARACTER	4	BRIV-CO-TRANSMIT-SEND-AREAS
(14)	CHARACTER	4	reserved
(18)	CHARACTER	4	BRIV-CO-AID
(1C)	FULLWORD	4	BRIV-CO-CPOSN
(20)	FULLWORD	4	BRIV-CO-DATA-LEN
(24)	CHARACTER		BRIV-CO-DATA

BRIV-CO-TRANSMIT-SEND-AREAS
This flag is a performance option that allows the client to limit the amount of data returned in the output message. Valid values are:

BRIVCOTSA-YES
The whole output message is returned.

BRIVCOTSA-NO
All output vectors created before the command that uses this vector are not returned in the output message.

The default is BRIVCOTSA-YES.

BRIV-CO-AID
The AID key used to transmit the input. This value is used to set EIBAID on completion of the RECEIVE MAP command. The first byte of this field contains equivalent values to EIBAID, as defined by DFHAID. The remaining three bytes are ignored. The default is DFHENTER.

BRIV-CO-CPOSN
The position of the cursor in the data. This value is used to set EIBCPOSN on completion of the RECEIVE MAP command. Valid values are:

BRIVCOCP-DEFAULT
top left of the screen

BRIVCOCP-MAX-CURSORPOSITION
bottom right of the screen

nn User specified value

The default is BRIVCOCP-DEFAULT.

BRIV-CO-DATA-LEN
The length of the data provided in this vector in BRIV-CO-DATA. This value is copied into the LENGTH or FLENGTH field specified in the CONVERSE command represented by this vector.

The default is zero (no data).

BRIV-CO-DATA
> Character field of length BRIV-CO-DATA-LEN to be copied into the INTO area, or referenced by the SET option, of the CONVERSE command represented by this vector.

Link3270 RECEIVE vector

This vector is used to supply data to an **EXEC CICS RECEIVE** command.

See for details of the command options.

The default vector is BRIV-RECEIVE-DEFAULT

Offset Hex	Type	Len	Name
(0)	STRUCTURE	36	BRIV-RECEIVE
(0)	STRUCTURE	16	INPUT header
(10)	CHARACTER	4	BRIV-RE-TRANSMIT-SEND-AREAS
(14)	CHARACTER	4	BRIV-RE-BUFFER-INDICATOR
(18)	CHARACTER	4	BRIV-RE-AID
(1C)	FULLWORD	4	BRIV-RE-CPOSN
(20)	FULLWORD	4	BRIV-RE-DATA-LEN
(24)	CHARACTER		BRIV-RE-DATA

BRIV-RE-TRANSMIT-SEND-AREAS
> This flag is a performance option that allows the client to limit the amount of data returned in the output message. Valid values are:
>
> **BRIVRETSA-YES**
>> The whole output message is returned.
>
> **BRIVRETSA-NO**
>> All output vectors created before the command that uses this vector are not returned in the output message.
>
> The default is BRIVRETSA-YES.

BRIV-RE-BUFFER-INDICATOR
> A flag indicating whether the data provided in the inbound vector is in a format to be received by a CICS RECEIVE command with the BUFFER option. Valid values are:
>
> **BRIVREBI-YES**
>> Data in BUFFER format.
>
> **BRIVREBI-NO**
>> Data not in BUFFER format.
>
> The default is BRIVREBI-NO.

BRIV-RE-AID

The AID key used to transmit the input. This value is used to set EIBAID on completion of the RECEIVE MAP command. The first byte of this field contains equivalent values to EIBAID, as defined by DFHAID. The remaining three bytes are ignored.

The default is DFHENTER.

BRIV-RE-CPOSN

The position of the cursor in the data. This value is used to set EIBCPOSN on completion of the RECEIVE MAP command. Valid values are:

BRIVRECP-DEFAULT

BRIVRECP-MAX-CURSORPOSITION

nn   User specified value

The default is BRIVRECP-DEFAULT.

BRIV-RE-DATA-LEN

The length of the data provided in this vector in BRIV-RE-DATA. This value is copied into the LENGTH or FLENGTH field specified in the RECEIVE command represented by this vector.

The default is zero (no data).

BRIV-RE-DATA

Character field of length BRIV-RE-DATA-LEN to be copied into the INTO area, or referenced by the SET option, of the RECEIVE command represented by this vector.

Link3270 RECEIVE MAP vector

This vector is used to supply data to an **EXEC CICS RECEIVE MAP** command.

See RECEIVE MAP in CICS Application Programming for details of the command options.

The default vector is BRIV-RECEIVE-MAP-DEFAULT

Offset Hex	Type	Len	Name
(0)	STRUCTURE	48	BRIV-RECEIVE-MAP
(0)	STRUCTURE	16	INPUT header
(10)	CHARACTER	4	BRIV-RM-TRANSMIT-SEND-AREAS
(14)	CHARACTER	8	BRIV-RM-MAPSET
(1C)	CHARACTER	8	BRIV-RM-MAP
(24)	CHARACTER	4	BRIV-RM-AID
(28)	FULLWORD	4	BRIV-RM-CPOSN
(2C)	FULLWORD	4	BRIV-RM-DATA-LEN

Offset Hex	Type	Len	Name
(30)	CHARACTER		BRIV-RM-DATA

BRIV-RM-TRANSMIT-SEND-AREAS
   This flag is a performance option that allows the client to limit the amount of data returned in the output message. Valid values are:

   BRIVRMTSA-YES
      The whole output message is returned.

   BRIVRMTSA-NO
      All output vectors created before the command that uses this vector are not returned in the output message.

   The default is BRIVRMTSA-YES.

BRIV-RM-MAPSET
   The name of the MAPSET containing the map used to format the data, or blanks. When the user transaction issues a RECEIVE MAP command, the Link3270 bridge uses the first remaining RECEIVE MAP vector in the message in which BRIV-RM-MAPSET matches MAPSET in the command or is blank and BRIV-RM-MAP matches the MAP in the command or is blank. RECEIVE MAP vectors which do not match the command are discarded.

   The default is blanks.

BRIV-RM-MAP
   The name of the MAP containing the map used to format the data, or blanks. When the user transaction issues a RECEIVE MAP command, the Link3270 bridge uses the first remaining RECEIVE MAP vector in the message in which BRIV-RM-MAPSET matches MAPSET in the command or is blank and BRIV-RM-MAP matches the MAP in the command or is blank. RECEIVE MAP vectors which do not match the command are discarded

   The default is blanks.

BRIV-RM-AID
   The AID key used to transmit the input. This value is used to set EIBAID on completion of the RECEIVE MAP command. The first byte of this field contains equivalent values to EIBAID, as defined by DFHAID. The remaining three bytes are ignored. The default is DFHENTER.

BRIV-RM-CPOSN
   The position of the cursor in the data. This value is used to set EIBCPOSN on completion of the RECEIVE MAP command. Valid values are:

   BRIVRMCP-DEFAULT

   BRIVRMCP-MAX-CURSORPOSITION

   nn   User specified value

   The default is BRIVRMCP-DEFAULT.

BRIV-RM-DATA-LEN
   The length of the Application Data Structure (ADS) in BRIV-RM-DATA. This value is copied into the LENGTH or FLENGTH field specified in the RECEIVE MAP command represented by this vector.

BRIV-RM-DATA
> The ADS to be copied into the INTO area, or referenced by the SET option, of the RECEIVE MAP command represented by this vector.

Link3270 RETRIEVE vector

This vector is used to supply data to an **EXEC CICS RETRIEVE** command.

See RETRIEVE in CICS Application Programming for details of the command options.

The default vector is BRIV-CONVERSE-DEFAULT

Offset Hex	Type	Len	Name
(0)	STRUCTURE	36	BRIV-RETRIEVE
(0)	STRUCTURE	16	INPUT header
(10)	CHARACTER	4	BRIV-RT-RTRANSID
(14)	CHARACTER	4	BRIV-RT-RTERMID
(18)	CHARACTER	8	BRIV-RT-QUEUE
(20)	FULLWORD	4	BRIV-RT-DATA-LEN
(24)	CHARACTER		BRIV-RT-DATA

BRIV-RT-RTRANSID
> The value to be returned in the RTRANSID field to the program that issued the RETRIEVE. A blank indicates that there is no RTRANSID. The default is blank.

BRIV-RT-RTERMID
> The value to be returned in the RTERMID field to the program that issued the RETRIEVE. A blank indicates that there is no RTERMID. The default is blank.

BRIV-RT-QUEUE
> The value to be returned in the QUEUE field to the program that issued the RETRIEVE. A blank indicates that there is no QUEUE. The default is blank.

BRIV-RT-DATA-LEN
> The length of the data provided in this vector in BRIV-RT-DATA that caused the bridge to be called. This value is copied into the LENGTH or FLENGTH field specified in the RETRIEVE command represented by this vector. The default is zero (no data).

data
> Character field of length BRIV-RT-DATA-LEN to be copied into the INTO area, or referenced by the SET option of the RETRIEVE command represented by this vector.

Note: The RETRIEVE vector is only valid in the first inbound message in session mode, or in single transaction mode. It is ignored in other messages.

Outbound Link3270 vectors

Outbound Link3270 bridge vectors all have a common header.

Table "Link3270 output vector header" shows the common header. One BRIV vector is required for each output EXEC CICS command issued by the user transaction. The following outbound vector types are supported:

- "Link3270 ISSUE ERASEAUP vector" on page 77
- "Link3270 SEND vector" on page 77
- "Link3270 SEND CONTROL vector" on page 79
- "Link3270 SEND MAP vector" on page 82
- "Link3270 SEND TEXT vector" on page 85
- "Link3270 SYNCPOINT vector" on page 89
- "Link3270 CONVERSE request vector" on page 90
- "Link3270 RECEIVE request vector" on page 92
- "Link3270 RECEIVE MAP request vector" on page 92
- "Link3270 SEND PAGE vector" on page 88
- "Link3270 PURGE MESSAGE vector" on page 89

Link3270 output vector header

This header precedes all the vectors in the message.

Offset Hex	Type	Len	Name
(0)	STRUCTURE	16	BRIV-OUTPUT-VECTOR-HEADER
(0)	FULLWORD	4	BRIV-OUTPUT-VECTOR-LENGTH
(4)	CHARACTER	4	BRIV-OUTPUT-VECTOR-DESCRIPTOR
(8)	CHARACTER	4	BRIV-OUTPUT-VECTOR-TYPE
(C)	n/a	4	reserved

BRIV-OUTPUT-VECTOR-LENGTH
> The length of the vector. This is rounded up to the next multiple of 4, to facilitate full word alignment of subsequent vectors in the message.

BRIV-OUTPUT-VECTOR-DESCRIPTOR
> An indicator to define the CICS command associated with this vector. Valid values are:

> **BRIVDSC-ISSUE-ERASEAUP (0418)**
>> ISSUE ERASEAUP

> **BRIVDSC-SEND (0404)**
>> SEND

> **BRIVDSC-SEND-MAP (1804)**
>> SEND MAP

> **BRIVDSC-SEND-TEXT (1806)**
>> SEND TEXT

> **BRIVDSC-SEND-CONTROL (1812)**
> > SEND CONTROL
>
> **BRIVDSC-SYNCPOINT (1602)**
> > SYNCPOINT
>
> **BRIVDSC-CONVERSE-REQUEST (0406)**
> > CONVERSE request
>
> **BRIVDSC-RECEIVE-REQUEST (0402)**
> > RECEIVE request
>
> **BRIVDSC-RECEIVE-MAP-REQUEST (1802)**
> > RECEIVE MAP request
>
> **BRIVDSC-SEND-PAGE (1808)**
> > SEND PAGE
>
> **BRIVDSC-PURGE-MESSAGE (180A)**
> > PURGE MESSAGE
>
> `BRIV-OUTPUT-VECTOR-TYPE`
> > This must be set to BRIVVT-OUTBOUND. This is the default.

Link3270 ISSUE ERASEAUP vector

This vector is the data supplied by an **EXEC CICS ISSUE ERASEAUP** command.

See ISSUE ERASEAUP in CICS Application Programming for details of the command options.

Offset Hex	Type	Len	Name
(0)	STRUCTURE	20	BRIV-ISSUE-ERASEAUP
(0)	STRUCTURE	16	Output header
(10)	CHARACTER	4	BRIV-IE-WAIT-INDICATOR

`BRIV-IE-WAIT-INDICATOR`
> The presence of the WAIT option on the ISSUE ERASEAUP command that caused the bridge to be called. Valid values are:
>
> **BRIVIEWI-YES**
> > WAIT specified.
>
> **BRIVIEWI-NO**
> > WAIT not specified.

Link3270 SEND vector

This vector is the data supplied by an **EXEC CICS SEND** command, or the output part of an **EXEC CICS CONVERSE** command, for which a RECEIVE vector was supplied.

For more information about these commands, see SEND (3270 logical) in CICS Application Programming and CONVERSE (VTAM default) in CICS Application Programming.

Offset Hex	Type	Len	Name
(0)	STRUCTURE	48	BRIV-SEND
(0)	STRUCTURE	16	Output header
(10)	CHARACTER	4	BRIV-SE-ERASE-INDICATOR
(14)	CHARACTER	4	BRIV-SE-CTLCHAR
(18)	CHARACTER	4	BRIV-SE-STRFIELD-INDICATOR
(1C)	CHARACTER	4	BRIV-SE-DEFRESP-INDICATOR
(20)	CHARACTER	4	BRIV-SE-INVITE-INDICATOR
(24)	CHARACTER	4	BRIV-SE-LAST-INDICATOR
(28)	CHARACTER	4	BRIV-SE-WAIT-INDICATOR
(2C)	FULLWORD	4	BRIV-SE-DATA-LEN
(30)	CHARACTER		BRIV-SE-DATA

BRIV-SE-ERASE-INDICATOR

The type of ERASE specified by the CICS **SEND** or **CONVERSE** command. Valid values are:

BRIVSEEI-NOERASE
> No ERASE.

BRIVSEEI-ERASE
> ERASE.

BRIVSEEI-ERASEALTERNATE
> ERASE ALTERNATE.

BRIVSEEI-DEFAULT
> ERASE DEFAULT.

BRIV-SE-CTLCHAR

The CTLCHAR value specified by the **SEND** or **CONVERSE** command. Valid values are:

cc   The value in CTLCHAR.

BRIVSECC-DEFAULT
> X'C3'.

BRIV-SE-STRFIELD-INDICATOR

The presence of STRFIELD on the **SEND** or **CONVERSE** command. Valid values are:

BRIVSESI-YES
> STRFIELD specified.

BRIVSESI-NO
> STRFIELD not specified.

BRIV-SE-DEFRESP-INDICATOR
> The presence of DEFRESP on the **SEND** or **CONVERSE** command. Valid values are:

> **BRIVSEDRI-YES**
>> DEFRESP specified.

> **BRIVSEDRI-NO**
>> DEFRESP not specified.

BRIV-SE-INVITE-INDICATOR
> The presence of INVITE on the **SEND** or **CONVERSE** command. Valid values are:

> **BRIVSEII-YES**
>> INVITE specified.

> **BRIVSEII-NO**
>> INVITE not specified.

BRIV-SE-LAST-INDICATOR
> The presence of LAST on the **SEND** or **CONVERSE** command. Valid values are:

> **BRIVSELI-YES**
>> LAST specified.

> **BRIVSELI-NO**
>> LAST not specified.

BRIV-SE-WAIT-INDICATOR
> The presence of WAIT on the **SEND** or **CONVERSE** command. Valid values are:

> **BRIVSEWI-YES**
>> WAIT specified.

> **BRIVSEWI-NO**
>> WAIT not specified.

BRIV-SE-DATA-LEN
> The length of the data associated with the FROM option of the **SEND** or **CONVERSE** command. This is explicitly defined in the `LENGTH` or `FLENGTH` option, or derived from the length of the field.

BRIV-SE-DATA
> Character field of length BRIV-SE-DATA-LEN containing the data addressed by the FROM option of the **SEND** or **CONVERSE** command.

Link3270 SEND CONTROL vector

This vector is the data supplied by an EXEC CICS SEND CONTROL command.

This vector is the data supplied by an **EXEC CICS SEND CONTROL** command. See SEND CONTROL in CICS Application Programming for details of the command options.

Offset Hex	Type	Len	Name
(0)	STRUCTURE	52	BRIV-SEND-CONTROL
(0)	STRUCTURE	16	Output header
(10)	CHARACTER	4	BRIV-SC-ERASE-INDICATOR

Offset Hex	Type	Len	Name
(14)	CHARACTER	4	BRIV-SC-ERASEAUP-INDICATOR
(18)	CHARACTER	4	BRIV-SC-FREEKB-INDICATOR
(1C)	CHARACTER	4	BRIV-SC-ALARM-INDICATOR
(20)	CHARACTER	4	BRIV-SC-FRSET-INDICATOR
(24)	CHARACTER	4	BRIV-SC-LAST-INDICATOR
(28)	CHARACTER	4	BRIV-SC-WAIT-INDICATOR
(2C)	FULLWORD	4	BRIV-SC-CURSOR
(30)	CHARACTER	4	BRIV-SC-MSR-DATA
(34)	**CHARACTER**	**4**	**BRIV-SC-ACCUM-INDICATOR[1]**

Note:

1. BRIV-SC-ACCUM-INDICATOR is available only for the Link3270 bridge with extended support.

BRIV-SC-ERASE-INDICATOR
> The type of ERASE specified by the CICS SEND CONTROL command. Valid values are:

> **BRIVSCEI-NOERASE**
>> No ERASE.

> **BRIVSCEI-ERASE**
>> ERASE.

> **BRIVSCEI-ERASEALTERNATE**
>> ERASE ALTERNATE.

> **BRIVSCEI-DEFAULT**
>> ERASE DEFAULT.

BRIV-SC-ERASEAUP-INDICATOR
> The presence of ERASEAUP on the SEND CONTROL command. Valid values are:

> **BRIVSCEUI-YES**
>> ERASEAUP specified.

> **BRIVSCEUI-NO**
>> ERASEAUP not specified.

BRIV-SC-FREEKB-INDICATOR
> The presence of FREEKB on the SEND CONTROL command. Valid values are:

> **BRIVSCFKI-YES**
>> FREEKB specified.

> **BRIVSCFKI-NO**
>> FREEKB not specified.

BRIV-SC-ALARM-INDICATOR
The presence of ALARM on the SEND CONTROL command. Valid values are:

BRIVSCAI-YES
ALARM specified.

BRIVSCAI-NO
ALARM not specified.

BRIV-SC-FRSET-INDICATOR
The presence of FRSET on the SEND CONTROL command. Valid values are:

BRIVSCFSI-YES
FRSET specified.

BRIVSCFSI-NO
FRSET not specified.

BRIV-SC-LAST-INDICATOR
The presence of LAST on the SEND CONTROL command. Valid values are:

BRIVSCLI-YES
LAST specified.

BRIVSCLI-NO
LAST not specified.

BRIV-SC-WAIT-INDICATOR
The presence of WAIT on the SEND CONTROL command. Valid values are:

BRIVSCWI-YES
WAIT specified.

BRIVSCWI-NO
WAIT not specified.

BRIV-SC-CURSOR
The presence of CURSOR(*data-value*) on the SEND CONTROL command. Valid values are:

BRIVSCCRS-DYNAMIC
CURSOR specified with dynamic cursor positioning.

BRIVSCCRS-NONE
CURSOR(*data-value*) not specified.

nn   The value of CURSOR(*data-value*) specified.

BRIV-SC-MSR-DATA
The value of the MSR option specified on the SEND CONTROL command.
Valid values are:

BRIVSCMSR-NONE
MSR option not specified.

other The value of the MSR option specified.

BRIV-SC-ACCUM-INDICATOR
Indicates whether or not the ACCUM option is specified for **EXEC CICS SEND TEXT**, **EXEC CICS SEND MAP**, or **EXEC CICS SEND CONTROL**. This parameter is only available for the Link3270 bridge with extended support. See "Link3270 bridge basic and extended support" on page 34 for a description of the different levels of support for the Link3270 bridge. Values are:

Y   The ACCUM option is specified.

N   The ACCUM option is not specified.

Link3270 SEND MAP vector

This vector is the data supplied by an **EXEC CICS SEND MAP** command.

See SEND MAP in CICS Application Programming for details of the command options.

Offset Hex	Type	Len	Name
(0)	STRUCTURE	88	BRIV-SEND-MAP
(0)	STRUCTURE	16	Output header
(10)	CHARACTER	4	BRIV-SM-ERASE-INDICATOR
(14)	CHARACTER	4	BRIV-SM-ERASEAUP-INDICATOR
(18)	CHARACTER	4	BRIV-SM-FREEKB-INDICATOR
(1C)	CHARACTER	4	BRIV-SM-ALARM-INDICATOR
(20)	CHARACTER	4	BRIV-SM-FRSET-INDICATOR
(24)	CHARACTER	4	BRIV-SM-LAST-INDICATOR
(28)	CHARACTER	4	BRIV-SM-WAIT-INDICATOR
(2C)	FULLWORD	4	BRIV-SM-CURSOR
(30)	CHARACTER	4	BRIV-SM-MSR-DATA
(34)	CHARACTER	8	BRIV-SM-MAPSET
(3C)	CHARACTER	8	BRIV-SM-MAP
(44)	CHARACTER	4	BRIV-SM-DATA-INDICATOR
(48)	FULLWORD	4	BRIV-SM-DATA-LEN
(4C)	FULLWORD	4	BRIV-SM-DATA-OFFSET
(50)	FULLWORD	4	BRIV-SM-ADSD-LEN
(54)	FULLWORD	4	BRIV-SM-ADSD-OFFSET
(58)	**FULLWORD**	**4**	**BRIV-SM-ACCUM-INDICATOR[1]**
	CHARACTER		BRIV-SM-DATA[2]

Note:

1. BRIV-SM-ACCUM-INDICATOR is available only for the Link3270 bridge with extended support.

2. BRIV-SM-DATA is deprecated and included only for compatibility with the basic support version of the Link3270 bridge provided by CICS Transaction Server for z/OS, Version 2 Release 2. The recommended method for addressing this field is by using BRIV-SM-DATA-OFFSET. Use BRIV-SM-ADSD-OFFSET to address ADSD data.

BRIV-SM-ERASE-INDICATOR
> The type of ERASE specified by the CICS SEND MAP command. Valid values are:

> **BRIVSMEI-NOERASE**
>> No ERASE.

> **BRIVSMEI-ERASE**
>> ERASE.

> **BRIVSMEI-ERASEALTERNATE**
>> ERASE ALTERNATE.

> **BRIVSMEI-DEFAULT**
>> ERASE DEFAULT.

BRIV-SM-ERASEAUP-INDICATOR
> The presence of ERASEAUP on the SEND MAP command. Valid values are:

> **BRIVSMEUI-YES**
>> ERASEAUP specified.

> **BRIVSMEUI-NO**
>> ERASEAUP not specified.

BRIV-SM-FREEKB-INDICATOR
> The presence of FREEKB on the SEND MAP command. Valid values are:

> **BRIVSMFKI-YES**
>> FREEKB specified.

> **BRIVSMFKI-NO**
>> FREEKB not specified.

BRIV-SM-ALARM-INDICATOR
> The presence of ALARM on the SEND MAP command. Valid values are:

> **BRIVSMAI-YES**
>> ALARM specified.

> **BRIVSMAI-NO**
>> ALARM not specified.

BRIV-SM-FRSET-INDICATOR
> The presence of FRSET on the SEND MAP command. Valid values are:

> **BRIVSMFSI-YES**
>> FRSET specified.

> **BRIVSMFSI-NO**
>> FRSET not specified.

BRIV-SM-LAST-INDICATOR
> The presence of LAST on the SEND MAP command. Valid values are:

> **BRIVSMLI-YES**
>> LAST specified.

> **BRIVSMLI-NO**
>> LAST not specified.

BRIV-SM-WAIT-INDICATOR

The presence of WAIT on the SEND MAP command. Valid values are:

BRIVSMWI-YES

WAIT specified.

BRIVSMWI-NO

WAIT not specified.

BRIV-SM-CURSOR

The presence of CURSOR or CURSOR(*data-value*) on the SEND MAP command. Valid values are:

BRIVSMCRS-DYNAMIC

CURSOR specified (dynamic cursor positioning).

BRIVSMCCRS-NONE

Neither CURSOR nor CURSOR(*data-value*) specified.

nn   The value of CURSOR(*data-value*) specified.

BRIV-SM-MSR-DATA

The value of the MSR option specified on the SEND MAP command. Valid values are:

BRIVSMMSR-NONE

MSR option not specified.

other   The value of the MSR option specified.

BRIV-SM-MAPSET

The value of the MAPSET option specified by the SEND MAP command.

BRIV-SM-MAP

The value of the MAP option specified by the SEND MAP command.

BRIV-SM-DATA-INDICATOR

The presence of MAPONLY and DATAONLY options on the SEND MAP command. Valid values are:

BRIVSMDI-DATAONLY

DATAONLY specified.

BRIVSMDI-MAPONLY

MAPONLY specified.

BRIVSMDI-DEFAULT

Neither DATAONLY nor MAPONLY specified.

BRIV-SM-DATA-LEN

The length of the data in BRIV-SM-DATA. This is the length of the symbolic map or ADS (application data structure).

BRIV-SM-DATA-OFFSET

The offset from the beginning of the SEND MAP vector to the data associated with the FROM option of the SEND MAP command.

BRIV-SM-ADSD-LEN

The length of the ADS descriptor associated with this map. This length is zero if the ADSD is not available or was not requested (BRIH-ADSDESCRIPTOR set to BRIHADSD-NONE).

BRIV-SM-ADSD-OFFSET

The offset from the beginning of the SEND MAP vector to the ADSD. This is zero if the ADSD is not available or was not requested.

BRIV-SM-ACCUM-INDICATOR
Indicates whether or not the ACCUM option is specified for EXEC CICS SEND TEXT, EXEC CICS SEND MAP, or EXEC CICS SEND CONTROL. This parameter is only available for the Link3270 bridge with extended support. See "Link3270 bridge basic and extended support" on page 34 for a description of the different levels of support for the Link3270 bridge. Values are:

Y   The ACCUM option is specified.

N   The ACCUM option is not specified.

BRIV-SM-DATA
This field is included only for compatibility with the basic support version of the Link3270 bridge. (See "Link3270 bridge basic and extended support" on page 34 for a description of the different levels of support for the Link3270 bridge.) It is recommended that you use BRIV-SM-DATA-OFFSET to address this field.

Link3270 SEND TEXT vector

This vector is the data supplied by an **EXEC CICS SEND TEXT** command.

See SEND TEXT in CICS Application Programming for details of the command options.

Offset Hex	Type	Len	Name
(0)	STRUCTURE	60	BRIV-SEND-TEXT
(0)	STRUCTURE	16	Output header
(10)	CHARACTER	4	BRIV-ST-ERASE-INDICATOR
(14)	CHARACTER	4	FILLER
(18)	CHARACTER	4	BRIV-ST-FREEKB-INDICATOR
(1C)	CHARACTER	4	BRIV-ST-ALARM-INDICATOR
(20)	**CHARACTER**	**4**	**BRIV-ST-ACCUM-INDICATOR**[1]
(24)	CHARACTER	4	BRIV-ST-LAST-INDICATOR
(28)	CHARACTER	4	BRIV-ST-WAIT-INDICATOR
(2C)	FULLWORD	4	BRIV-ST-CURSOR
(30)	CHARACTER	4	BRIV-ST-MSR-DATA
(34)	CHARACTER	4	BRIV-ST-TEXT-TYPE
(38)	FULLWORD	4	BRIV-ST-DATA-LEN
(3C)	**FULLWORD**	**4**	**BRIV-ST-DATA-OFFSET**[1]
(40)	**FULLWORD**	**4**	**BRIV-ST-HEADER-LEN**[1]

Offset Hex	Type	Len	Name
(44)	FULLWORD	4	BRIV-ST-HEADER-OFFSET[1]
(48)	FULLWORD	4	BRIV-ST-TRAILER-LEN[1]
(4C)	FULLWORD	4	BRIV-ST-TRAILER-OFFSET[1]
50	CHARACTER		BRIV-ST-DATA[2]

Note:

1. The fields in bold type are available only for the Link3270 bridge with extended support.
2. BRIV-ST-DATA is deprecated and included only for backward compatibility with the basic support version of the Link3270 bridge provided by CICS Transaction Server for z/OS, Version 2 Release 2. The recommended method for addressing this field is by using BRIV-ST-DATA-OFFSET.

BRIV-ST-ERASE-INDICATOR
> The type of ERASE specified by the CICS SEND TEXT command. Valid values are:

> **BRIVSTEI-NOERASE**
>> No ERASE.

> **BRIVSTEI-ERASE**
>> ERASE.

> **BRIVSTEI-ERASEALTERNATE**
>> ERASE ALTERNATE.

> **BRIVSTEI-DEFAULT**
>> ERASE DEFAULT.

BRIV-ST-FREEKB-INDICATOR
> The presence of FREEKB on the SEND TEXT command. Valid values are:

> **BRIVSTFKI-YES**
>> FREEKB specified.

> **BRIVSTFKI-NO**
>> FREEKB not specified.

BRIV-ST-ALARM-INDICATOR
> The presence of ALARM on the SEND TEXT command. Valid values are:

> **BRIVSTAI-YES**
>> ALARM specified.

> **BRIVSTAI-NO**
>> ALARM not specified.

BRIV-ST-ACCUM-INDICATOR
> Indicates whether or not the ACCUM option is specified for EXEC CICS SEND TEXT, EXEC CICS SEND MAP, or EXEC CICS SEND CONTROL. This parameter is only available for the Link3270 bridge with extended support. See "Link3270 bridge basic and extended support" on page 34 for a description of the different levels of support for the Link3270 bridge. Values are:

Y The ACCUM option is specified.

N The ACCUM option is not specified.

BRIV-ST-LAST-INDICATOR
The presence of LAST on the SEND TEXT command. Valid values are:

BRIVSTLI-YES
LAST specified.

BRIVSTLI-NO
LAST not specified.

BRIV-ST-WAIT-INDICATOR
The presence of WAIT on the SEND TEXT command. Valid values are:

BRIVSTWI-YES
WAIT specified.

BRIVSTWI-NO
WAIT not specified.

BRIV-ST-CURSOR
The presence of CURSOR(*data-value*) on the SEND TEXT command. Valid
values are:

BRIVSTCRS-DYNAMIC
CURSOR specified (dynamic cursor positioning).

BRIVSTCRS-NONE
CURSOR(*data-value*) not specified.

nn The value of CURSOR(*data-value*) specified.

BRIV-ST-MSR-DATA
The value of the MSR option specified on the SEND TEXT command. Valid
values are:

BRIVSTMSR-NONE
MSR option not specified.

other The value of the MSR option specified.

BRIV-ST-TEXT-TYPE
The presence of MAPPED or NOEDIT options on the SEND TEXT command.
Valid values are:

BRIVSTTT-MAPPED
MAPPED specified.

BRIVSTTT-NOEDIT
NOEDIT specified.

BRIVSTTT-DEFAULT
Neither MAPPED nor NOEDIT specified.

BRIV-ST-DATA-LEN
The length of the data in BRIV-ST-DATA[1].

BRIV-ST-DATA-OFFSET
The offset of the data from the start of the vector. This parameter is only
available for the Link3270 bridge with extended support. See "Link3270 bridge
basic and extended support" on page 34 for a description of the different levels
of support for the Link3270 bridge.

BRIV-ST-HEADER-LEN

The length of the text header. This parameter is only available for the Link3270 bridge with extended support. See "Link3270 bridge basic and extended support" on page 34 for a description of the different levels of support for the Link3270 bridge.

BRIV-ST-HEADER-OFFSET

The offset of the text header from the start of the vector. Use this value to address the header. This parameter is only available for the Link3270 bridge with extended support. See "Link3270 bridge basic and extended support" on page 34 for a description of the different levels of support for the Link3270 bridge.

BRIV-ST-TRAILER-LEN

The length of the text trailer. This parameter is only available for the Link3270 bridge with extended support. See "Link3270 bridge basic and extended support" on page 34 for a description of the different levels of support for the Link3270 bridge.

BRIV-ST-TRAILER-OFFSET

The offset of the text trailer from the start of the vector. Use this value to address the trailer. This parameter is only available for the Link3270 bridge with extended support. See "Link3270 bridge basic and extended support" on page 34 for a description of the different levels of support for the Link3270 bridge.

BRIV-ST-DATA

This field is included only for compatibility with the basic support version of the Link3270 bridge. See "Link3270 bridge basic and extended support" on page 34 for a description of the different levels of support for the Link3270 bridge. It is recommended that you use BRIV-ST-DATA-OFFSET to access the data contained in the FROM option of the SEND TEXT command.

Note:

1. If the MAPPED option is used, you must add a 4 byte page control area (PGA) to the end of the data. See SEND TEXT in CICS Application Programming for a description of the PGA. These 4 bytes are not included in BRIV-ST-DATA-LEN, but are included in BRIV-OUTPUT-VECTOR-LENGTH and BRIH-DATALENGTH.

Link3270 SEND PAGE vector

This vector is the data supplied by an EXEC CICS SEND PAGE command.

See SEND PAGE in CICS Application Programming for details of the command options. This vector is only available for the Link3270 bridge with extended support. See "Link3270 bridge basic and extended support" on page 34 for a description of the different levels of support for the Link3270 bridge.

Offset Hex	Type	Len	Name
(0)	STRUCTURE	88	BRIV-SEND-PAGE
(0)	STRUCTURE	16	Output header
(10)	CHARACTER	4	BRIV-PG-RELEASE-INDICATOR

Offset Hex	Type	Len	Name
(14)	CHARACTER	4	BRIV-PG-RETAIN-INDICATOR
(18)	CHARACTER	4	BRIV-PG-LAST-INDICATOR
(1C)	CHARACTER	4	BRIV-PG-TRANSID
(20)	FULLWORD	4	BRIV-PG-TRAILER-LEN
(24)	FULLWORD	4	BRIV-PG-TRAILER-OFFSET

BRIV-PG-RELEASE-INDICATOR
Indicates whether or not the RELEASE option is specified. Valid values are:

Y   RELEASE is specified.

N   RELEASE is not specified.

BRIV-PG-RETAIN-INDICATOR
Indicates whether or not the RETAIN option is specified. Valid values are:

Y   RETAIN is specified.

N   RETAIN is not specified.

BRIV-PG-LAST-INDICATOR
Indicates whether or not LAST is specified. Valid values are:

Y   LAST is specified.

N   LAST is not specified.

BRIV-PG-TRANSID
The name of the transaction to be used on the next message.

BRIV-PG-TRAILER-LEN
The length of the trailer.

BRIV-PG-TRAILER-OFFSET
The offset of the trailer from the start of the vector.

Link3270 PURGE MESSAGE vector

This vector is the data supplied by an **EXEC CICS PURGE MESSAGE** command.

See PURGE MESSAGE in CICS Application Programming for details of the command options. This vector is only available for the Link3270 bridge with extended support. See "Link3270 bridge basic and extended support" on page 34 for a description of the different levels of support for the Link3270 bridge.

There are no parameters for this vector.

Link3270 SYNCPOINT vector

This vector is the data supplied by an **EXEC CICS SYNCPOINT** command.

See SYNCPOINT in CICS Application Programming for details of the command options.

This vector is supplied when the application issues one of the following:

- An **EXEC CICS SYNCPOINT** command
- A CICS command such as **EXEC CICS SYNCONRETURN** or **EXEC CICS CREATE** which issues an implicit syncpoint
- An RMI request which issues an implicit syncpoint

Note: The vector is not supplied on the implicit syncpoint which occurs when a transaction completes.

Offset Hex	Type	Len	Name
(0)	STRUCTURE	20	BRIV-SYNCPOINT
(0)	STRUCTURE	16	Output header
(10)	CHARACTER	4	BRIV-SP-ROLLBACK
(14)	CHARACTER	4	BRIV-SP-EXPLICIT
(18)	FULLWORD	4	BRIV-SP-RESP

BRIV-SP-ROLLBACK
> The presence of ROLLBACK on the **EXEC CICS SYNCPOINT** command. Valid values are:

> **BRIVSPR-YES**
>> ROLLBACK specified.

> **BRIVSPR-NO**
>> ROLLBACK not specified.

BRIV-SP-EXPLICIT
> Whether the syncpoint was explicit (resulting from a SYNCPOINT command) or implicit (resulting from a CICS or RMI command which issues an implicit syncpoint). Refer to the description above for more information. Valid values are:

> **BRIVSPE-YES**
>> The SYNCPOINT command was issued.

> **BRIVSPE-NO**
>> The syncpoint was implicit.

BRIV-SP-RESP
> The EIBRESP value returned from the SYNCPOINT command.

Link3270 CONVERSE request vector

This vector is the data supplied by an **EXEC CICS CONVERSE** request that was issued by the user application, but there was no CONVERSE vector in the input message.

See CONVERSE (3270 logical) in CICS Application Programming for details of the command options.

Offset Hex	Type	Len	Name
(0)	STRUCTURE	48	BRIV-CONVERSE-REQUEST
(0)	STRUCTURE	16	Output header
(10)	CHARACTER	4	BRIV-COR-ERASE-INDICATOR
(14)	CHARACTER	4	BRIV-COR-CTLCHAR
(18)	CHARACTER	4	BRIV-COR-STRFIELD-INDICATOR
(1C)	CHARACTER	4	BRIV-COR-DEFRESP-INDICATOR
(20)	CHARACTER	12	(reserved)
(2C)	FULLWORD	4	BRIV-COR-DATA-LEN
(30)	CHARACTER		BRIV-COR-DATA

BRIV-COR-ERASE-INDICATOR
> The type of ERASE specified by the CICS CONVERSE command. Valid values are:

> **BRIVCOREI-NOERASE**
>> No ERASE.

> **BRIVCOREI-ERASE**
>> ERASE.

> **BRIVCOREI-ERASEALTERNATE**
>> ERASE ALTERNATE.

> **BRIVCOREI-DEFAULT**
>> ERASE DEFAULT.

BRIV-COR-CTLCHAR
> The CTLCHAR value specified by the CONVERSE command. Valid values are:

> **BRIVCORCC-DEFAULT**
>> X'C3'.

> **cc** The value in CTLCHAR.

BRIV-COR-STRFIELD-INDICATOR
> The presence of STRFIELD on the CONVERSE command. Valid values are:

> **BRIVCORSI-YES**
>> STRFIELD specified.

> **BRIVCORSI-NO**
>> STRFIELD not specified.

BRIV-COR-DEFRESP-INDICATOR
> The presence of DEFRESP on the CONVERSE command. Valid values are:

> **BRIVCORDRI-YES**
>> DEFRESP specified.

BRIVCORDRI-NO
DEFRESP not specified.

BRIV-COR-DATA-LEN
The length of the data in BRIV-COR-DATA. This is explicitly defined in the LENGTH or FLENGTH option, or derived from the length of the field.

BRIV-COR-DATA
Character field of length BRIV-COR-DATA-LEN containing the data addressed by the FROM option of the CONVERSE command.

Link3270 RECEIVE request vector

This vector is the data supplied by an **EXEC CICS RECEIVE** request.

See RECEIVE (3270 logical) in CICS Application Programming for details of the command options.

Offset Hex	Type	Len	Name
(0)	STRUCTURE	20	BRIV-RECEIVE-REQUEST
(0)	STRUCTURE	16	Output header
(10)	CHARACTER	4	BRIV-RER-BUFFER-INDICATOR

BRIV-RER-BUFFER-INDICATOR
The presence of BUFFER on the CICS RECEIVE command. Valid values are:

BRIVRERBI-YES
BUFFER specified.

BRIVRERBI-NO
BUFFER not specified.

Link3270 RECEIVE MAP request vector

This vector is the data supplied by an EXEC CICS RECEIVE MAP request.

See RECEIVE MAP in CICS Application Programming for details of the command options.

Offset Hex	Type	Len	Name
(0)	STRUCTURE	36	BRIV-RECEIVE-MAP-REQUEST
(0)	STRUCTURE	16	Output header
(10)	CHARACTER	8	BRIV-RMR-MAPSET
(18)	CHARACTER	8	BRIV-RMR-MAP
(20)	FULLWORD	4	BRIV-RMR-ADSD-LEN
(24)	CHARACTER		BRIV-RMR-ADSD

BRIV-RMR-MAPSET
   The value of the MAPSET option on the RECEIVE MAP command.

BRIV-RMR-MAP
   The value of the MAP option on the RECEIVE MAP command.

BRIV-RMR-ADSD-LEN
   The length of the ADS descriptor associated with this map. This length is zero if the ADSD is not available or was not requested (BRIH-ADSDESCRIPTOR set to BRIHADSD-NONE).

BRIV-RMR-ADSD
   The ADS descriptor associated with the requested map. No data is sent if BRIV-RMR-ADSD-LEN is zero.

Link3270 ADS descriptor

The ADS descriptor contains a header with general information about the map, and a field descriptor for every field that appears in the ADS, corresponding to every named field in the map definition macro. It can be located in the mapset from an offset field in DFHMAPDS.

ADS descriptor header

The ADS descriptor header contains general information about the map and a pointer to the first of a variable number of chained field descriptions.

Offset Hex	Type	Len	Name
(0)	STRUCTURE	38	ADS-DESCRIPTOR
(0)	HALFWORD	2	ADSD-LENGTH
(2)	CHARACTER	4	ADSD-EYECATCHER
(6)	HALFWORD	2	ADSD-MAP-INDEX
(8)	HALFWORD	2	ADSD-FIELD-COUNT
(A)	HALFWORD	2	ADSD-STRUCTURE-LENGTH
(C)	HALFWORD	2	ADSD-ATTRIBUTE-NUMBER
(E)	CHARACTER	12	ADSD-ATTRIBUTE-TYPE-CODES
(1A)	CHARACTER	1	ADSD-MAP-JUSTIFY-HOR
(1B)	CHARACTER	1	ADSD-MAP-JUSTIFY-VER
(1C)	HALFWORD	2	ADSD-MAP-STARTING-LINE
(1E)	HALFWORD	2	ADSD-MAP-STARTING-COLUMN
(20)	HALFWORD	2	ADSD-MAP-LINES
(22)	HALFWORD	2	ADSD-MAP-COLUMNS
(24)	CHARACTER	1	ADSD-WRITE-CONTROL-CHARACTER
(25)	CHARACTER	1	(reserved)
(26)	STRUCTURE		ADSD-FIRST-FIELD

ADSD-LENGTH
   The length of the ADS descriptor.

ADSD-EYECATCHER
   An eye-catcher ('ADSD') to identify this as an ADS descriptor.

ADSD-MAP-INDEX
   The index number of the map within the mapset.

ADSD-FIELD-COUNT
   The number of fields within the ADS; that is, the number of named fields in

the map definition. A separate field is counted for each element of an array defined with the OCCURS parameter, but subfields of group fields (GRPNAME) are not counted. The field count may be zero, in which case there are no field descriptors following the header.

ADSD-STRUCTURE-LENGTH
The length of the application data structure.

ADSD-ATTRIBUTE-NUMBER
The number of extended attributes in fields used in the map; that is, the number of attributes specified in DSATTS in the map definition.

ADSD-ATTRIBUTE-TYPE-CODES
a 1-character code for the attribute types in each field, in order, derived from DSATTS:

- C = COLOR
- P = PS
- H = HILIGHT
- V = VALIDN
- O = OUTLINE
- S = SOSI
- T = TRANSP

ADSD-MAP-JUSTIFY-HOR
The horizontal justification for the map, either L (LEFT) or R (RIGHT) from the JUSTIFY operand on the map definition.

ADSD-MAP-JUSTIFY-VER
The vertical justification for the map, from the JUSTIFY operand on the map definition. This can have the values F (FIRST), L (LAST), B (BOTTOM), or blank (no vertical JUSTIFY operand).

ADSD-MAP-STARTING-LINE
The starting line for the map, from the LINE operand on the DFHMDI macro, (LINE = NEXT gives a value of 255; LINE = SAME gives a value of 254.)

ADSD-MAP-STARTING-COLUMN
The starting column for the map, from the COLUMN operand on the DFHMDI macro. (COLUMN = NEXT gives a value of 255; COLUMN = SAME gives a value of 254.)

ADSD-MAP-LINES
The number of lines in the map from the SIZE operand.

ADSD-MAP-COLUMNS
The number of columns in the map from the SIZE operand.

ADSD-WRITE-CONTROL-CHAR
The 3270 encoded WCC derived from the CONTROL operand.

ADSD-FIRST-FIELD
The first field descriptor. The address of the first field descriptor in the ADSD (zero if ADSD-FIELD-COUNT is zero).

ADS field descriptor

After the header, the ADS descriptor contains a variable number of field descriptors.

Each field descriptor has the following format:

Offset Hex	Type	Len	Name
(0)	STRUCTURE	42	ADS-FIELD-DESCRIPTOR
(0)	CHARACTER	32	ADSD-FIELD-NAME
(20)	HALFWORD	2	ADSD-FIELD-NAME-LEN
(22)	HALFWORD	2	ADSD-OCCURS-INDEX
(24)	HALFWORD	2	ADSD-FIELD-OFFSET
(26)	HALFWORD	2	ADSD-FIELD-DATA-LEN
(28)	CHARACTER	1	ADSD-FIELD-JUSTIFY
(29)	CHARACTER	1	ADSD-FIELD-FILL-CHAR
(2A)	CHARACTER		ADSD-NEXT-FIELD

ADSD-FIELD-NAME

The unsuffixed field name padded with blanks on the right.

ADSD-FIELD-NAME-LEN

The number of characters in the field name.

ADSD-OCCURS-INDEX

When OCCURS is specified for a field definition there is a separate field descriptor for each element of the array, and ADSD-OCCURS-INDEX indicates the array index for the particular field. If OCCURS is not specified, then ADSD-OCCURS-INDEX is 0.

ADSD-FIELD-OFFSET

The offset of the field within the ADS. The offset is to the beginning of the (fullword) length field, and you must add 2 (for the length field) + 1 (for the 3270 attribute) + ADSD-ATTRIBUTE-NUMBER to obtain the offset of the data part of the field.

ADSD-FIELD-DATA-LEN

The length of the field in the ADS.

ADSD-FIELD-JUSTIFY

A 1-character field indicating whether the data is to be justified left 'L' or right 'R' if the supplied length is less than the length in the ADS.

ADSD-FIELD-FILL-CHAR

The character (blank or '0') to be used to pad the remainder of the field in the ADS.

ADSD-NEXT-FIELD

The next field descriptor. The address of ADSD-NEXT-FIELD can be used to update a pointer for the field descriptor.

Chapter 6. Link3270 diagnostics

Link3270 messages are subject to a number of validation stages.

The following types of validation error are described
- Invalid message
- Invalid BRIH
- Invalid facility
- Invalid BRIV
- Invalid application data

Return codes and abend codes provided to assist in diagnosis of errors. Note that the order in which checks are made is subject to change, and therefore should not be used as an interface. "BRIH-RETURNCODE values" on page 98 shows the possible values of BRIH-RETURNCODE and the contents of any related diagnostic fields (BRIH-COMPCODE and BRIH-REASON). Where no specific value is shown, these fields are set to 0.

Invalid Message

If a COMMAREA is passed to Link3270 that is too small to contain a BRIH, or does not have the appropriate BRIH header, this will result in a transaction abend code:

ABR4 No COMMAREA

ABR5 COMMAREA too small to contain BRIH

ABR6 COMMAREA does not contain BRIH

Invalid BRIH

Only relevant fields are validated on each request. If these are invalid, then BRIH-RETURNCODE is set to BRIHRC-VALIDATION-ERROR-BRIH and BRIH-ERROROFFSET points to the field in error. The system state is not changed by a validation error. Therefore user transactions are neither started nor abended.

Invalid bridge facility

If the facility token is invalid, or has expired, this will result in BRIH-RETURNCODE being set to BRIHRC-INVALID-FACILITYTOKEN. Facilities which have expired are described by the state errors.

Invalid BRIV

BRIVs are validated as they are used. Therefore if a BRIV is not used, it is not checked. If these are invalid then BRIH-RETURNCODE is set to BRIHRC-VALIDATION-ERROR-BRIV and BRIH-ERROROFFSET points to the field in error

The transaction is abended with an ABXF abend code. BRIH-ABENDCODE is set to this value.

Invalid Application data

Application data cannot be checked by the bridge. Incorrect data will give unexpected results that may result in transaction abends or erroneous processing. You should ensure that your client program creates the data correctly. If validation of the client data is essential, you can do this by creating a program in the router region that accepts the COMMAREA, validates the ADS and then passes it to the bridge with a link to Link3270.

BRIH-RETURNCODE values

BRIHRC-OK (0)
The request completed successfully.

BRIHRC-AI-LINK-FAILED (23)
The link to the autoinstall URM failed.

> **BRIH-COMPCODE**
>
> > **1** Set by router region code

BRIHRC-AI-REJECTED (22)
The terminal autoinstall URM rejected the bridge install request.

> **BRIH-COMPCODE**
>
> > **1** Set by router region code

BRIHRC-AI-NETNAME-INVALID (21)
The netname supplied by the terminal autoinstall URM is invalid.

> **BRIH-COMPCODE**
>
> > **1** Set by router region code

BRIHRC-AI-TERMID-INVALID (20)
The terminal id returned by the autoinstall URM is invalid.

> **BRIH-COMPCODE**
>
> > **1** Set by router region code

BRIHRC-APPLICATION-ABEND (160)
The user transaction abended. Additional diagnostics:

> **BRIH-ABENDCODE**
> The transaction abend code

BRIHRC-CICS-TERMINATION (66)
The CICS region is terminating and the Link3270 request has been rejected.

> **BRIH-COMPCODE**
>
> > **1** Set by router region code

BRIHRC-CLIENT-NETNAME-INVALID (24)
The client supplied an invalid netname.

> **BRIH-COMPCODE**
>
> > **1** Set by router region code

BRIHRC-CLIENT-TERMID-INVALID (25)
The client supplied an invalid terminal id.

> **BRIH-COMPCODE**
>
> > **1** Set by router region code

BRIHRC-DFHBRNSF-UNAVAILABLE (65)
File DFHBRNSF is unavailable.

> **BRIH-COMPCODE**
>
> > **1** Set by router region code

BRIHRC-FACILITYLIKE-INVALID (26)
The client supplied an invalid facilitylike.

BRIH-COMPCODE

> **2**   Set by router region code

BRIHRC-FACILITYTOKEN-IN-USE (63)
A transaction is already running with this facilitytoken.

BRIH-COMPCODE

> **1**   Set by router region code

> **2**   Set by driver code

BRIHRC-INVALID-BRIH-DATALENGTH (140)
The BRIH datalength value supplied by the client is not valid.

BRIH-COMPCODE

> **1**   Set by router region code

BRIHRC-INVALID-CONTINUE_REQ (143)
The message contained no BRIVs.

BRIH-COMPCODE

> **1**   Set by router region code

BRIHRC-INVALID-FACILITY-TOKEN (61)
The bridge facilitytoken in the BRIH is invalid.

BRIH-COMPCODE

> **1**   Set by router region code

> **2**   Set by driver code

BRIHRC-INVALID-KEEPTIME (142)
A KEEPTIME of zero was set on an allocate request.

BRIH-COMPCODE

> **1**   Set by router region code

BRIHRC-NO-DATA (120)
A BRIHT-GET-MORE-MESSAGE request failed because there was no more data to send.

BRIH-COMPCODE

> **1**   Set by router region code

> **2**   Set by driver code

BRIHRC-NO-FREE-NAME (62)
All bridge facilities are already allocated.

BRIH-COMPCODE

> **1**   Set by router region code

BRIHRC-NO-STORAGE (64)
Insufficient storage in either the router region or AOR to run the request.

BRIH-COMPCODE

> **2**   Set by driver code

BRIHRC-NOT-SHUTDOWN-ENABLED (80)
An attempt was made to run a transaction at shutdown that is not enabled for running at shutdown.

BRIH-COMPCODE

    2   Set by driver code

BRIHRC-PROFILE-NOT-FOUND (87)

The transaction PROFILE for the target transaction was not found.

BRIH-COMPCODE

    2   Set by partner code

BRIHRC-RETRIEVE-NOT-SUPPORTED (121)

Retrieve vectors are only supported in the initial request.

BRIH-COMPCODE

    2   Set by driver code

BRIHRC-ROUTING-BACKLEVEL-CICS (45)

The Link3270 request was routed to a back level CICS system that does not support Link3270. Additional diagnostics:

BRIH-COMPCODE
EIBRESP

BRIH-REASON
EIBRESP2

BRIH-FUNCTION
EIBFN

BRIHRC-ROUTING-CONNECTION (43)

The Link3270 request could not be routed to the remote region because of a connection error. Additional diagnostics:

BRIH-COMPCODE
EIBRESP

BRIH-REASON
EIBRESP2

BRIH-FUNCTION
EIBFN

BRIHRC-ROUTING-TERMERR (44)

The EXEC CICS LINK from the DFHL3270 to the AOR failed with TERMERR. Additional diagnostics:

BRIH-COMPCODE
EIBRESP

BRIH-REASON
EIBRESP2

BRIH-FUNCTION
EIBFN

BRIHRC-ROUTING-TRANDEF-ERROR (42)

The TRANSACTION resource definition in the routing region does not allow the transaction to be routed to the chosen target region.

BRIH-COMPCODE

    1   Set by router region code

BRIHRC-ROUTING-URM-LINK-FAILED (40)

The link to the dynamic routing URM failed. Additional diagnostics:

BRIH-COMPCODE

3	URM abended
4	AMODE error
5	No PROGRAM definition
6	Fetch error
7	Disabled
8	Program defined as remote

BRIHRC-ROUTING-URM-REJECTED (41)

The dynamic routing URM rejected the bridge routing request. Additional diagnostics:

BRIH-COMPCODE

3	Select rejected
4	Sysid not found
5	Sysid not in service
6	Allocate rejected
7	Queue purged
8	Function not shipped
9	Netname not found
10	Sysid/netname mismatch

BRIH-REASON

Return code from dynamic routing URM (DYRRETC).

BRIHRC-SECURITY-ERROR (100)

A Link3270 request in session mode has been issued with a different userid than that used in the request that allocated the bridge facility token.

BRIH-COMPCODE

1	Set by router region code
2	Set by driver code

BRIHRC-STATE-SYSTEM-ATTACH (82)

The user transaction can only be system attached, and so cannot be run under a bridge facility.

BRIH-COMPCODE

2	Set by driver code

BRIHRC-TRANSACTION-DISABLED (84)

The user transaction to be run under the bridge is disabled.

BRIH-COMPCODE

1	Set by router region code
2	Set by driver code
3	DTRTRAN disabled

BRIHRC-TRANSACTION-NOT-FOUND (85)

The user transaction to be run under the bridge was not found. Additional diagnostics:

BRIH-COMPCODE

> **1**   Set by router region code

> **2**   Set by driver code

> **3**   DTRTRAN rejected by routing program

BRIHRC-TRANSACTION-NOT-RUNNING (86)
> The next leg of a pseudoconversation cannot be run because there is no transaction running on the bridge facility.

BRIH-COMPCODE

> **1**   Set by router region code

> **2**   Set by driver code

BRIHRC-VALIDATION-ERROR-BRIV (141)
> A BRIV is invalid. BRIH-ERROROFFSET points to the field in error.

BRIH-COMPCODE

> **2**   Set by driver code

BRIHRC-ROUTER-BACKLEVEL
> The router region does not support the version of the Link3270 message.

BRIH-COMPCODE

> **1**   Set by router region code

BRIHRC-AOR-BACKLEVEL
> The bridge driver task in the AOR does not support the version of the Link3270 message.

BRIH-COMPCODE

> **2**   Set by driver code

Chapter 7. Using the Link3270 samples

CICS provides sample client programs that use the ECI, EXCI and LINK interfaces to call the Link3270 bridge to run the sample transaction NACT.

The main objective of the sample programs is to provide coded examples that you can use to help you write your own client programs. NACT was chosen for this purpose as it has a well documented BMS interface.

The samples are not written to illustrate how a business client should process the data, so the business clients do not perform any special formatting of the data extracted from the user application.

The samples are designed to illustrate the two most common scenarios:

1. **Host Client**

   The client program executes on the host system, using LINK or EXCI to drive the user application. In this scenario, the sample programs show how you can divide the client logic into a *business-client* that is concerned only with the business data and its representation in the client end-user environment, and a *bridge-client* that builds the bridge messages and manages the communication with the bridge. In this way, you can develop the more complex back-end using CICS , and can make it reusable.

   The LINK and EXCI samples show how this common logic can be shared.

   See Figure 7 on page 30 for an illustration of the host client.

2. **Workstation Client**

   The client program executes on a remote workstation, using ECI to drive the user application. In this scenario, a single sample program is used, combining the business logic in the client environment and the interface to the bridge. In this environment, the programmer needs some, but not extensive, CICS knowledge.

   See Figure 9 on page 31 for an illustration of the workstation client.

The following sample client programs and copybooks are supplied in source code, in the SDFHSAMP library.

Lang.	LINK	ECI	EXCI	Copybook
C	DFH$BRCC DFH$BRLC	DFH$BREC	DFH$BRXC	DFH$BRSH
COBOL	DFH0CBRC DFH0CBRL		DFH0CBRX	DFH0CBRA

DFH$BRCC

This is the C language host business client sample, driven by transaction BRCH. DFH$BRCC formats a COMMAREA using the structures defined in the DFH$BRSH header file, to contain the business data that will be passed to the NACT transaction. DFH$BRCC then LINKS to DFH$BRLC (the C bridge client) , passing the COMMAREA, to perform the following functions:

• Run the NACT search function to obtain an account number from a name.

- Run the NACT display function to obtain account details for the account number.

When DFH$BRLC returns with the requested data, DFH$BRCC writes it to TS queue BRCH.

DFH0CBRC

This is the COBOL language host business client sample, driven by transaction BRCO. DFH0CBRC formats a COMMAREA using the structures defined in the DFH0CBRA copybook, to contain the business data that will be passed to the NACT transaction. DFH0CBRC then LINKS to DFH0CBRL, passing the COMMAREA, to perform the following functions:

- Run the NACT search function to obtain an account number from a name.
- Run the NACT display function to obtain account details for the account number.

When DFH0CBRL returns with the requested data, DFH0CBRC writes it to TS queue BRCO.

DFH$BRLC/DFH0CBRL

DFH$BRLC and DFH0CBRL LINK to DFHL3270 to drive the NACT transaction, using Link3270 in session mode. They do the following:

- Allocate the bridge facility
- Format the vectors to send in the Link3270 message
- Call the NACT transaction using the Link3270 bridge
- Return the requested data in the COMMAREA
- Delete the bridge facility

DFH$BRXC

This is the C language EXCI business client sample. DFH$BRXC formats a COMMAREA using the structures defined in the DFH$BRSH header file, to contain the business data that will be passed to the NACT transaction. DFH$BRXC then LINKS to DFH$BRLC, using the EXCI interface, passing the COMMAREA, to perform the following functions:

- Run the NACT search function to obtain an account number from a name.
- Run the NACT display function to obtain account details for the account number.

When DFH$BRLC returns with the requested data, DFH$BRXC writes it to SYSPRINT.

DFH0CBRX

This is the COBOL language EXCI business client sample. DFH0CBRX formats a COMMAREA using the structures defined in the DFH0CBRA copy book, to contain the business data that will be passed to the NACT transaction. DFH0CBRX then LINKS to DFH0CBRL, using the EXCI interface, passing the COMMAREA, to perform the following functions:

- Run the NACT search function to obtain an account number from a name.
- Run the NACT display function to obtain account details for the account number.

When DFH0CBRL returns with the requested data, DFH0CBRX writes it to SYSPRINT.

DFH$BREC

- Allocates the bridge facility

- Sends an ECI request to program DFHL3270 to run NACT and obtain the menu screen
- Formats a message with BRIV vectors to run the NACT search function to obtain an account number from a name.
- Sends an ECI request to program DFHL3270 to run the NACT search function.
- Extracts the search output data from the outbound message.
- Sends an ECI request to program DFHL3270 to run NACT and obtain the menu screen
- Formats a message with BRIV vectors to run the NACT display function to obtain account details for the account number.
- Extracts the account details from the outbound message.
- Formats a client response using the structures defined in the DFH$BRSH header file and displays it to the client end-user.
- Deletes the bridge facility

The NACT transaction

The NACT sample transaction demonstrates the design and development of CICS applications. It is a COBOL pseudoconversational 3270-based CICS application that operates on the customer account file of a fictitious company, KanDoIT.

The NACT sample transaction is taken from the book *Designing and Programming CICS Applications* , published by O'Reilly & Associates, Inc; ISBN 1–56592–676–5.

The NACT application provides the following services:
- access to an account record by account number
- addition of a new account number and account record
- modification of an account record
- deletion of an account record
- access to an account record by customer name

The logic of the application is divided into the following pseudoconversational steps:
1. A menu is displayed to allow selection of the service required.
2. The updated menu screen is read; the requested record is obtained and displayed
3. If the record is modified, the changes are received and the file updated; the menu is re-displayed.

The Link3270 sample client programs request the record number for customer name JACOB JONES; retrieve the record and display, or store, the retrieved record.

Running the sample client programs

You must perform the following setup to run the samples.

About this task

Setup the Link3270 environment
About this task

Set up the Link3270 environment as described in Chapter 4, "Managing the Link3270 bridge environment," on page 47. Ensure that you have defined:

- Link3270 system initialization parameters
- The DFHBRNSF file

Setup for the CICS-based clients

Add the load library containing the load modules to the RPL concatenation of your CICS startup job.

Procedure

1. Install and setup the NACT transaction, as described in "Setup for the NACT transaction" on page 108

2. Translate, compile and link the COBOL or C language programs DFH0CBRC, DFH0CBRL, DFH$BRCC, and DFH$BRLC, using a Language Environment® conforming compiler, ensuring the library containing the DFH$BRSH and DFH0CBRA copybooks is accessible. See Device dependent support: DDS , in the *CICS Application Programming Guide* for guidance on translating and compiling CICS programs. Translate, compile and link the COBOL or C language programs DFH0CBRC, DFH0CBRL, DFH$BRCC, and DFH$BRLC, using a Language Environment conforming compiler, ensuring the library containing the DFH$BRSH and DFH0CBRA copybooks is accessible. See Device dependent support: DDS , in the *CICS Application Programming Guide* for guidance on translating and compiling CICS programs.

3. Install resource definitions for the following CICS resources:

Table 13. Resource definitions for sample clients

Resource	Description
DFH0CBRC	COBOL sample business client
DFH0CBRL	COBOL sample bridge client
DFH$BRCC	C sample business client
DFH$BRLC	C sample bridge client
BRCO	Transaction to drive DFHC0BRC
BRCH	Transaction to drive DFH$BRCC

Examples of these resource definitions are provided for you in group DFH$BRLK. Install this group, or add it to the grouplist installed during CICS startup.

4. Run the samples by entering the transaction name BRCO (for the COBOL sample) or BRCH (for the C sample) at a CICS terminal.

Setup for z/OS based client

You must perform these tasks to set up the z/OS based client.

About this task

1. Install and setup the NACT transaction in CICS, as described in "Setup for the NACT transaction" on page 108.

2. Edit the DFH0CBRX or DFH$BRXC sample to pass the netname of the CICS region where the bridge client program (DFH$BRLC or DFH0CBRL) is installed. Compile and link the COBOL or C language programs DFH0CBRX or DFH$BRXC, using a Language Environment conforming compile, ensuring the library containing the DFH$BRSH and DFH0CBRA copybooks is accessible. Ensure also that the CICS supplied SDFHEXCI data set is concatenated to SYSLIB for your compile step. Place the output load modules in an appropriate z/OS library.

3. Translate, compile and link the COBOL or C language programs DFH0CBRL, and DFH$BRLC, using a Language Environment conforming compiler, ensuring the library containing the DFH$BRSH and DFH0CBRA copybooks is accessible. See Device dependent support: DDS , in the *CICS Application Programming Guide* for guidance on translating and compiling CICS programs.

   Add the load library containing the load modules to the RPL concatenation of your CICS startup job.

4. Create and install a CONNECTION resource definition to define the interface between CICS and z/OS that will be used by the EXCI request. See Chapter 13, "Defining connections to CICS," on page 161, and Chapter 8, "Introduction to the external CICS interface," on page 113 for an introduction and guidance on how to use of the EXCI interface.

5. To run the EXCI samples, you can use the following JCL supplied in file DFH$BRXJ in SDFHINST. You should edit this JCL to replace *hlq* with your own prefix, assigned during CICS installation, and to replace *application library* with the name of the library that contains the load modules created in step 1. Run this job in the z/OS batch environment.

```
     ----------------------------------------------------------------------
//DFH$BRXJ JOB (accounting information)
//         CLASS=A,MSGCLASS=A,MSGLEVEL=(1,1),REGION=12M
//LINK3270 EXEC PGM=DFH0CBRX
//*********************************************************************
//*                                                                  *
//* JCL NAME = DFH$BRXJ                                              *
//*                                                                  *
//* DESCRIPTIVE NAME = Link3270 bridge EXCI business client sample   *
//*                                                                  *
//* FUNCTION =                                                       *
//*                                                                  *
//*     Sample JCL for running the Link3270 bridge EXCI              *
//*     business client samples DFH0CBRX and DFH$BRXC.               *
//*                                                                  *
//*     The file DSN qualifier hlq must be changed.                  *
//*     This JCL runs the COBOL sample DFH0CBRX                      *
//*     This must be compiled into application library               *
//*     before the JCL is run.                                        *
//*     application library must be changed.                         *
//*     To run the C Sample change DFH0CBRX to DFH$BRXC.             *
//*                                                                  *
//*     The CICS External Interface Guide contains a detailed        *
//*     description of the Link3270 bridge.                          *
//*                                                                  *
//*********************************************************************
//STEPLIB  DD DSN=application library,DISP=SHR
//         DD DSN= hlq.SDFHEXCI,DISP=SHR
//SYSPRINT DD SYSOUT=*
//SYSABEND DD SYSOUT=*
//
```

Figure 14. EXCI sample JCL

Note: The output from the NACT sample is returned in the output from this job.

Setup for the workstation client

You must perform the following steps to set up the workstation client.

Procedure

1. Install the CICS Transaction Gateway on your workstation, as described in the relevant CICS Transaction Gateway administration manual for your workstation platform, ensuring that the developer kit is included.
2. Download the following programs and header files:

Table 14. Required files

File	Source library
dfh$brec.c	CICSTS42.CICS.SDFHSAMP
dfh$brxc.h	CICSTS42.CICS.SDFHSAMP
dfh$brmh.h	CICSTS42.CICS.SDFHSAMP
dfhbrich.h	CICSTS42.CICS.SDFHC370
dfhbrihh.h	CICSTS42.CICS.SDFHC370

Compile dfh$brec, with these header files, and DFHAID.h in your path.

3. Setup the ECI connection in CICS as described in the *CICS Family: Communicating from CICS on zSeries* manual.
4. Setup the ECI connection in your workstation as described in the appropriate Client Administration manual.
5. Install and setup the NACT transaction in CICS, as described in "Setup for the NACT transaction"

Setup for the NACT transaction

You can set up the NACT sample to use with the bridge sample clients

About this task

The following table shows the components that form the NACT sample application, supplied during CICS installation.

Table 15. NACT sample components

File	Type	Library
DFH0CNA1	COBOL source	SDFHSAMP
DFH0CNA2	COBOL source	SDFHSAMP
DFH0CNA3	COBOL source	SDFHSAMP
DFH0CNA4	COBOL source	SDFHSAMP
DFH0CNA5	COBOL source	SDFHSAMP
DFH0MNA	Mapset	SDFHSAMP
DFH0CNAA	Copybook	SDFHSAMP
DFH0CNAB	Copybook	SDFHSAMP
DFH0CNAC	Copybook	SDFHSAMP
DFH0CNAE	Copybook	SDFHSAMP

Table 15. NACT sample components (continued)

File	Type	Library
DFH0CNAF	Copybook	SDFHSAMP
DFH0CNAG	Copybook	SDFHSAMP
DFH0CNAL	Copybook	SDFHSAMP
DFH0CNAM	Copybook	SDFHSAMP
DFH0CNAR	Copybook	SDFHSAMP
DFH0CNAU	Copybook	SDFHSAMP
DFH0CNAW	Copybook	SDFHSAMP
DFH$NACT	RDO group	CSD
DFHNADEF	JCL	XDFHINST

To set up the NACT sample to use with the bridge sample clients, use the following procedure:

1. Assemble and link the map DFH0MNA. The map copybook can be created in this step, but the map copybook DFH0CNAM is supplied. See Installing map sets and partition sets in the *CICS Application Programming Guide* for guidance on assembling CICS maps.

   Add the load library containing the load modules to the RPL concatenation of your CICS startup job.

2. Translate, compile and link the COBOL programs DFH0CNA1–5, ensuring that the copybooks listed inTable 15 on page 108 are accessible. See Device dependent support: DDS , in the *CICS Application Programming Guide* for guidance on translating and compiling CICS programs.

   Add the load library containing the load modules to the RPL concatenation of your CICS startup job.

3. Create NACT files. Edit the JCL provided in file DFHNADEF to conform to your own installation naming conventions, and run it to create the following NACT files:

   xxx.ACCTFILE
   > The account file

   xxx.ACCTNAIX
   > The names alternate index

   xxx.ACTINUSE
   > record locking file

4. Edit the resource definitions in sample group DFH$NACT to conform to the naming conventions you used in step 3. Install this group, or add it to the grouplist installed during CICS startup.

5. You can now test that installation is complete by entering the transaction NACT on a CICS terminal.

Part 3. External CICS Interface

The external CICS interface is an application programming interface that enables a non-CICS program (a client program) running in MVS to call a program (a server program) running in a CICS region and to pass and receive data by means of a communications area.

This part contains:

Chapter 8. Introduction to the external CICS interface

The external CICS interface is an application programming interface that enables a non-CICS program (a client program) running in MVS to call a program (a server program) running in a CICS region and to pass and receive data by using a communications area.

The CICS application program is invoked as if linked to by another CICS application program.

This programming interface allows a user to allocate and open sessions, or *pipes* (a one-way communication path between a sending process and a receiving process) to a CICS region, and to pass distributed program link (DPL) requests over them. The multiregion operation (MRO) facility of CICS interregion communication (IRC) facility supports these requests, and each pipe maps onto one MRO session, where the client program represents the sending process and the CICS server region represents the receiving process. There is a default limit of 100 pipes per EXCI address space; the limit can be changed when MVS is IPLed.

This limit is implemented to avoid EXCI clients monopolizing MRO resources, which could prevent CICS regions from using MRO. The limit is applied in both MRO and cross system MRO (XCF/MRO) environments. An ALLOCATE_PIPE request results in an MRO LOGON request being issued and there is a limit on the total number of MRO LOGON requests allowed from all address spaces. This is critical when using XCF/MRO, where the limit on the number of members in an XCF group also limits the total number of MRO LOGONs.

The external CICS interface identifies the CICS region to communicate with by the CICS region APPLID, as defined in the **APPLID** system initialization parameter. You can specify the APPLID either on an EXCI API call or by using the DFHXCURM user-replaceable program. For more information about DFHXCURM, see Chapter 14, "The EXCI user-replaceable module," on page 167.

Note: Do not confuse the term *generic applid* with *generic resource name*. Generic resource names apply only to z/OS Communications Server generic resource groups, which are not supported by EXCI.

The client program and the CICS server region (the region where the server program runs or is defined) must be in the same MVS image unless:
- The CICS region is running in a sysplex that supports cross-system MRO.
- All DPL requests issued by the client program specify the SYNCONRETURN option.

Alternatively, if there is no local CICS region in the MVS image, you must specify the SVC parameter that the external CICS interface is to use, by coding a CICSSVC parameter in the DFHXCOPT table. Although the external CICS interface does not support the cross-memory access method, it can use the XCF access method provided by the CICS XCF/MRO facility. See Chapter 15, "Using the EXCI options table, DFHXCOPT," on page 169 for information about XCF/MRO.

A client program that uses the external CICS interface can operate multiple sessions for different users (either under the same or separate TCBs) all coexisting in the same MVS address space without knowledge of, or interference from, each other.

Where a client program attaches another client program, the attached program runs under its own TCB.

This chapter describes:
- "The EXCI programming interfaces"
- "Resource recovery" on page 118
- "EXCI concepts" on page 12
- "Requirements for the external CICS interface" on page 122

The EXCI programming interfaces

The external CICS interface provides two forms of programming interface: the EXCI CALL interface and the EXEC CICS interface.

The EXCI CALL interface: This interface consists of six commands that allow you to:
- Allocate and open sessions to a CICS system from non-CICS programs running under MVS
- Issue DPL requests on these sessions from the non-CICS programs
- Close and deallocate the sessions on completion of the DPL requests.

The six EXCI commands are:
- Initialize-User
- Allocate_Pipe
- Open_Pipe
- DPL call
- Close_Pipe
- Deallocate_Pipe

The EXEC CICS interface: The external CICS interface provides a single, composite command `EXEC CICS LINK PROGRAM` that performs all six commands of the EXCI CALL interface in one invocation.

This command is similar but not identical to the distributed program link command of the CICS command-level application programming interface.

API restrictions for server programs
> A CICS server program invoked by an external CICS interface request is restricted to the DPL subset of the CICS application programming interface. This subset (the DPL subset) of the API commands is the same as for a CICS-to-CICS server program.
>
> See Distributed Program Link (DPL) in CICS Application Programming for details of the DPL subset for server programs.

Choosing between the EXEC CICS and the CALL interface

You can use both the CALL interface (all six commands) and the EXEC CICS LINK command in the same program, to perform separate requests. As a general rule, it is unlikely that you would want to do this in a production program.

"Using EXCI sample application programs" on page 178 illustrates the various language versions of the CICS-supplied sample client program.

Each form of the external CICS interface has its particular benefits.

- For low-frequency or single DPL requests, you are recommended to use the EXEC CICS LINK command.

  It is easier to code, and therefore less prone to programming errors.

  Note that each invocation of an EXEC CICS LINK command causes the external CICS interface to perform all the functions of the CALL interface, which results in unnecessary overhead.

  Note also that this overhead is greatly increased if you use the EXEC CICS LINK command to communicate with a CICS server region in a different LPAR. In this case each invocation of the EXEC CICS LINK command generates a great deal of XCF activity because of the IRP logon, connect, disconnect and logoff which is required. You might find that you experience severe degradation of elapsed time between EXEC CICS LINK commands issued to a CICS server region in a separate LPAR, compared to the elapsed time of the same commands issued to a CICS server region in the same LPAR.

- For multiple or frequent DPL requests from the same client program, you are recommended to use the EXCI CALL interface.

  This is more efficient, because you need only perform the Initialize_User and Allocate_Pipe commands once, at or near the beginning of your program, and the Deallocate_Pipe once on completion of all DPL activity. In between these functions, you can open and close the pipe as necessary, and while the pipe is opened, you can issue as many DPL calls as you want.

Illustrations of the external CICS CALL interface

These four diagrams illustrate the external CICS interface using the EXCI CALL interface.

```
MVS Client Application              CICS Server Region

┌─────────────────────┐         ┌─────────────────────────┐
│                     │         │                         │
│                     │         │                         │
│                     │    ◄─────  MRO EXCI CONNECTION    │
│ External CICS user  │    ◄─────  installed with 3       │
│ environment         │    ◄─────  sessions               │
│ established         │         │  (PROTOCOL=EXCI)         │
│                     │         │  (RECEIVECOUNT=3)        │
│                     │         │                         │
│                     │         │                         │
│                     │         │                         │
│                     │         │                         │
└─────────────────────┘         └─────────────────────────┘
```
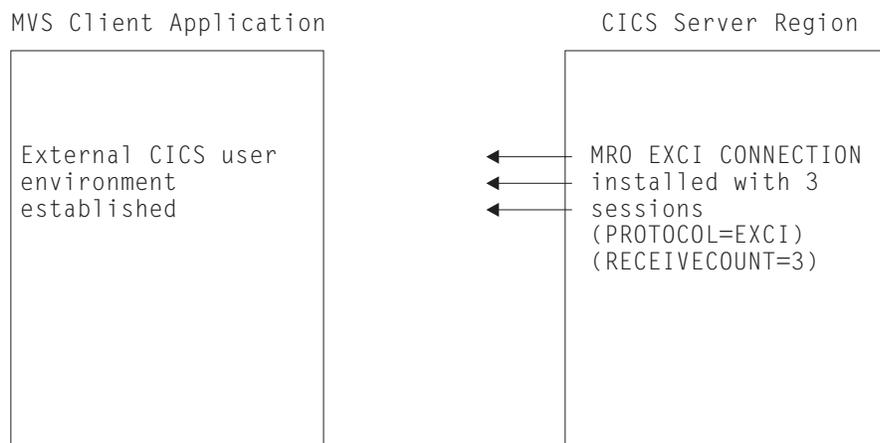
Figure 15. Stage 1: Status after an INITIALIZE_USER call

Note:

1. In Figure 15, the target CICS region is running with IRC open, and one EXCI connection with three sessions installed, at the time the client application program issues an INITIALIZE_USER call.

2. The client application program address space is initialized with the EXCI user environment. There is no MRO activity at this stage, and no pipe exists.

```
MVS Client Application                    CICS Server Region

┌─────────────────────────┐          ┌─────────────────────────┐
│                         │          │                         │
│                         │          │                         │
│  Pipe allocated      ───┼──────▶ ◀─┼── MRO EXCI CONNECTION    │
│                         │       ◀──┼── installed with 3       │
│                         │       ◀──┼── sessions               │
│                         │          │   (PROTOCOL=EXCI)        │
│                         │          │   (RECEIVECOUNT=3)       │
│                         │          │                         │
│                         │          │                         │
│                         │          │                         │
└─────────────────────────┘          └─────────────────────────┘
```
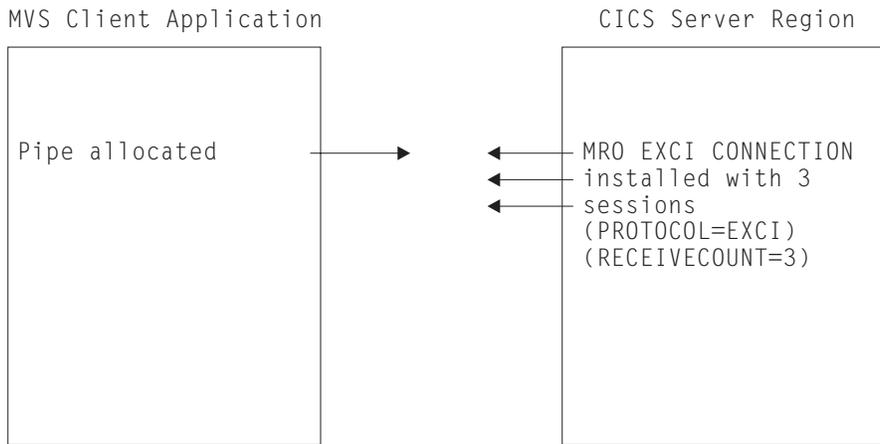
Figure 16. Stage 2: Status after the first ALLOCATE_PIPE call

Note: In Figure 16, the external CICS interface logs on to MRO, identifying the target CICS server region.

```
MVS Client Application                    CICS Server Region

┌─────────────────────────┐          ┌─────────────────────────┐
│                         │          │                         │
│                         │          │                         │
│  Pipe opened         ◀──┼──────────┼─▶ MRO EXCI CONNECTION    │
│                         │       ◀──┼── installed with 3       │
│                         │       ◀──┼── sessions               │
│                         │          │   (PROTOCOL=EXCI)        │
│                         │          │   (RECEIVECOUNT=3)       │
│                         │          │                         │
│                         │          │                         │
│                         │          │                         │
└─────────────────────────┘          └─────────────────────────┘
```
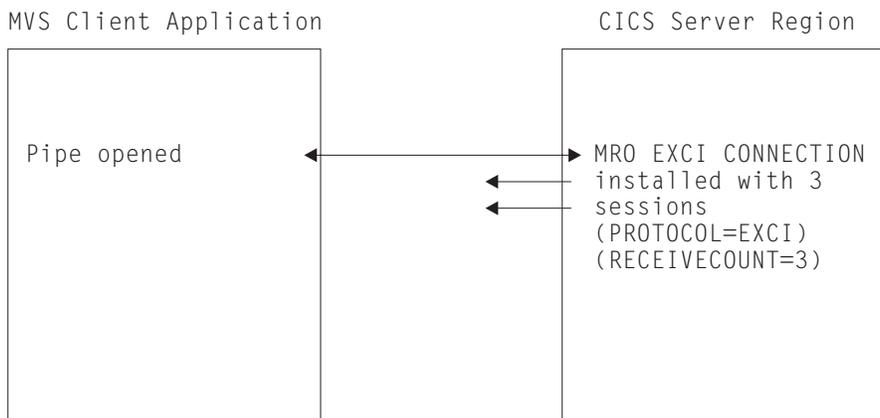
Figure 17. Stage 3: Status after the OPEN_PIPE call

Note:

1. In Figure 17, the external CICS interface connects to the CICS server region, and the pipe is now available for use.
2. The remaining two EXCI sessions are free, and can be used by further open pipe requests from the same, or a different, client application program (provided the connection is generic).

```
MVS Client Application                          CICS Server Region

                            DPL Request and data

     Pipe opened ◀                                ▶ MRO EXCI CONNECTION

                             Response and data ◀── installed with 3
                                               ◀── sessions
                                                   (PROTOCOL=EXCI)
                                                   (RECEIVECOUNT=3)
```
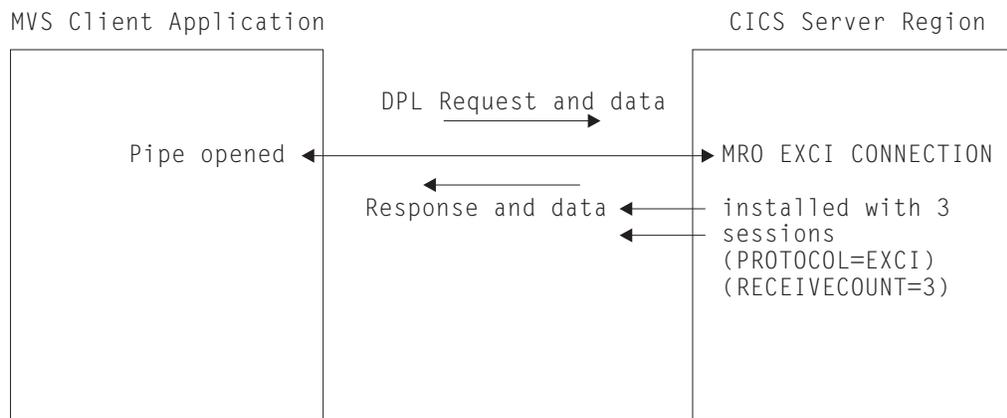
Figure 18. Stage 4: Status with one open pipe, processing a DPL call

Note: In Figure 18, the external CICS interface passes the DPL request over the open pipe, with any associated data. The CICS server region returns a response and data over the open pipe.

Closing pipes: When the client application program closes a pipe, it remains allocated ready for use by the same user, and the status is as shown in Figure 16 on page 116. At this stage, the MRO session is available for use by another open pipe request, from the same or from a different client application program (provided the connection is generic).

Deallocating pipes: When the client application program deallocates a pipe, it logs off from MRO and frees all the storage associated with the session. This leaves the status as shown in Figure 15 on page 115.

Illustration of the EXCI EXEC CICS interface

This diagram illustrates the EXEC CICS interface, and how it resolves to the six EXCI CALLs.

```
                MVS Client Application

  ┌─────────────────────────────┐
  │                             │
  │ EXEC CICS LINK command──┐   │
  │                         │   │
  │                         │   │
  ├─────────────────────────▼───┤
  │ The EXEC Interface Stub     │
  │      (DFHXCSTB)             │
  │                             │
  │ The stub calls the EXCI     │   ┌────────────────────────────┐
  │ EXEC interface program.─────┼───│ EXEC interface program     │
  └─────────────────────────────┘   │ issues following calls:    │
                                     │                            │
                                     │ INITIALIZE_USER            │
                                     │                            │
                                     │ ALLOCATE_PIPE  ─────────────▶  pipe to CICS server
                                     │                            │     region is allocated
                                     │ OPEN_PIPE      ◀──────────>    pipe opened
                                     │                            │
                                     │ DPL           ◀──────────>    Request and data sent and
                                     │                            │     response and data received
                                     │ CLOSE_PIPE    ─────────────▶  pipe closed
                                     │                            │
                                     │ DEALLOCATE_PIPE            │
                                     └────────────────────────────┘
```

Figure 19. Illustration of the external CICS interface using the EXEC CICS command

Resource recovery

Resource recovery consists of the protocols and program interfaces that allow an application program to make consistent changes to multiple protected resources. The external CICS interface supports resource recovery.

A CICS server program that is invoked by an external CICS interface request can update recoverable resources; the changes are committed when the mirror transaction in the CICS server region takes a sync point. The client program can determine when sync pointing should occur. There are two options:

* Resource recovery controlled by the CICS server regions. In this case, changes to recoverable resources are committed at the completion of each DPL request, independently of the client program. Also, in addition to the sync point taken when the server program returns control to CICS (the SYNCONRETURN option), the server program can take explicit sync points during execution.
* Resource recovery controlled by the EXCI client program with the support of recoverable resource management services (RRMS). When the client program requests it, updates made by the server program in successive DPL requests are committed together.

  To support this option, CICS and the external CICS interface both use resource recovery services (RRS), the z/OS sync point manager[1], which is an MVS component of recoverable resource management services (RRMS). In the context of RRMS, CICS is a **resource manager**; the client program can issue requests to

---

1. RRMS comprises three z/OS components: registration services, context services, and resource recovery services (RRS)

other resource managers, and have resources owned by those resource managers committed in the same **unit-of-recovery (UR).**[2]

These options are controlled as follows:

- By the DPL_opts parameter of the DPL_request.
- By the SYNCONRETURN option, either specified or omitted, on the EXEC CICS LINK PROGRAM command.

If you specify SYNCONRETURN, a sync point is taken on completion of each DPL request. If SYNCONRETURN is omitted, a sync point is taken when the client program requests it using the interfaces described in "Use of sync points in the client program" on page 122.

Use of RRMS with the external CICS interface

You can use z/OS recoverable resource management services (RRMS) to coordinate distributed program link (DPL) requests.

To use RRMS to coordinate DPL requests, ensure that the following conditions are met:

- The EXCI client and the CICS region to which it sends DPL requests run in the same MVS image. This is an RRMS restriction, and does not apply to DPL requests that specify SYNCONRETURN.
- The CICS region that receives the DPL requests is started with `RRMS=YES` specified as a system initialization parameter (the default is `RRMS=NO`).
- Resource recovery services (RRS) run in the MVS image where CICS and the client program execute. See *z/OS MVS Programming: Resource Recovery*.

The following figure shows how the external CICS interface and CICS use RRMS. It shows the flow between the MVS batch region that contains the external CICS interface and the EXCI client program, and the CICS server region that contains the CICS mirror and a CICS application program. The numbers on the diagram refer to the principal steps in a unit of recovery (UR), as listed after the figure.

---

2. A unit of recovery is analogous to a CICS unit of work
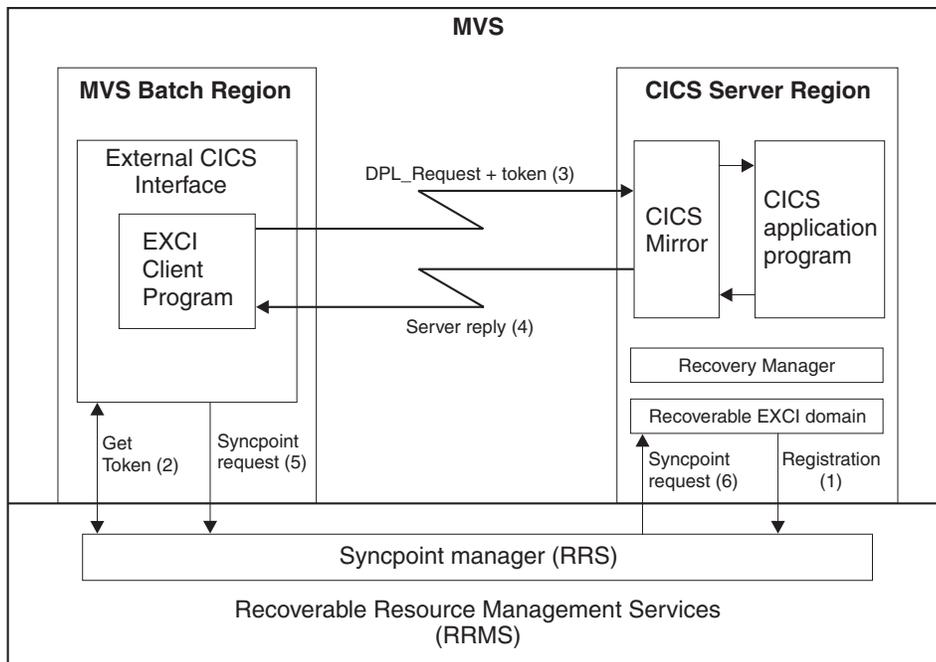
Figure 20. Conceptual view of EXCI client and CICS server region using RRMS

1. If the CICS system initialization parameter **RRMS=YES** is specified, CICS registers with RRMS as a resource manager. This registration occurs during CICS initialization.

2. When the EXCI client program issues a DPL_Request call in 2-phase commit mode (a call that omits the SYNCONRETURN option), it receives the following from RRMS:

   • A unit-of-recovery identifier (URID)

   • A context token

   • A pass token

3. The URID and the tokens obtained by EXCI on behalf of the client program are included on the DPL request that is passed to the CICS server region. If the DPL request is the first one in the UR, CICS calls RRS to express an interest in the UR, attaches a new mirror transaction, and validates the tokens. If the request is valid, the mirror program links to the specified server application program. The server program completes its work, which is all performed in the UR. This work can include updating recoverable resources in the local server region, or daisy chaining to other CICS regions.

4. When the server program completes, it returns the communications area (COMMAREA) and return codes to the client program.

   Note: Steps 3 and 4 can repeated many times for the same UR.

5. When the EXCI client program is ready to commit or back out its changes, the program invokes RRS to begin the 2-phase commit protocol.

6. RRS acts as coordinator and completes one of the following actions:

   • RRS asks the resource managers to prepare to commit all updates in the UR. Resource managers other than the CICS server region might also express an interest in the UR. If all vote yes, RRS tells them to go ahead and commit the changes. If any vote no, RRS tells all the resource managers to back out all the changes made in the UR.

- RRS tells all the resource managers that express an interest in the UR to back out all the changes made in the UR.

The UR is now complete and CICS detaches the mirror task. If the EXCI client sends any new DPL requests after this point, EXCI starts a new UR and CICS attaches a new mirror transaction.

Each DPL request that specifies the SYNCONRETURN option attaches a new mirror task in the target CICS region. The first DPL request that does not specify SYNCONRETURN also attaches a new mirror task, but subsequent requests are directed to the same mirror task. When a sync point takes place, the mirror task ends, and the next non-SYNCONRETURN request attaches a new mirror. See Figure 21. In this figure, an MVS client application issues DPL requests with and without SYNCONRETURN. The numbers on the figure refer to the principal flow, as listed after the figure.
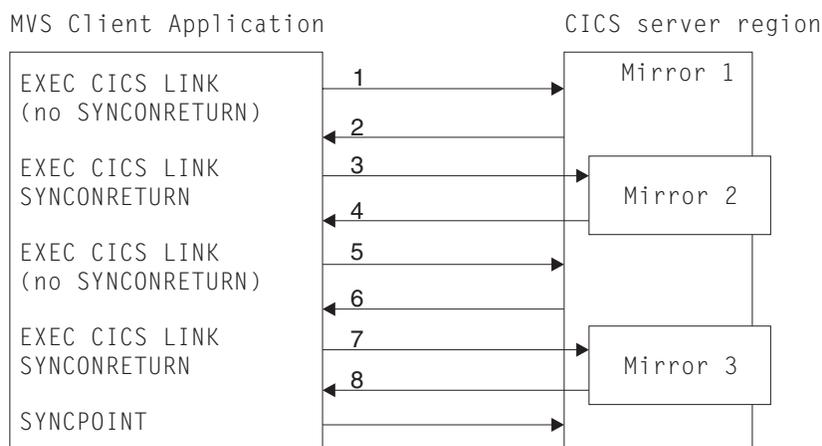
```
MVS Client Application                    CICS server region
                                              Mirror 1
┌──────────────────────────┐      1   ┌──────────────────────┐
│ EXEC CICS LINK           │─────────▶│                      │
│ (no SYNCONRETURN)        │      2   │                      │
│                          │◀─────────│                      │
│                          │      3   │   ┌──────────────────┤
│ EXEC CICS LINK           │─────────▶│   │                  │
│ SYNCONRETURN             │      4   │   │   Mirror 2       │
│                          │◀─────────│   │                  │
│ EXEC CICS LINK           │      5   │   └──────────────────┤
│ (no SYNCONRETURN)        │─────────▶│                      │
│                          │      6   │                      │
│                          │◀─────────│                      │
│ EXEC CICS LINK           │      7   │   ┌──────────────────┤
│ SYNCONRETURN             │─────────▶│   │   Mirror 3       │
│                          │      8   │   │                  │
│                          │◀─────────│   └──────────────────┤
│ SYNCPOINT                │          │                      │
└──────────────────────────┘          └──────────────────────┘
```

Figure 21. Effect of mixing DPL requests with and without the SYNCONRETURN option

1. Client issues a DPL request without the SYNCONRETURN option.

   Because no mirror transaction is running, a new mirror (Mirror 1) is attached.

2. The DPL request completes, and because it was issued without the SYNCONRETURN option, the mirror transaction waits for another request.

3. Client issues DPL request with the SYNCONRETURN option.

   A new mirror transaction (Mirror 2) is attached.

4. On completion of the DPL request, resources updated by the mirror transaction are committed, and the mirror transaction ends.

5. Client issues another DPL request without the SYNCONRETURN option. Mirror 1 receives and executes the DPL request.

6. The DPL request completes, and once again, the mirror transaction waits for another request.

7. Client issues DPL request with the SYNCONRETURN option.

   A new mirror transaction (Mirror 3) is attached.

8. On completion of the DPL request, resources updated by the mirror transaction are committed, and the mirror transaction ends.

9. The client program requests a syncpoint. Resources updated by mirror 1 are committed, and the transaction ends.

Use of sync points in the client program

A client program can request that a sync point is taken by using an MVS callable service to commit or back out changes.

To commit changes instigated by the client program, use one of the following MVS callable services:

Application_Commit_UR (SRRCMIT)
> For a description of Application_Commit_UR, see *z/OS MVS Programming: Callable Services for High-Level Languages*.

Commit_UR (ATRCMIT)
> For a description of Commit_UR, see *z/OS MVS Programming: Resource Recovery*.

To back out changes in the client program, use one of the following MVS callable services:

Application_Backout_UR (SRRBACK)
> For a description of Application_Backout_UR, see *z/OS MVS Programming: Callable Services for High-Level Languages*.

Backout_UR (ATRBACK)
> For a description of Backout_UR, see *z/OS MVS Programming: Resource Recovery*.

If none of these interfaces are used, changes are committed or backed out explicitly when the client program either ends normally or abends. It is not advisable to use implicit commit or backout for the following reasons:

- The client program cannot tell whether updates were committed or backed out. Even if the program ends normally, a resource manager might back out any changes.
- The runtime environment for high level languages might intercept errors that would otherwise result in an operating system abend. If such an error is intercepted and the client program does not take any explicit action, the program might terminate normally and updates might be committed. Code your client program to ensure that resources are committed or backed out correctly in these situations. For example, a PL/I program can include an ON unit that issues an **SRRBACK** command when errors are encountered. Similarly, a COBOL program can use the ON phrase on statements that might encounter errors.

Requirements for the external CICS interface

Client programs running in an MVS address space can communicate only with CICS server regions running under Version 4 of CICS for MVS/ESA, or a later release.

Also, the client program can connect to the server CICS region only through the Version 4 of CICS for MVS/ESA (or later) level of the interregion communication program, DFHIRP.

For information about DFHIRP, and its requirement to be installed in the MVS extended link pack area (ELPA), see Installing MRO support, in the *CICS Transaction Server for z/OS Installation Guide*.

When you use RRMS to coordinate DPL requests, the server CICS region must be running CICS TS Release 3 or later.

Chapter 9. The EXCI CALL interface

The EXCI CALL interface consists of six commands.

The commands allow you to:
- Allocate and open sessions to a CICS system from non-CICS programs running under MVS
- Issue distributed program link (DPL) requests on these sessions from the non-CICS programs
- Close and deallocate sessions on completion of DPL requests.

The six EXCI commands are:
- Initialize_User
- Allocate_Pipe
- Open_Pipe
- DPL_Request
- Close_Pipe
- Deallocate_Pipe

The application program stub, DFHXCSTB: The EXCI commands invoke the external CICS interface via an application programming stub provided by CICS, called DFHXCSTB. You must include this stub when you link-edit your non-CICS program.

The EXCI CALL interface commands

In the description of each command that follows, the syntax box illustrates the assembler form of the command. The syntax shows VL,MF=(E,(1)) for each command, indicating the execute form of the CALL macro, with the parameter list storage area addressed by Register 1.

The commands are also supported by C, COBOL, and PL/I programming languages, using the CALL conventions appropriate for these languages.

There are examples of these CALLs, in all the supported languages, in the sample client programs provided by CICS. See "Using EXCI sample application programs" on page 178 for information about these.

Initialize_User

Initialize the user environment. This includes obtaining authority to use IRC facilities. The environment is created for the lifetime of the TCB, so the command needs to be issued only once per user per TCB. Further commands from this user must be issued under the same TCB.

Syntax

```
CALL DFHXCIS,(version_number,return_area,user_token,call_type,
             user_name),VL,MF=(E,(1))
```

Parameters

version_number
> A fullword binary input area indicating the version of the external CICS interface parameter list being used. It must be set to 1 in the client program.
>
> The equated value for this parameter in the CICS-supplied copybook DFHXCPL*x* (where *x* indicates the language) is VERSION_1. See "Return area and function call EQUATE copybooks" on page 144 for copybook details.

return_area
> A 5-word output area to receive response and reason codes, and a message pointer field. For more details see "Return area for the EXCI CALL interface" on page 144.

user_token
> A 1-word output area containing a 32-bit token supplied by the CICS external interface to represent the client program.
>
> The user token corresponds to the *user-name* parameter. The client program must pass this token on all subsequent external CICS interface commands made for the user defined on the *user_name* parameter.

call_type
> A 1-word input area indicating the function of the command. It must be set to 1 in the client program to indicate that this is an Initialize_User command.
>
> The equated value for this call in the CICS-supplied copybook DFHXCPL*x* (where *x* indicates the language) is INIT_USER. See "Return area and function call EQUATE copybooks" on page 144 for copybook details.

user_name
> An input area holding a name that identifies the user of the external CICS interface. Generally, this is the client program. If this user is to use a specific pipe, then the value in *user_name* must match that of the NETNAME attribute of the CONNECTION definition for the specific pipe.

Responses and reason codes

For all non-zero response codes, a unique reason code value identifies the reason for the response.

Note: All numeric response and reason code values are in decimal.

The following is a summary of the response and reason codes that the external CICS interface can return on the Initialize_User call:

Response OK
> Command executed successfully (RC 0). Reason code:
>
> **0**   Normal response

Response WARNING
> The command executed successfully, but with an error (RC 4). Reason codes:
>
> **3**   VERIFY_BLOCK_FM_ERROR
>
> > Initialize_User processing requires storage below 16MB to build the parameter list for the SSI Verify call, and an error has occurred during the FREEMAIN for this area.
>
> **4**   WS_FREEMAIN_ERROR

An attempt to FREEMAIN working storage has resulted in an MVS FREEMAIN error.

Response RETRYABLE

The command failed because of setup errors but can be reissued (RC 8). Reason code:

201 NO_CICS_IRC_STARTED

An Initialize_User command has been issued on an MVS image that has had no IRC activity since the previous IPL, and the external CICS interface cannot determine the CICS SVC number.

Response USER_ERROR

The command failed because of an error in either the client or the server (RC 12). Reason codes:

401 INVALID_CALL_TYPE

An invalid call-type parameter value is specified on this EXCI request.

402 INVALID_VERSION_NUMBER

The version_number parameter does not specify a value of 1 or 2.

403 INVALID_USER_NAME

The user_name parameter consists of all blank characters (X'40').

410 DFHMEBM_LOAD_FAILED

During Initialize_User processing, the external CICS interface attempted to load the main message module in preparation for issuing external CICS interface messages, and the load of this module failed.

411 DFHMET4E_LOAD_FAILED

The load of message module, DFHMET4E, has failed. During Initialize_User processing, the external CICS interface attempted to load its message table in preparation for issuing messages. The load of this module failed.

412 DFHXCURM_LOAD_FAILED

During Initialize_User processing, the external CICS interface attempted to load the user-replaceable module, DFHXCURM. The load of this module failed.

413 DFHXCTRA_LOAD_FAILED

During Initialize_User processing, the external CICS interface attempted to load the trap module (DFHXCTRA). The load of this module has failed.

419 CICS_AFCB_PRESENT

An Initialize_User request has been issued on a TCB that has already been used by CICS or by CICS batch shared database. The external CICS interface cannot share a TCB with CICS, ensuring that a CICS application program cannot issue EXCI requests.

420 DFHXCOPT_LOAD_FAILED

During Initialize_User processing, the external CICS interface attempted to load its options module, DFHXCOPT. The load of this module failed.

421   RUNNING_UNDER_AN_IRB

The EXCI call is issued under an MVS IRB, which is not permitted.

422   SERVER_ABENDED

While processing a DPL request, the CICS server application program abended without handling the error.

423   SURROGATE_CHECK_FAILED

A DPL request has been issued specifying a USERID parameter.

424   RRMS_NOT_SUPPORTED

A DPL request omitting the SYNCONRETURN option has been made on a system that is not running OS/390 Version 2 Release 5 or higher.

425   UOWID_NOT_ALLOWED

A DPL request omitted the SYNCONRETURN option, but specified a value of UOWID. This combination of parameters is not permitted on a DPL request.

426   INVALID_TRANSID2

A DPL request has been issued with a `TRANSID2` parameter that consists of all blanks.

427   INVALID_CCSID

A DPL request has been issued with a `CCSID` parameter that specifies an invalid value.

428   INVALID_ENDIAN

A DPL request has been issued with a endian parameter that specifies an invalid value.

429   DFHXCEIX_LOAD_FAILED

During Initialize_User processing, the external CICS interface attempted to load the module (DFHXCEIX). The load of this module has failed.

430   DFHXCPRX_LOAD_FAILED

During Initialize_User processing, the external CICS interface attempted to load the module (DFHXCPRX). The load of this module has failed.

Response SYSTEM_ERROR
The command failed (RC 16). Reason codes:

601   WS_GETMAIN_ERROR

During Initialize_User processing, a GETMAIN for working storage failed.

602   XCGLOBAL_GETMAIN_ERROR

During Initialize_User processing, a GETMAIN failed for a critical control block (XCGLOBAL).

603	XCUSER_GETMAIN_ERROR

During Initialize_User processing, a GETMAIN request failed for the user control block (XCUSER).

605	VERIFY_BLOCK_GM_ERROR

During Initialize_User processing, a GETMAIN failed for an EXCI internal control block.

606	SSI_VERIFY_FAILED

A VERIFY call to the MVS subsystem interface (SSI) to obtain the current CICS SVC number failed.

607	CICS_SVC_CALL_FAILURE

During Initialize_User processing, a call to the currently installed CICS SVC failed.

622	ESTAE_SETUP_FAILURE

In order to protect itself from possible program checks the external CICS interface establishes an MVS ESTAE. In this case, the MVS ESTAE macro has failed.

623	ESTAE_INVOKED

A program check is encountered during call processing, and the ESTAE is invoked.

627	INCORRECT_SVC_LEVEL

The release level of the CICS SVC (DFHCSVC) is not the same (or higher) than the release level of the external CICS interface.

For more information about response codes, see "EXCI call response code values" on page 143.

For information about the reason codes, see Chapter 19, "Response and reason codes returned on EXCI calls," on page 215.

Allocate_Pipe

Allocate a single session, or pipe, to a CICS region. This command does not connect the client program to a CICS region; this happens on the Open_Pipe command.

You can allocate up to 250 pipes in an EXCI address space. The default limit is 100 pipes. However, you can use the CICS system initialization parameter, LOGONLIM, to change the limit when MVS is IPLed. This limit is set to prevent EXCI clients from monopolizing MRO resources, which could prevent CICS systems from using MRO. The limit is applied in both MRO and cross system MRO (XCF/MRO) environments.

An ALLOCATE_PIPE request results in an MRO LOGON request being issued and there is a limit on the total number of MRO LOGON requests allowed from all address spaces. This is particularly critical when using XCF/MRO, where the limit on the number of members in a XCF group also limits the total number of MRO LOGONs

Syntax

```
CALL DFHXCIS,(version_number,return_area,user_token,call_type,
              pipe_token,CICS applid,allocate_opts),VL,MF=(E,(1))
```

Parameters

version_number

A fullword binary input area indicating the version of the external CICS interface parameter list being used. It must be set to 1 in the client program.

The equated value for this parameter in the CICS-supplied copybook DFHXCPL*x* (where *x* indicates the language) is VERSION_1. See Table 18 on page 145 for copybook details.

return_area

A 5-word output area to receive response and reason codes, and a message pointer field. For more details see "Return area for the EXCI CALL interface" on page 144.

user_token

The 1-word token returned on the Initialize_User command.

call_type

A 1-word input area indicating the function of the command. It must be set to 2 in the client program to indicate that this is an Allocate_Pipe command.

The equated value for this call in the CICS-supplied copybook DFHXCPL*x* (where *x* indicates the language) is ALLOCATE_PIPE. See Table 18 on page 145 for copybook details.

pipe_token

A 1-word output area. CICS returns a 32-bit token in this area to represent the allocated session. This token must be used on any subsequent command that uses this session.

CICS_applid (or null_ptr)

An 8-byte input area containing the generic applid of the CICS system to which the allocated session is to be connected.

Although an applid is required to complete the Allocate_Pipe function, this parameter is optional on the Allocate_Pipe call. You can either specify the applid on this parameter to the Allocate_Pipe call, or in the user-replaceable module, DFHXCURM, using the URMAPPL parameter (DFHXCURM is always invoked during Allocate_Pipe processing). You can also use the URMAPPL parameter in DFHXCURM to override an applid specified on the Allocate_Pipe call. See Chapter 14, "The EXCI user-replaceable module," on page 167 for information about the URMAPPL parameter.

If you omit the applid from the call, you must ensure that the CALL parameter list contains a null address for *CICS_applid*. How you do this depends on the language you are using for the non-CICS client program. For an example of a call that omits an optional parameter, see "Example of EXCI CALLs with null parameters" on page 146.

allocate_opts

A 1-byte input area to represent options specified on this command. The options specify which type of session is to be used—specific or generic. X'00' represents a specific session. X'80' represents a generic session.

The equated value for these options in the CICS-supplied copybook DFHXCPL*x* (where *x* indicates the language) are SPECIFIC_PIPE and GENERIC_PIPE. See Table 18 on page 145 for copybook details.

Responses and reason codes

For all non-zero response codes, a unique reason code value identifies the reason for the response.

Note: All numeric response and reason code values are in decimal.

The following is a summary of the response and reason codes that the external CICS interface can return on the Allocate_Pipe call:

Response OK
> Command executed successfully (RC 0). Reason code:
>
> **0**   Normal response

Response USER_ERROR
> The command failed because of an error in either the client or the server (RC 12). Reason codes:
>
> **401**   INVALID_CALL_TYPE
>
> **402**   INVALID_VERSION_NUMBER
>
> **404**   INVALID_USER_TOKEN
>
> **421**   RUNNING_UNDER_AN_IRB

Response SYSTEM_ERROR
> The command failed (RC 16). Reason codes:
>
> **604**   XCPIPE_GETMAIN_ERROR
>
> **608**   IRC_LOGON_FAILURE
>
> **622**   ESTAE_SETUP_FAILURE
>
> **623**   ESTAE_INVOKED
>
> **628**   IRP_LEVEL_CHECK_FAILURE

For information about response codes, see "EXCI call response code values" on page 143.

For information about the reason codes, see Chapter 19, "Response and reason codes returned on EXCI calls," on page 215.

Open_Pipe

Cause IRC to connect an allocated pipe to a receive session.

Open_Pipe causes IRC to connect an allocated pipe to a receive session of the appropriate connection defined in the CICS region named either on the Allocate_Pipe command, or in DFHXCURM. The appropriate connection is either:

- The EXCI connection with a NETNAME value equal to the *user_name* parameter on the Initialize_User command (that is, you are using a specific connection, dedicated to this client program)
- The EXCI connection defined as generic.

In an XCF environment, the Open_Pipe command causes the interregion communication program, DFHIRP, to connect to the LPAR that receives the request. This request is asynchronous, so although the Open_Pipe command can receive a good return code, the subsequent DPL_Request call might fail.

If you shut down CICS without the support of the supplied shutdown-assist transaction (CESD) or an equivalent, and sessions are left open, CICS might not be able to shut its IRC facility in an orderly manner. A normal shutdown of CICS without the support of the shutdown assist transaction waits if any EXCI sessions are not closed. CICS issues message DFHIR2321 indicating the following information:

- The netname of the session if it is on a specific connection
- The word GENERIC if the open sessions are on a generic connection.

If you use the supplied shutdown-assist transaction, CESD, sessions left open do not present a problem to normal shutdown, because CESD issues an immediate close of IRC. Provided that at least one DPL_Request call has been issued on the session, message DFHIR2321 also shows the job name, step name, and procedure name of the client job that is using the session, and the MVS ID of the MVS image on which the client program is running.

Syntax

```
CALL DFHXCIS,(version_number,return_area,user_token,call_type,
             pipe_token),VL,MF=(E,(1))
```

Parameters

version_number
> A fullword binary input area indicating the version of the external CICS interface parameter list being used. It must be set to 1 in the client program.
>
> The equated value for this parameter in the CICS-supplied copybook DFHXCPL*x* (where *x* indicates the language) is VERSION_1. See Table 18 on page 145 for copybook details.

return_area
> A 5-word output area to receive response and reason codes, and a message pointer field. For more details, see "Return area for the EXCI CALL interface" on page 144.

user_token
> The 1-word token returned on the Initialize_User command.

call_type
> A 1-word input area indicating the function of the command. This must be set to 3 in the client program to indicate that this is an Open_pipe command.
>
> The equated value for this call in the CICS-supplied copybook DFHXCPL*x* (where *x* indicates the language) is OPEN_PIPE. See Table 18 on page 145 for copybook details.

pipe_token
> A 1-word output area containing the token passed by CICS on the Allocate_Pipe command. It represents the pipe being opened on this command.

Responses and reason codes

For all non-zero response codes, a unique reason code value identifies the reason for the response.

Note: All numeric response and reason code values are in decimal.

The following is a summary of the response and reason codes that the external CICS interface can return on the Open_Pipe call:

Response OK

Command executed successfully (RC 0). Reason code:

0   NORMAL

Response WARNING

The command executed successfully, but with an error (RC 4). Reason codes:

1   PIPE_ALREADY_OPEN

Response RETRYABLE

The command failed because of setup errors but can be reissued (RC 8). Reason codes:

202   NO_PIPE

203   NO_CICS

Response USER_ERROR

The command failed because of an error in either the client or the server (RC 12). Reason codes:

401   INVALID_CALL_TYPE

402   INVALID_VERSION_NUMBER

404   INVALID_USER_TOKEN

418   INVALID_PIPE_TOKEN

421   RUNNING_UNDER_AN_IRB

Response SYSTEM_ERROR

The command failed (RC 16). Reason codes:

609   IRC_CONNECT_FAILURE

621   PIPE_RECOVERY_FAILURE

622   ESTAE_SETUP_FAILURE

623   ESTAE_INVOKED

For information about response codes, see "EXCI call response code values" on page 143.

For information about the reason codes, see Chapter 19, "Response and reason codes returned on EXCI calls," on page 215.

DPL_Request

Issue a distributed program link request across an open pipe connected to the CICS system on which the server (or target) application program resides.

The command is synchronous, and the TCB waits for a response from CICS. After a pipe is opened, any number of DPL requests can be issued before the pipe is closed. To the server program, the link request appears just like a standard **EXEC CICS LINK** request from another CICS region, and it is not aware that it is sent from a non-CICS client program using EXCI.

The syntax of the call is shown in two forms: the parameters that can be used when *version_number* is set to VERSION_1, and the parameters that can be used when *version_number* is set to VERSION_2.

Syntax

VERSION_1

```
CALL DFHXCIS,(version_number,return_area,user_token,call_type,
              pipe_token,pgmname,COMMAREA,COMMAREA_len,data_len,
              transid,uowid,userid,dpl_retarea,DPL_opts),
              VL,MF=(E,(1))
```

VERSION_2

```
CALL DFHXCIS,(version_number,return_area,user_token,call_type,
              pipe_token,pgmname,COMMAREA,COMMAREA_len,data_len,
              transid,uowid,userid,dpl_retarea,DPL_opts,
              transid2,ccsid,endian),
              VL,MF=(E,(1))
```

Parameters

version_number
> A fullword binary input area that indicates the version of the external CICS interface parameter list being used. This can be set to 1 or 2 in the client program.
>
> The equated value for this parameter in the CICS-supplied copybook DFHXCPL*x* (where *x* indicates the language) is either VERSION_1 or VERSION_2. See "Return area and function call EQUATE copybooks" on page 144 for copybook details.

return_area
> A 5-word output area to receive response and reason codes, and a message pointer field. For more details, see "Return area for the EXCI CALL interface" on page 144.

user_token
> A 1-word input area that specifies the user token returned to the client program on the Initialize_User command.

call_type
> A 1-word input area that indicates the function of the command. This must be set to 6 in the client program to indicate that the pipe is now being used for the DPL_Request call.
>
> The equated value for this call in the CICS-supplied copybook DFHXCPL*x* (where *x* indicates the language) is DPL_REQUEST. See "Return area and function call EQUATE copybooks" on page 144 for copybook details.

pipe_token
> A 1-word input area that specifies the token returned by EXCI on the Allocate_Pipe command. It represents the pipe being used for the DPL_Request call.

pgmname
> The 8-character name of the CICS application program being called as the server program.
>
> This is either the name as specified on a predefined PROGRAM resource definition installed in the CICS server region, or as it is known to a user-written autoinstall program if the program is to be autoinstalled. The program can be defined in the CICS server region as a local program, or it can be defined as remote. Programs defined as remote enable *daisy-chaining*, where EXCI-CICS DPL calls become EXCI-CICS-CICS DPL calls.

COMMAREA (or null_ptr)

A variable length input area for the communications area (COMMAREA) between the client and server programs. The length is defined by *COMMAREA_len*.

This is the storage area that contains the data to be sent to the CICS application program. This area is also used to receive the updated COMMAREA from the CICS application program (the server program).

This parameter is optional. If it is not required, you must ensure that the CALL parameter list contains a null address for this parameter. How you do this depends on the language you are using for the non-CICS client program. For an example of a call that omits an optional parameter, see "Example of EXCI CALLs with null parameters" on page 146.

COMMAREA_len

A fullword binary input area. This parameter specifies the length of the COMMAREA. It is also the length of the server program's COMMAREA (EIBCALEN).

If you specify a COMMAREA, you must also specify this parameter to define the length.

This value should not exceed 32 500 bytes if the COMMAREA is to be passed between any two CICS servers (for any combination of product/version/release). Otherwise, if you are confident that the COMMAREA will not be passed on a further LINK request, you can use a COMMAREA up to 32K in length.

If you do not specify a COMMAREA, this parameter is ignored.

data_len

A fullword binary input area. This parameter specifies the length of contiguous storage, from the start of the COMMAREA, to be sent to the server program.

This parameter restricts the amount of data sent to the server program, and should be used to optimize performance if, for example, the COMMAREA is large but the amount of data being passed is small.

On return from the server program, the EXCI data transformer program ensures that the COMMAREA in the non-CICS client program is the same as that of the server program. This caters for the following conditions:

- The data returned is more than the data passed in the original COMMAREA.
- The data returned is less than the data passed in the original COMMAREA.
- There is no data returned because it is unchanged.
- The server is returning null data.

The value of *data_len* must not be greater than the value of *COMMAREA_len*. A value of zero is valid and results in no data being sent to the server program.

If you do not specify a COMMAREA, this parameter is ignored.

transid (or null_ptr)

A 4-character input area that contains the id of the CICS mirror transaction under which the server program is to run. This transaction must be defined to the CICS server region, and its definition must specify:

PROGRAM(DFHMIRS)

The initial program must be the CICS supplied mirror program DFHMIRS.

Failure to specify DFHMIRS as the initial program means that a COMMAREA passed from the client application program is not passed to the CICS server program. Also, the DPL request fails and the client application program receives a response of SYSTEM_ERROR and reason SERVER_PROTOCOL_ERROR.

PROFILE(DFHCICSA)
The DFHCICSA profile specifies the correct value for the INBFMH parameter, which must be specified as INBFMH(ALL) for a mirror transaction.

When the CICS server region receives a DPL request, it attaches the mirror transaction and invokes DFHMIRS. The mirror program then passes control to the requested server program, passing the COMMAREA supplied by the client program. The COMMAREA passed to the server program is primed with the data only, the remainder of the COMMAREA being set to nulls.

The purpose of the transid parameter is to distinguish between different invocations of the server program. This enables you to run different invocations of the server program under transactions that specify different attributes. For example, you can vary the transaction priorities, or the security requirements.

A transid is optional. By default, the CICS server region uses the CICS-supplied mirror transaction, CSMI. If you do not want to specify *transid*, you must ensure that the CALL parameter list contains a null address for this parameter. How you do this depends on the language you are using for the non-CICS client program. For an example of a call that omits an optional parameter, see "Example of EXCI CALLs with null parameters" on page 146.

If you issue multiple requests in the same MVS unit-of-recovery, the same transid must be used in all of them.

uowid **(or null_ptr)**
An input area that contains a unit-of-work identifier, using the APPC architected format, that is passed on the DPL_Request for correlation purposes.

For DPL requests that are committed when the CICS program returns control to the MVS application, this parameter is optional.

For DPL requests that are part of an RRMS unit-of-recovery, null_ptr must be specified. The unit-of-work identifier that is already associated with the RRMS unit-of-recovery is used, if there is one; if not, CICS (or another RRMS resource manager) generates a unique unit-of-work identifier and associates it with the RRMS unit-of-recovery.

If you do not want to specify a *uowid* parameter , you must ensure that the CALL parameter list contains a null address for this parameter. How you do this depends on the language you are using for the non-CICS client program. For an example of a call that omits an optional parameter, see "Example of EXCI CALLs with null parameters" on page 146.

The *uowid* parameter is passed to the CICS server region, which uses it as the UOWID for the first unit of work executed by the CICS server program. If the server program issues intermediate sync points before returning to the client program, CICS uses the supplied *uowid* for the subsequent units of work, but with the two-byte sequence number incremented for each new logical unit of work. If the CICS server program updates remote resources, the client-supplied UOWID is distributed to the remote systems that own the resources.

The *uowid* parameter is supplied on the EXCI CALL interface for correlation purposes only, to allow units of work that originated from a particular client program to be identified in CICS. The *uowid* is not provided for recovery purposes between CICS and the client program.

The *uowid* can be a maximum of 27 bytes long and has the following format:

- A 1-byte length field that contains the overall length of the UOWID (excluding this field).
- A 1-byte length field that contains the length of the logical unit name (excluding this field).
- A logical unit name field of variable length up to a maximum of 17 bytes.

  To conform to APPC architecture rules, the LUNAME must be of the form *AAAAAAAA.BBBBBBBB*, where *AAAAAAAA* is optional and:

  - AAAAAAAA and BBBBBBBB are 8-byte names separated by a period
  - If AAAAAAAA is omitted, the period must also be omitted
  - AAAAAAAA and BBBBBBBB must be type-1134 symbol strings (that is, character strings consisting of one or more EBCDIC uppercase letters A - Z and 0 - 9, the first character of which must be an uppercase letter).

- The clock value; the middle 6 bytes of an 8-byte store clock (STCK) value.
- A 2-byte sequence number.

If you omit a unit-of-work identifier (by specifying a null pointer), and the DPL request is not part of an RRMS unit-of-recovery, the external CICS interface generates one for you, consisting of the following:

- A 1-byte length field set to X'1A'.
- A 1-byte LU length field set to X'11'.
- A 17-byte LU name consisting of:

  - An 8-byte eyecatcher set to DFHEXCIU.
  - A 1-byte field that contains a period (.)
  - A 4-byte field that contains the MVS system identifier (SYSID), in characters, under which the client is running.
  - A 4-byte field that contains the address space id (ASID) in which the MVS client program is running. The field contains the four character EBCDIC representation of the 2-byte hex address space id.

- The clock value; the middle 6 bytes of an 8-byte store clock (STCK) value
- A 2-byte sequence number set to X'0001'.

userid **(or null_ptr)**

An 8-character input area that contains the RACF userid for user security checking in the CICS region. The external CICS interface passes this userid to the CICS server region for user resource and command security checking in the server application program.

A userid is required only if the MRO connection specifies the ATTACHSEC(IDENTIFY) attribute. If the connection specifies ATTACHSEC(LOCAL), the CICS server region applies link security checking. See Intercommunication security in the RACF Security Guide for information about link security on MRO connections.

See also Chapter 17, "EXCI security," on page 187 for information about external CICS interface security considerations.

This parameter is optional. However, if you do not specify a userid, the external CICS interface passes the security userid under which the client

program is running. For example, if the client program is running as an MVS batch job, the external CICS interface obtains and passes the userid specified on the USER parameter of the JOB statement.

If you specify a userid and SURROGCHK=YES is specified in the EXCI options table DFHXCOPT, the userid under which the EXCI job is running is subject to a surrogate user check. This check is performed by the external CICS interface to ensure that the client job userid is authorized to use the userid specified on the DPL call. For more information about surrogate user security checking, see Chapter 17, "EXCI security," on page 187.

If you want to let *userid* default, you must ensure that the CALL parameter list contains a null address for this parameter. How you do this depends on the language you are using for the non-CICS client program. For an example of a call that omits an optional parameter, see "Example of EXCI CALLs with null parameters" on page 146.

If you issue multiple requests in the same MVS unit-of-recovery, the same userid must be used in all of them. If the unit-of-recovery also includes EXEC CICS calls, you should allow the userid on all DPL_requests to default to the security userid under which the client program is running.

dpl_retarea
A 12-byte output area into which the DPL_Request processor places responses to the DPL request. Generally, these responses are from CICS, but in some cases the error detection occurs in the external CICS interface, which returns exception conditions that are the equivalent of those returned by an EXEC CICS LINK command.

This field is only meaningful in the following circumstances:

- The response field of the EXCI return-area has a zero value
- The EXCI return-area indicates that the server program has abended (response=USER_ERROR and reason=SERVER_ABENDED).

The 12 bytes form three fields, providing the following information:

Field 1 (*fullword value***)**
This field is a fullword that contains a RESP value from the DPL_Request call. See "Error codes" on page 151 for the RESP values that can be returned on a DPL_Request call.

If the DPL_Request call reaches CICS, this field contains the EIBRESP value, otherwise it contains an equivalent response set by the external CICS interface. If this field is set by the external CICS interface, RESP is further qualified by a RESP2 value in the second field.

A zero value is the normal response, which equates to EXEC_NORMAL in the return codes copybooks.

Field 2 (*fullword value***)**
This field is a fullword that can contain a RESP2 value from the link request, further qualifying the RESP value in field 1.

If the DPL_Request call reaches CICS, the RESP2 field generally is null (CICS does not return RESP2 values across MRO links). However, if the RESP field indicates SYSIDERR (value 53), this field provides a reason code. See"Dpl_retarea return codes" on page 145 for more information.

If the RESP field is set by the external CICS interface, it is further qualified by a RESP2 value in this second field. For example, if the *data_len*

parameter specifies a value greater than the *COMMAREA_len* parameter, the external CICS interface returns the RESP value 22 (which equates to EXEC_LENGERR in the return codes copybooks), and a RESP2 value of 13.

See the LINK conditions in LINK in CICS Application Programming for full details of the possible RESP and RESP2 values.

Note: The data transformer program makes special use of the RESP2 field. If any error occurs in the transformer, the error is returned in RESP2.

Field 3 (*fullword value***)**

The third field, a 4-character field, contains the following information:

- The abend code if the server program abended
- Four blanks if the server program did not abend.

If a server program abends, it is backed out to its last syncpoint, which can be the start of the task, or an intermediate syncpoint. The server program can issue intermediate syncpoints because SYNCONRETURN is forced.

DPL_opts **(or null_ptr)**

A 1-byte input area that indicates options to be used on the DPL_Request call.

If you omit this parameter, it defaults to the value X'00' (see below). If you want to omit *DPL_opts* and let it default, ensure that the CALL parameter list contains a null address for this parameter. How you do this depends on the language you are using for the non-CICS program. For an example of a call that omits an optional parameter, see "Example of EXCI CALLs with null parameters" on page 146.

Currently, the *DPL_opts* parameter applies only to resource recovery, using the following values:

X'00' Indicates that you specified NOSYNCONRETURN, because you want the client batch program to control resource recovery, using 2-phase commit protocols supported by MVS RRS. With this option, the CICS server region does not perform a syncpoint when the server program returns control to CICS. Furthermore, the CICS server application program must not take any explicit syncpoints, otherwise it is abended by CICS. For more information, see "Resource recovery" on page 118.

X'80' indicates that SYNCONRETURN is required in the CICS server region.

SYNCONRETURN specifies that the server region is to take a syncpoint on successful completion of the server program, independently of the client program. This option does not prevent a server program from taking its own explicit syncpoints.

The equated value for this parameter in the CICS-supplied copybook DFHXCPL*x* (where *x* indicates the language) is SYNCONRETURN. See "Return area and function call EQUATE copybooks" on page 144 for copybook details.

transid2 **(or null_ptr) VERSION_2 only**

A 4-character input area that contains a CICS transaction id.

The server program runs under a CICS supplied mirror transaction, CSMI or CPMI. However the transaction id made available to the server program through the EIBTRNID field in the Exec Interface Block is the one specified by the transid2 parameter. The transid2 parameter is ignored if the transid parameter is specified. The following table gives an example of different combinations of transid and transid2:

Table 16. Use of transid2

transid	transid2	program executes under	EIBTRNID seen by program
UTRN	omitted	UTRN	UTRN
UTRN	UEIB	UTRN	UTRN
omitted	omitted	CSMI	CSMI
omitted	UEIB	CSMI	UEIB

The transid2 parameter is useful for server programs that access DB2, because EIBTRNID is used to determine which DB2ENTRY definition to use. Previously, EIBTRNID could only be set by using transid, which then required you to define a mirror transaction to CICS. Using transid2, EIBTRNID is set, but the mirror program executes under the CICS provided definition CSMI.

ccsid **(or null_ptr)VERSION_2 only**
A fullword binary input area that indicates the Coded Character Set Identifier (CCSID) of the character data contained in the COMMAREA. The ccsid parameter must be specified if character data has to be converted when the COMMAREA is passed to, or returned from, the server program. The parameter can take the following values:

-1 (X'FFFFFFFF')
Indicates that conversion of character data is required and that the source CCSID is defined in the conversion template installed in the server.

1 <= ccsid <= 65535
Indicates that conversion of character data is required and that the value specified overrides the source CCSID defined in the conversion template installed in the server.

endian **(or null_ptr) VERSION_2 only**
A fullword binary input area that indicates the format, big endian or little endian, for binary data contained in the COMMAREA. Big endian indicates that the leftmost byte contains the most significant digits, as used, for example, in System 390 architecture. Little endian indicates that the rightmost byte contains the most significant digits, as used, for example, in Intel architecture. The endian parameter should be specified if binary data has to be converted when the COMMAREA is passed to, or returned from, the server program. If the ccsid parameter indicates that conversion is required, but endian is not specified (defaults to null), conversion of binary data depends on what is specified in the DFHCNV conversion template installed in the server. The parameter can take the following values:

16909060 (X'01020304')
Binary data is held in big endian format.

67305985 (X'04030201')
Binary data is held in little endian format.

Responses and reason codes

For all non-zero response codes, a unique reason code value identifies the reason for the response.

Note: All numeric response and reason code values are in decimal.

The following is a summary of the response and reason codes that the external CICS interface can return on the DPL call:

Response OK

Command executed successfully (RC 0). Reason code:

0   NORMAL

Response WARNING

The command executed successfully, but with an error (RC 4). Reason codes:

6   IRP_IOAREA_FM_FAILURE

7   SERVER_TERMINATED

Response RETRYABLE

The command failed because of setup errors but can be reissued (RC 8). Reason codes:

203   NO_CICS

204   WRONG_MVS_FOR_RRMS

205   RRMS_NOT_AVAILABLE

Response USER_ERROR

The command failed because of an error in either the client or the server (RC 12). Reason codes:

401   INVALID_CALL_TYPE

402   INVALID_VERSION_NUMBER

404   INVALID_USER_TOKEN

406   PIPE_NOT_OPEN

407   INVALID_USERID

408   INVALID_UOWID

409   INVALID_TRANSID

414   IRP_ABORT_RECEIVED

415   INVALID_CONNECTION_DEFN

416   INVALID_CICS_RELEASE

417   PIPE_MUST_CLOSE

418   INVALID_PIPE_TOKEN

421   RUNNING_UNDER_AN_IRB

422   SERVER_ABENDED

423   SURROGATE_CHECK_FAILED

425   UOWID_NOT_ALLOWED

426   INVALID_TRANSID2

427   INVALID_CCSID

428   INVALID_ENDIAN

Response SYSTEM_ERROR

The command failed (RC 16). Reason codes:

612	TRANSFORM_1_ERROR
613	TRANSFORM_4_ERROR
614	IRP_NULL_DATA_RECEIVED
615	IRP_NEGATIVE_RESPONSE
616	IRP_SWITCH_PULL_FAILURE
617	IRP_IOAREA_GM_FAILURE
619	IRP_BAD_IOAREA
620	IRP_PROTOCOL_ERROR
622	ESTAE_SETUP_FAILURE
623	ESTAE_INVOKED
624	SERVER_TIMEDOUT
625	STIMER_SETUP_FAILURE
626	STIMER_CANCEL_FAILURE
629	SERVER_PROTOCOL_ERROR
630	RRMS_ERROR
631	RRMS_SEVERE_ERROR
632	XCGUR_GETMAIN_ERROR

For information about response codes, see "EXCI call response code values" on page 143.

For information about the reason codes, see Chapter 19, "Response and reason codes returned on EXCI calls," on page 215.

Close_PIPE

Disconnect an open pipe from CICS. The pipe remains in an allocated state, and its tokens remain valid for use by the same user. To reuse a closed pipe, the client program must first reissue an Open_Pipe command using the pipe token returned on the Allocate_Pipe command for the pipe.

Pipes should not be left open when not in use because this prevents CICS from shutting down its IRC facility in an orderly manner. Therefore, the Close_Pipe command should be issued as soon as possible after all DPL_Request calls have completed.

Syntax

```
CALL DFHXCIS,(version_number,return_area,user_token,call_type,
              pipe_token),VL,MF=(E,(1))
```

Parameters

version_number
A fullword binary input area indicating the version of the external CICS interface parameter list being used. It must be set to 1 in the client program.

The equated value for this parameter in the CICS-supplied copybook DFHXCPL*x* (where *x* indicates the language) is VERSION_1. See "Deallocate_Pipe" on page 142 for copybook details.

return_area
>A 5-word output area to receive response and reason codes, and a message pointer field. For more details, see "Return area for the EXCI CALL interface" on page 144.

user_token
>The 1-word input area specifying the token, returned to the client program by EXCI on the Initialize_User command, that represents the user of the pipe being closed.

call_type
>A 1-word input area indicating the function of the command. This must be set to 4 in the client program to indicate that this is a Close_Pipe command.
>
>The equated value for this call in the CICS-supplied copybook DFHXCPL*x* (where *x* indicates the language) is CLOSE_PIPE. See "Return area and function call EQUATE copybooks" on page 144 for copybook details.

pipe_token
>A 1-word input area specifying the token, returned to the client program by EXCI on the original Allocate_Pipe command, that represents the pipe being closed.

Responses and reason codes

For all non-zero response codes, a unique reason code value identifies the reason for the response.

Note: All numeric response and reason code values are in decimal.

The following is a summary of the response and reason codes that the external CICS interface can return on the Close_Pipe call:

Response OK
>Command executed successfully (RC 0). Reason code:
>
>| | |
>|---|---|
>| **0** | NORMAL |

Response WARNING
>The command executed successfully, but with an error (RC 4). Reason codes:
>
>| | |
>|---|---|
>| **2** | PIPE_ALREADY_CLOSED |

Response USER_ERROR
>The command failed because of an error in either the client or the server (RC 12). Reason codes:
>
>| | |
>|---|---|
>| **401** | INVALID_CALL_TYPE |
>| **402** | INVALID_VERSION_NUMBER |
>| **404** | INVALID_USER_TOKEN |
>| **418** | INVALID_PIPE_TOKEN |
>| **421** | RUNNING_UNDER_AN_IRB |

Response SYSTEM_ERROR
>The command failed (RC 16). Reason codes:
>
>| | |
>|---|---|
>| **610** | IRC_DISCONNECT_FAILURE |
>| **622** | ESTAE_SETUP_FAILURE |

For information about response codes, see "EXCI call response code values" on page 143.

For information about the reason codes, see Chapter 19, "Response and reason codes returned on EXCI calls," on page 215.

Deallocate_Pipe

Deallocate a pipe from CICS. On completion of this command, the pipe can no longer be used, and its associated tokens are invalid. This command should be issued for pipes that are no longer required. This command frees storage associated with the pipe.

Syntax

```
CALL DFHXCIS,(version_number,return_area,user_token,call_type,
             pipe_token),VL,MF=(E,(1))
```

Parameters

version_number
> A fullword binary input area indicating the version of the external CICS interface parameter list being used. It must be set to 1 in the client program.
>
> The equated value for this parameter in the CICS-supplied copybook DFHXCPL*x* (where *x* indicates the language) is VERSION_1. See "Return area and function call EQUATE copybooks" on page 144 for copybook details.

return_area
> A 5-word output area to receive response and reason codes, and a message pointer field. For more details, see "Return area for the EXCI CALL interface" on page 144.

user_token
> A 1-word input area containing the token returned on the Initialize_User command.

call_type
> A 1-word input area indicating the function of the command. This must be set to 5 in the client program to indicate that this is a Deallocate_Pipe command.
>
> The equated value for this call in the CICS-supplied copybook DFHXCPL*x* (where *x* indicates the language) is DEALLOCATE_PIPE. See "Return area and function call EQUATE copybooks" on page 144 for copybook details.

pipe_token
> A 1-word input area containing the token passed back on the original Allocate_Pipe command, that represents the pipe now being deallocated.

Responses and reason codes

For all non-zero response codes, a unique reason code value identifies the reason for the response.

Note: All numeric response and reason code values are in decimal.

The following is a summary of the response and reason codes that the external CICS interface can return on the Deallocate_Pipe call:

Response OK

> Command executed successfully (RC 0). Reason code:

> **0**   NORMAL

Response WARNING

> The command succeeded successfully, but with an error (RC 4). Reason codes:

> **5**   XCPIPE_FREEMAIN_ERROR

> **6**   IRP_IOAREA_FM_FAILURE

Response USER_ERROR

> The command failed because of an error in either the client or the server (RC 12). Reason codes:

> **401**   INVALID_CALL_TYPE

> **402**   INVALID_VERSION_NUMBER

> **404**   INVALID_USER_TOKEN

> **405**   PIPE_NOT_CLOSED

> **418**   INVALID_PIPE_TOKEN

> **421**   RUNNING_UNDER_AN_IRB

Response SYSTEM_ERROR

> The command failed (RC 16). Reason codes:

> **611**   IRC_LOGOFF_FAILURE

> **622**   ESTAE_SETUP_FAILURE

> **623**   ESTAE_INVOKED

For information about response codes, see "EXCI call response code values."

For information about the reason codes, see Chapter 19, "Response and reason codes returned on EXCI calls," on page 215.

EXCI call response code values

This table shows the values that can be returned in the response field. All values are in decimal.

Table 17. EXCI response codes (returned in response field of return_area)

Code	Meaning	Explanation
0	OK	For all EXCI CALL commands other than the DPL_request, the call was successful. If an OK response is received for a DPL_request, you must also check *dpl_retarea* to ensure CICS did not return a condition code. If the EIBRESP field of *Dpl_retarea* is zero, the DPL call was successful.
4	WARNING	The external CICS interface detected an error, but this did not stop the CALL command completing successfully. The reason code field describes the error detected.

Table 17. EXCI response codes (returned in response field of return_area) (continued)

Code	Meaning	Explanation
8	RETRYABLE	The EXCI CALL command failed. This class of failure relates to errors in the setup of the system environment, and not errors in the external CICS interface or client program. The reason code documents the specific error in the environment setup.
		The external CICS interface command can be reissued without changing the client program once the environment error has been corrected. The environmental errors concerned are ones that do not require an MVS re-IPL. Each reason code value for a RETRYABLE response documents whether the CALL can be reissued directly, or whether the pipe being used has to be closed and reopened first.
12	USER_ERROR	The EXCI CALL command failed. This class of error means there is an error either in the client program, or in the CICS server program, or in the CICS server region. An example of an error in the CICS server system would be a failed security check, or an abend of the CICS server program, in which case the abend code is set in the abend code field of *dpl_retarea*. Each reason code value for a response of USER_ERROR explains whether the command can be reissued directly, or whether the pipe being used has to be closed and reopened first.
16	SYSTEM_ ERROR	The EXCI CALL command failed. This class of error means that the external CICS interface has detected an error. The reason code value identifies the specific error. If the error can be corrected, then the command can be reissued. Each reason code value for a SYSTEM_ERROR response explains whether the command can be reissued directly, or whether the pipe being used has to be closed and reopened first.

Return area for the EXCI CALL interface

This is the format of the 5-word return area for the EXCI CALL interface.

1. 1–word response field.
2. 1–word reason field.
3. Two 1–word subreason fields—subreason field-1 and subreason field-2.
4. 1–word CICS message pointer field. This is zero if there is no message present. If a message is present, this field contains the address of the storage area containing the message, which is formatted as follows:

   - A 2-byte LL field. LL is the length of the message plus the length of the LLBB field.
   - A 2-byte BB field, set to binary zero.
   - A variable length field containing the text of the message.

Return area and function call EQUATE copybooks

CICS provides four language-specific copybooks that map the storage areas for the *return_area* and *dpl_retarea* parameters of the EXCI CALL commands. The copybooks also provide EQUATE statements for each type of EXCI CALL.

These copybooks, and the libraries they are supplied in for the supported languages, are shown in Table 18 on page 145.

Table 18. Supplied copybooks of return areas and equated names

Copybook name	Language	Library
DFHXCPLD	Assembler	CICSTS42.CICS.SDFHMAC
DFHXCPLH	C	CICSTS42.CICS.SDFHC370
DFHXCPLO	COBOL	CICSTS42.CICS.SDFHCOB
DFHXCPLL	PL/I	CICSTS42.CICS.SDFHPL1

Return codes

All the possible return codes are contained in a CICS-supplied copybook, which you must include in the program source of your external, non-CICS program.

The names of the copybooks for the supported languages, and the libraries that they are supplied in, are shown in Table 19.

Table 19. Supplied copybooks of RESPONSE and REASON codes

Copybook name	Language	Library
DFHXCRCD	Assembler	CICSTS42.CICS.SDFHMAC
DFHXCRCH	C	CICSTS42.CICS.SDFHC370
DFHXCRCO	COBOL	CICSTS42.CICS.SDFHCOB
DFHXCRCL	PL/I	CICSTS42.CICS.SDFHPL1

z/OS provides copybooks for use with the interfaces described in "Use of sync points in the client program" on page 122. These are described in *z/OS MVS Programming: Resource Recovery* and *z/OS MVS Programming: Callable Services for High-Level Languages*.

Dpl_retarea return codes

These are the same as for CICS-to-CICS EXEC CICS DPL commands but with the following additions for the EXCI call interface.

Table 20. Exceptional conditions. RESP and RESP2 values returned to dpl_retarea

Condition	RESP2	Meaning
LENGERR	22	COMMAREA_LEN_TOO_BIG
LENGERR	23	COMMAREA_BUT_NO_COMMAREA_LEN

SYSIDERR also can be returned on an EXCI DPL_Request, if the DPL_Request specifies a program defined in the CICS server region as a remote program, and the link between the server and the remote CICS region is not open. In this situation, SYSIDERR is returned in the first word of the DPL_Retarea (code 53). The reason code qualifying SYSIDERR is placed in the second word of this area (the equivalent of a RESP2 value).For SYSIDERR, the information stored in this field is derived from bytes 1 and 2 of the CICS EIBRCODE field. For example, if a remote link is not available, the EIBRCODE value stored in bytes 2 and 3 of the DPL_Retarea RESP2 field is X'0800'. For a list of the SYSIDERR reason codes that can be returned in the RESP2 field, see the SYSIDERR section of the notes on EIBRCODE in Attention identifier constants, DFHAID, in the *CICS Application Programming Reference* manual.

TERMERR also may be returned on an EXCI DPL request if the DPL request was to a program defined as remote, and an unrecoverable error occurs during

conversation with the mirror on the remote CICS system. For example, suppose client program BATCH1 issues an EXCI DPL request to CICSA for program PROG1, which is defined as remote, and the request is function-shipped to CICSB where the program resides. If the session between CICSA and CICSB fails, or CICSB itself fails while executing the program PROG1, then TERMERR is returned to CICSA, and in turn to BATCH1.

No unique EXCI_DPL_RESP2 values are returned for TERMERR, PGMIDERR, NOTAUTH, and ROLLBACK.

Example of EXCI CALLs with null parameters

If you omit an optional parameter, such as *userid* on a DPL_Request, you must ensure that the parameter list is built with a null address for the missing parameter.

The example that follows illustrates how to issue an EXCI DPL_Request with the *userid* and *uowid* parameters omitted in a COBOL program.

DPL CALL without userid and uowid (COBOL): In this example, the DPL parameters used on the call are defined in the WORKING-STORAGE SECTION, as follows:

DPL parameter	COBOL variable	Field definition
version_number	01 VERSION-1	PIC S9(8) COMP VALUE 1.
return_area	01 EXCI-RETURN-CODE.	(structure)
user_token	01 USER-TOKEN	PIC S9(8) COMP VALUE ZERO.
call_type	03 DPL-REQUEST	PIC S9(8) COMP VALUE 6.
pipe_token	01 PIPE-TOKEN	PIC S9(8) COMP VALUE ZERO.
pgmname	01 TARGET-PROGRAM	PIC X(8) VALUE "DFHœAXCS".
commarea	01 COMMAREA.	(structure)
commarea_len	01 COMM-LENGTH	PIC S9(8) COMP VALUE 98.
data_len	01 DATA-LENGTH	PIC S9(8) COMP VALUE 18.
transid	01 TARGET-TRANSID	PIC X(4) VALUE "EXCI".
dpl_retarea	01 EXCI-DPL-RETAREA.	(structure)
dpl_opts	01 SYNCONRETURN	PIC X VALUE X'80'.

The variable used for the null address is defined in the LINKAGE SECTION, as follows:

```
    LINKAGE  SECTION.
      01   NULL-PTR          USAGE  IS  POINTER.
```

Using the data names specified in the WORKING-STORAGE SECTION as described above, and the NULL-PTR name as described in the LINKAGE SECTION, the following invocation of the DPL function omits the *uowid* and the *userid* parameters, and replaces them in the parameter list with the NULL-PTR variable:

```
    DPL-SECTION.
  *
      SET ADDRESS OF NULL-PTR TO NULLS.
  *
      CALL 'DFHXCIS' USING  VERSION-1   EXCI-RETURN-CODE  USER-TOKEN
                            DPL-REQUEST    PIPE-TOKEN   TARGET-PROGRAM
```

```
COMMAREA        COMM-LENGTH  DATA-LENGTH
TARGET-TRANSID NULL-PTR      NULL-PTR
EXCI-DPL-RETAREA    SYNCONRETURN.
```

This example is taken from the CICS-supplied sample external CICS interface program, DFH0CXCC, which is supplied in CICSTS42.CICS.SDFHSAMP. For an example of how to omit the same parameters from the DPL call in the other supported languages, see the following sample programs:

DFH$AXCC
> The assembler sample

DFH$PXCC
> The PL/I sample

DFH$DXCC
> The C sample.

Chapter 10. The EXCI EXEC CICS interface

The external CICS interface provides this as a single, composite command, to invoke all the calls of the EXCI CALL interface. Each time you issue an EXEC CICS LINK PROGRAM command in a client application program, the external CICS interface invokes on your behalf each of the six EXCI calls.

This chapter describes:
- "Using EXEC CICS LINK command"
- "Retries on an EXEC CICS LINK command" on page 153
- "Translation required for EXEC CICS LINK command" on page 155

Using EXEC CICS LINK command

Link from an MVS client program to the specified server program in a server CICS region.

Format

LINK

```
►►──LINK──PROGRAM──(──name──)──RETCODE──(──data-area──)──────────────────────►
                                               └─SYNCONRETURN─┘

►──────────────────────────────────────────────────────────────────────────►
   └─COMMAREA──(──data-area──)──LENGTH──(──data-value──)─┘

►──────────────────────────────────────────────────────────────────────────►
   └─APPLID──(──name──)─┘  └─TRANSID──(──name──)─┘

►──────────────────────────────────────────────────────────────────────────►◄
   └─DATALENGTH──(──data-value──)─┘
```
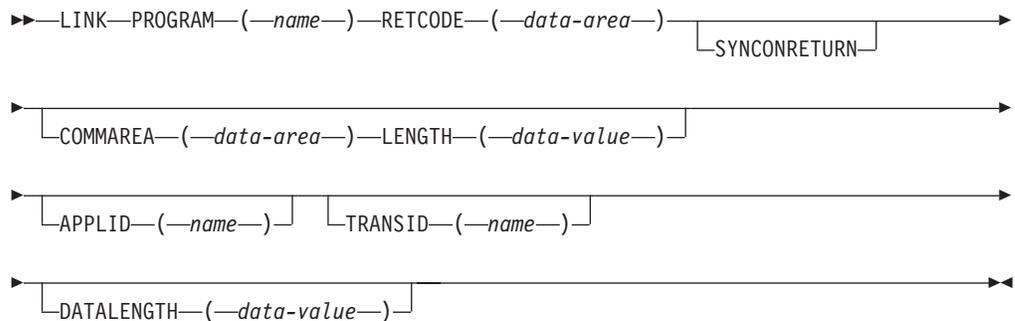
Error conditions:LENGERR, LINKERR, NOTAUTH, PGMIDERR, RESUNAVAIL, ROLLEDBACK, SYSIDERR, TERMERR, WARNING

Comments

With the exception of the APPLID and RETCODE parameters, the external CICS interface parameters for an EXEC CICS LINK command are the same as for a CICS-CICS DPL command.

This information describes only those parameters that you can use with the external CICS interface. For programming information about the **EXEC CICS LINK PROGRAM** command, see LINK in CICS Application Programming .

Note that the LENGTH and DATALENGTH parameters specify halfword binary values, unlike the corresponding *COMMAREA_len* and *data_len* parameters of the EXCI CALL interface, which specify fullword values.

An external CICS interface **EXEC CICS LINK** command always uses a generic connection.

Parameters

The parameters that you can use on the external CICS interface form of the LINK command, are as follows:

APPLID(*name*)

Specifies the generic APPLID of the target CICS server region.

Although an applid is required for an external CICS interface command, this parameter is optional on the LINK command itself because you can also specify it in the user-replaceable module, DFHXCURM. If you omit the generic APPLID from the LINK command, you must ensure it is specified by the user-replaceable module, DFHXCURM, on the URMAPPL parameter. You can also use the URMAPPL parameter in DFHXCURM to override an applid specified on the LINK command. See Chapter 14, "The EXCI user-replaceable module," on page 167 for information about the URMAPPL parameter.

COMMAREA(*data-area*)

Specifies a communication area that is to be made available to the invoked program. In this option, a pointer to the data area is passed.

See the *CICS Application Programming Guide* for more information about passing data to CICS application programs.

DATALENGTH(*data-value*)

Specifies a halfword binary value that is the length of a contiguous area of storage from the start of the COMMAREA. If the amount of data in a COMMAREA is small, but the COMMAREA itself is large, specify DATALENGTH to improve performance.

LENGTH(*data-value*)

Specifies a halfword binary value that is the length in bytes of the COMMAREA.

This value should not exceed 32 500 bytes if the COMMAREA is to be passed between any two CICS servers (for any combination of product/version/release), otherwise, if you are confident that the COMMAREA will not be passed on a further LINK request, you can use a COMMAREA up to 32K in length.

PROGRAM(*name*)

Specifies the program name (1-8 characters) of the CICS server application program to which control is to be passed unconditionally. The specified name must either have been defined as a program to CICS, or the CICS server region must be capable of autoinstalling a definition for the named program.

Note the use of quotes:

```
EXEC CICS LINK PROGRAM('PROGX')
```

PROGX is in quotes because it is the program name.

```
EXEC CICS LINK PROGRAM(DAREA)
```

DAREA is not in quotes because it is the name of a data area that contains the 8-character program name.

RETCODE(*data-area*)

Specifies a 20-byte area into which the external CICS interface places return code information. This area is formatted into five 1–word fields as follows:

RESP The primary response code indicating whether the external CICS interface LINK command caused an exception condition during its execution.

RESP2
>> The secondary response code that further qualifies, where necessary, some of the conditions raised in the RESP parameter.

ABCODE
>> Contains a valid CICS abend code if the server program abended in the server region.

MSGLEN
>> Indicates the length of the message (if any) issued by the CICS server region during the execution of the server program. Note that the length is the actual length of the message text only, and does not include this 1—word length field.

MSGPTR
>> This is the address of the message text returned by the CICS server region.

> **Note:** MSGLEN and MSGPTR are only valid on a LINKERR condition, with the RESP2 value 414.

SYNCONRETURN
> Specifies that the CICS server region, named on the APPLID parameter, is to take a syncpoint on successful completion of the server program.

TRANSID(*name***)**
> Specifies the name of the mirror transaction that the remote region is to attach, and under which it is to run the server program. If you omit the TRANSID option, the CICS server region attaches CSMI.

> **Note:** The TRANSID option specified on the LINK command overrides any TRANSID option specified on the program resource definition installed in the CICS server region.

> While you can specify your own name for the mirror transaction initiated by DPL requests, the transaction *must* be defined in the server region, and the transaction definition must specify the mirror program, DFHMIRS. Defining your own transaction to invoke the mirror program gives you the freedom to specify appropriate values for some other options on the transaction resource definition.

> See also the important rules about specifying transid with a DPL_Request on transid, parameter of DPL_Request command.

Error codes

Most of the exception conditions that are returned on the external CICS interface LINK command are the same as for the CICS-to-CICS distributed program link command. Those that are the same, and their corresponding numeric values are as follows:

LENGERR
> 22

PGMIDERR
> 27

SYSIDERR
> 53

NOTAUTH
70

TERMERR
81

ROLLEDBACK
82

RESUNAVAIL
121

These exception condition codes are returned in the RESP field.

RESP and RESP2: References to the RESP and RESP2 fields in this section are to the first two fields of the RETCODE parameter.

The exception conditions that are specific to the external CICS interface are as follows:

- The RESP2 values on the error condition LENGERR is specific to the external CICS interface.
- The exception conditions WARNING and LINKERR are specific to the external CICS interface.

The WARNING and LINKERR exceptions are a result of responses to individual EXCI calls issued by the external CICS interface in response to an EXEC CICS LINK command. These WARNING and LINKERR exceptions correspond to EXCI call responses as follows:

WARNING (RESP value 4)
> This is returned when the EXCI module handling the EXEC CICS LINK request receives a USER_ERROR or SYSTEM_ERROR response to a Close_Pipe or Deallocate_Pipe request issued on behalf of an EXEC CICS LINK command. The RESP value is set to WARNING because the DPL request to CICS completed successfully, but an error occurred in subsequent processing.
>
> The RESP2 field is set to the EXCI reason code, which gives more information about the error.

LINKERR (RESP value 88)
> This is returned when the EXCI module handling the EXEC CICS LINK request receives a RETRYABLE, USER_ERROR, or SYSTEM_ERROR response to an EXCI call issued on behalf of the EXEC CICS LINK command. The DPL request has failed. The RESP2 field is set to the EXCI reason code, which gives more information about the error.

See Chapter 19, "Response and reason codes returned on EXCI calls," on page 215 for descriptions of EXCI reason codes.

Note: The external CICS interface ignores any WARNING conditions that occur in response to EXCI calls it issues on behalf of an EXEC CICS LINK command. It treats the WARNING on an EXCI call as a good response and continues normally. If no other errors occur, the EXEC CICS command completes with a zero response in the EXEC_RESP field.

Retries on an EXEC CICS LINK command

If the external CICS interface receives a RETRYABLE response on an EXCI call that it makes on behalf of an EXEC CICS LINK command, it automatically retries the EXEC CICS LINK command up to five times, providing more serious errors do not occur.

If the RETRYABLE response is still received after the fifth retry, the RESP field is set to LINKERR, and the reason returned on the EXCI CALL request that causes the exception is returned in the RESP2 field.

The external CICS interface retries the EXEC CICS LINK command by first closing and deallocating the pipe, then reissuing the six EXCI CALL commands. During Allocate_Pipe processing, the EXCI CALL interface calls the user-replaceable module, DFHXCURM, to give you the opportunity to change the APPLID of the CICS system to which the request has been sent. See Chapter 14, "The EXCI user-replaceable module," on page 167 for details of DFHXCURM.

Table 21 lists all the exception conditions and RESP2 values that are specific to the EXEC CICS LINK command for the external CICS interface.

Table 21. Exceptional conditions. RESP and RESP2 values returned from the EXEC API.

Condition (RESP)	RESP2	Meaning
LENGERR (22)	22	COMMAREA length greater than 32763 bytes specified
	23	COMMAREA specified but no LENGTH parameter specified
WARNING (4)	401	Invalid *call_type* parameter value specified on Close_Pipe or Deallocate_Pipe call
	402	Invalid *version_number* parameter specified on Close_Pipe or Deallocate_Pipe call
	404	Invalid *user_token* specified on Close_Pipe or Deallocate_Pipe call
	405	A Deallocate_Pipe call has been issued against a pipe that is not yet closed
	418	An invalid pipe token has been issued on a Close_Pipe or Deallocate_Pipe call
	421	A Close_Pipe or Deallocate_Pipe command has been issued under an IRB
	610	There has been a CICS IRP logoff failure on a Deallocate_Pipe call
	611	There has been a CICS IRC disconnect failure on a Close_Pipe call
	622	There has been an MVS ESTAE setup failure on a Close_Pipe or Deallocate_Pipe call
	623	A program check on a Close_Pipe or Deallocate_Pipe call has caused the ESTAE to be invoked
LINKERR (88)	201	Command has been issued on an MVS image which has had no IRC activity since the previous IPL
	202	There are no available sessions
	203	CICS has not yet been brought up, or (2) has not yet opened IRC, or (3) no generic connection is installed, or (4) no specific connection is installed with the required netname.
	204	An EXEC CICS LINK command without the SYNCONRETURN option has been issued specifying a CICS system on a different MVS image.
	205	An EXEC CICS LINK command without the SYNCONRETURN option has been issued when RRS is not available
	401	Invalid parameter

Table 21. Exceptional conditions (continued). RESP and RESP2 values returned from the EXEC API.

Condition (RESP)	RESP2	Meaning
	402	Invalid version number
	403	User name is all blanks
	404	Invalid address in user token
	405	Command has been issued against a pipe that is not closed
	406	Command has been issued against a pipe that is not open
	407	Userid of all blanks has been passed
	408	Error in UOWID parameter
LINKERR (88)	409	Transid consisting of all blanks or zero has been passed
	410	Load of message module, DFHMEBMX, failed
	411	Load of message module, DFHMET4E, failed
	412	Load of DFHXCURM failed
	413	Load of DFHXCTRA failed
	414	If run as a CICS-to-CICS linked-to program, this server program would have resulted in an error with an appropriate message sent to the terminal. Running the program as an EXCI server program returns the message addressed by the MSGPTR field of the RETCODE area.
	415	Target connection is an MRO connection, not an EXCI connection
	416	Command has been issued against a CICS region running under a release of CICS earlier than CICS for MVS/ESA 4.1
	417	Command has been issued against a pipe in the MUST CLOSE state. Further EXCI EXEC CICS LINK commands will have unpredictable results and are, therefore, not permitted
	418	Pipe_token does not address an XCPIPE control block, or there is a mismatch between user_token and pipe_token
	419	CICS runs, or did run, under the TCB that this command is attempting to use. This is not permitted and the command fails
	420	Load of DFHXCOPT failed
	421	The command has been issued under an MVS IRB, which is not permitted
	422	The server has abended
	423	Surrogate user check failed
	424	An EXEC CICS LINK command without the SYNCONRETURN option has been issued on a system that does not support RRMS
	425	A DPL request omitted the SYNCONRETURN option, but specified a value of UOWID.
	601	A GETMAIN of working storage failed. This error leads to user abend 408
	602	A GETMAIN failed. This error leads to user abend 403.
	603	A GETMAIN failed. This error leads to user abend 410
	604	A GETMAIN failed
	605	A GETMAIN for the VERIFY block failed. This error leads to user abend 409.
	606	An SSI verify request (to obtain CICS SVC instruction) failed. This error leads to user abend 405.
	607	An SVC call failed. This error leads to user abend 406.
	608	Logon to IRP failed
	609	Connect to IRP failed
	610	Disconnect from IRP failed
	611	Logoff from IRP failed
	612	Invalid data input to transformer_1
	613	Invalid data input to transformer_4

Table 21. Exceptional conditions (continued). RESP and RESP2 values returned from the EXEC API.

Condition (RESP)	RESP2	Meaning
LINKERR (88)	614	CICS has responded but has not sent any data
	615	CICS cannot satisfy the request
	616	IRP_SWITCH_PULL request (to read data sent from CICS into a larger input/output area) has failed
	617	A GETMAIN for a larger input/output area failed
	619	IRP has had a problem with the input/output area passed from the client program
	620	IRP has disconnected from EXCI
	621	A DISCONNECT command is issued in an error situation following an IRP CONNECT. The DISCONNECT has failed, indicating a serious error.
	622	XCPRH ESTAE setup command failed This error leads to user abend 402.
	623	XCPRH ESTAE invoked due to program check during the processing of this command. ESTAE attempts backout and takes a SYSMDUMP. Further requests are permitted although the pipe is now in a MUST CLOSE state.
	624	The DPL request has been passed to CICS but the time specified in DFHXCOPT has been exceeded. The request is canceled.
	625	An MVS STIMERM macro call failed
	626	An MVS STIMERM CANCEL request failed
	627	The CICS SVC is at the incorrect level. This error leads to user abend 407.
	628	DFHIRP is at the incorrect level.
	629	A response to a DPL request has been returned by CICS but the external CICS interface does not understand the response.
	630	An unexpected return code was received from RRMS when processing an EXEC CICS LINK command without the SYNCONRETURN option .
	631	An unexpected error was encountered when processing an EXEC CICS LINK command without the SYNCONRETURN option.
	632	A GETMAIN for DFHXCGUR's working storage failed while processing an EXEC CICS LINK command without the SYNCONRETURN option .
	903	AN XCEIP ESTAE setup command failed
	904	The server program abended with the abend code in the ABCODE field of the RETCODE area
	905	An XCEIP ESTAE invoked

See "Return codes" on page 145 for details of the various copybooks that contain full details of all response and reason codes, including equated values.

Note: All numeric response and reason code values are shown in decimal.

Translation required for EXEC CICS LINK command

Application programs that use the **EXEC CICS LINK** form of the external CICS interface command must translate their programs before assembly or compilation. You do this using the version of the CICS translator that is appropriate for the language of your client program, specifying the translator option EXCI.

The translator option EXCI is mutually exclusive with the CICS and DLI options. For more information about translating programs that contain EXEC CICS commands, see Translation and compilation in CICS Application Programming.

For information about compiling and link-editing external CICS interface client programs, see Chapter 16, "Compiling and link-editing EXCI client programs," on page 175.

Chapter 11. Setting up EXCI for static routing

You can statically route requests to CICS programs from applications that use the EXCI.

Before you begin

Before you begin, verify that the MVS parameter **Maxmember** is set to a high value. This parameter controls how many connections can be made to the DFHIRP00 resource.

Procedure

1. Add RDO group EXCIXXXX to the grouplist of the CICS region. If EXCIXXXX is not available, make a copy from the supplied DFH$EXCI RDO group. This group contains all connections required for EXCI functions and can support up to 100 connections for batch requests.
2. Add the RDO group for the application to the grouplist of the CICS region.
3. Assemble DFHXCOPT into the SDFHEXCI load library. Ensure that DFHXCOPT has SURROGATE=YES.
4. Assemble and compile your application programs. If your application program is written in Assembler, use the linkage editor parameters AMODE(31) and RMODE(ANY). Link the program into your application load library.
5. Configure the batch JCL to run your application program.
   a. Edit the JCL to specify to which CICS region the batch program will send the EXCI request:

      ```
      //step0010 EXEC PGM=program,PARM='applid,userid'
      ```

      applid is the CICS region and *userid* is a RACF user ID.
   b. Ensure your load library is concatenated as follows:

      ```
      //STELIB DD Disp=shr,Dsn=SYS5C.CICn.CICS670.SDFHEXCI
      //DD Disp=shr,Dsn=Your.application.loadlib
      ```
6. Run the batch program and check that the results are as expected.

Chapter 12. Setting up EXCI for dynamic routing

You can dynamically route requests to CICS programs from applications that use the EXCI using CICSPlex SM.

About this task

Procedure

1. Specify the following system initialization parameters in the CICS region:
   - DSRTPGM=EYU9XLOP
   - DTRPGM=EYU9XLOP
2. Update your RDO group as follows:
   a. Add an RDO entry for the EXCI server program, DFHMIRS. You can use another transaction instead of EXCI if required, but it must point to program DFHMIRS and have a profile of DFHCICSA. Model it after the EXCI transaction in group DFH$EXCI. This transaction will point to DFHMIRS program.
   b. Add the RDO group to the terminal-owning region (TOR) LIST.

   Ensure that the TOR region has program autoinstall disabled.
3. Log into CICSPlex SM and define the transaction for the routing region into these CPSM groups: TXNGRP, WLMDEF, WLMGRP WLMSPEC.
4. For each application-owning region (AOR), create the RDO group in the same way as the sample definition DFH$EXCI. This group must contain only the connections, sessions, and application programs, or equivalent transactions. If you used a different transaction instead of EXCI, you must specify it in this group.

Results

When an application issues a distributed program link, CICSPlex SM checks if the incoming transaction is under its control. When it finds that EXCI or its equivalent is valid in its transaction group, CICSPlex SM routes the transaction to one of the candidate AORs for processing.

An alternative to this approach is to define an RDO definition for the program or transaction with DYNAMIC=Yes. CICSPlex SM routes the request to the selected region.

Chapter 13. Defining connections to CICS

Connections between an EXCI client program and a CICS region require connection definitions in the CICS region. You define these using the CONNECTION and the SESSIONS resource definition facilities provided by CICS.

The following options are provided specifically for the external CICS interface:

- CONNTYPE on the CONNECTION resource definition
- EXCI on the PROTOCOL attribute of the CONNECTION and SESSIONS resource definitions.

This chapter describes:
- "CONNECTION resource definition for EXCI"
- "SESSIONS resource definitions for EXCI connections" on page 162
- "Inquiring on the state of EXCI connections" on page 165

CONNECTION resource definition for EXCI

The EXCI option is provided on the PROTOCOL attribute of the CONNECTION resource definition to indicate that the connection is for use by an MVS program using the external CICS interface.

The CONNTYPE attribute is provided on the CONNECTION resource definition. For EXCI connections, this indicates whether the connection is generic or specific. It is not to be used for any protocol other than the external CICS interface.

The following parameters are relevant to EXCI:

CONNTYPE({SPECIFIC|GENERIC})
> For external CICS interface connections, indicates the nature of the connection.
>
> **SPECIFIC**
>> The connection is for communication from a non-CICS client program to the CICS region, and is specific. A specific connection is an MRO link with one or more sessions dedicated to a single user in a client program.
>>
>> **Note:** A *user* is a program that has issued an Initialize_User request (or for which an Initialize_User request has been issued), with a unique name per TCB. For example:
>> - A simple client program running under MVS can be a single user of the external CICS interface.
>> - A client program running under MVS can open several pipes and issue external CICS interface calls over them sequentially, on behalf of different vendor packages. In this case, from the viewpoint of the client program, each of the packages is a user, identified by a unique user name. Thus a single client program can operate on behalf of multiple users.
>> - A program running under MVS can attach several TCBs, under each of which a vendor package issues external CICS interface calls on its own behalf. Each package is a client program in its own right, and runs under its own TCB. Each is also a user, with a unique user name.
>>
>> For a specific connection, NETNAME is mandatory.

GENERIC

The connection is for communication from a non-CICS client program to the CICS system, and is generic. A generic connection is an MRO link with a number of sessions to be shared by multiple EXCI users. For a generic connection you cannot specify the NETNAME attribute.

Note: You must install only one generic EXCI connection in a CICS region.

NETNAME

For an external CICS interface connection, NETNAME corresponds to the name of the user of a specific pipe, as specified on the *user_name* parameter of an INITIALISE_USER call.

For an external CICS interface specific pipe, you must specify a NETNAME.

For external CICS interface generic pipes, you must leave NETNAME blank.

PROTOCOL({APPC|LU61|EXCI|blank})

The type of protocol that is to be used for the link.

blank

For MRO between CICS regions. You must leave the PROTOCOL blank for MRO, and on the SESSIONS definition you must specify LU6.1 as the PROTOCOL.

APPC (LUTYPE6.2 protocol)

Advanced program-to-program communication, or APPC protocol. This is the default value for ACCESSMETHOD(VTAM). Specify this for CICS-CICS ISC.

Note: VTAM is now the z/OS Communications Server.

LU61

LUTYPE6.1 protocol. Specify this for CICS-CICS ISC or CICS-IMS ISC, but not for MRO.

EXCI

The external CICS interface. Specify this to indicate that this connection is for use by a non-CICS client program using the external CICS interface.

If you specify PROTOCOL(EXCI), you must also specify ACCESSMETHOD(IRC). EXCI is implemented in MRO using the CICS interregion communication program, DFHIRP, and cannot use MRO links that use MVS cross-memory services (XM). EXCI can use also XCF MRO links, which also work through DFHIRP.

SESSIONS resource definitions for EXCI connections

You indicate on the PROTOCOL attribute of the SESSIONS resource definition whether the sessions allocated on the MRO connection are for use by the external CICS interface.

For full details of the SESSIONS resource definition, see SESSIONS resourcesin the *CICS Resource Definition Guide*. The following parameters are relevant to EXCI:

PROTOCOL({APPC|LU61|EXCI})

Indicates the type of protocol that is to be used for an intercommunication link (ISC or MRO).

APPC (LUTYPE6.2)

Advanced program-to-program communication (APPC) protocol. Specify this for CICS-CICS ISC.

LU61

LUTYPE6.1 protocol. Specify this for CICS-CICS ISC, for CICS-IMS, or for MRO.

EXCI

The external CICS interface. Specify this to indicate that the sessions are for use by a non-CICS client program using the external CICS interface. If you specify EXCI, you must leave SENDCOUNT blank.

RECEIVECOUNT({blank|*number*})

The number of MRO, LUTYPE6.1, or EXCI sessions that usually receive before sending.

For MRO, receive sessions can only receive before sending.

blank

These sessions can send only; there are no receive sessions.

number

Specifies the number of receive sessions on connections that specify blank, LU61, or EXCI on the protocol parameter of the CONNECTION definition. CICS uses the number to generate the last two or three characters of the session names (see RECEIVEPFX for details).

If you are using the default receive prefix (<), or your own 1-character prefix, specify a number in the range 1 through 999.

If you specify a 2-character prefix, the number is restricted to the range 1 through 99.

Except for external CICS interface (EXCI) connections, the RECEIVECOUNT in this system should equal SENDCOUNT in the other system.

RECEIVEPFX(<|*prefix*)

Specifies a 1- or 2-character prefix that CICS is to use as the first 1 or 2 characters of the receive session names (the names of the terminal control table terminal entries (TCTTEs) for the sessions).

Prefixes must not cause a conflict with an existing connection or terminal name.

< (MRO and EXCI sessions)

For MRO sessions, if you do not specify your own receive prefix, CICS enforces the default prefix—the less-than symbol (<), which is used in conjunction with the receive count to generate receive session names.

CICS creates the last three characters of the session names from the alphanumeric characters A through Z, and 1 through 9. These 3-character identifiers begin with the letters AAA, and continue in ascending sequence until the number of session entries reaches the limit set by the RECEIVECOUNT value. Note that receive session names are generated *after* the send sessions, and they follow in the same sequence.

For example, if the last session name generated for the send sessions is <AAJ, using the default prefix (<) CICS generates the receive session names as <AAK, <AAL, <AAM, and so on. (This method of generation of session identifiers is the same as for APPC sessions, except for the initial prefix symbol.)

Note: If you specify your own prefix, CICS generates the session names as in earlier releases, which is the same as for LUTYPE6.1 sessions.

prefix **(LUTYPE6.1 sessions)**

If the sessions are on LUTYPE6.1 ISC connections, you must specify a 1- or 2-character prefix. Do not use the default < symbol for LUTYPE6.1 sessions.

For LUTYPE6.1 sessions (and MRO if you specify your own 1- or 2-character prefix) CICS generates session names by appending a number to the prefix, either in the range 1 through 99, or 1 through 999. The number begins with 1 and is incremented by 1 until the specified RECEIVECOUNT is reached.

SENDCOUNT(blank|*number*)

The number of MRO or LUTYPE6.1 sessions that usually send before receiving.

For MRO, send sessions must send before they can receive.

blank

These sessions can receive only; there are no send sessions.

You must leave this field blank when the sessions are on an external CICS interface (EXCI) connection.

number

Specifies the number of send sessions on connections that specify blank or LU61 on the protocol parameter of the CONNECTION definition. CICS uses the number to generate the last two or three characters of the session names (see SENDPFX for details).

If you are using the default send prefix (>), or your own 1-character prefix, specify a number in the range 1 through 999.

If you specify a 2-character prefix, the number is restricted to the range 1 through 99.

Except for external CICS interface (EXCI) connections the SENDCOUNT in the sending system should equal RECEIVECOUNT in the receiving system.

SENDPFX(>|*prefix*)

Specifies a 1- or 2-character prefix that CICS is to use as the first 1 or 2 characters of the send session names (the names of the terminal control table terminal entries (TCTTEs) for the sessions).

Prefixes must not cause a conflict with an existing connection or terminal name.

> (MRO sessions)

For MRO sessions, if you do not specify your own send prefix, CICS enforces the default prefix—the greater-than symbol (>), which is used in conjunction with the send count to generate send session names.

CICS creates the last three characters of the session names from the alphanumeric characters A through Z, and 1 through 9. These 3-character identifiers begin with the letters AAA, and continue in ascending sequence until the number of session entries reaches the limit set by the SENDCOUNT value.

For example, using the default prefix (>) CICS generates session names as >AAA, >AAB, >AAC, and so on. (This method of generation of session identifiers is the same as for APPC sessions, except for the initial symbol.)

> **Note:** If you specify your own prefix, CICS generates the session names as in earlier releases, which is the same as for LUTYPE6.1 sessions.

prefix **(for LUTYPE6.1 sessions)**
> If the sessions are on LUTYPE6.1 ISC connections, you must specify a 1- or 2-character prefix. Do not use the default > symbol for LUTYPE6.1 sessions.
>
> For LUTYPE6.1 sessions (and MRO if you specify your own 1- or 2-character prefix) CICS generates session names by appending a number to the prefix, either in the range 1 through 99, or 1 through 999. The number begins with 1 and is incremented by 1 until the specified SENDCOUNT is reached.

USERID(userid**)**
> The preset user identifier to be used for link security checking.
>
> If you do not specify a preset userid for link security, CICS uses the userid passed from the remote user as the userid for link security. For an external CICS interface link, this is the client userid.

Inquiring on the state of EXCI connections

If you have access, through a CICS terminal, to the CICS server region, you can inquire about batch jobs that are running a client application program, and which are using the external CICS interface to link to a server program in CICS.

About this task

To obtain this information about batch jobs linked to CICS through MRO, you use the CEMT INQUIRE EXCI command. This command enables you to identify the names of external CICS interface batch jobs currently connected to CICS through the interregion communication (IRC) facility.

CICS returns job identifications in the form:

```
jobname.stepname.procname - mvsid
```

Either `stepname`, or `procname`, or both may not be present, indicated by the periods (.) being adjacent to one another.

The `mvsid` identifies the MVS system on which the job is running. If XCF/MRO is in use, the job can reside on a different MVS image from that on which CICS is running.

Information about jobs using the external CICS interface is available only when the job has issued at least one DPL request. A non-zero task number indicates that a DPL request is currently active. A zero task number indicates an external CICS interface session is still open (connected) for that job, although no DPL request is currently active.

See CEMT - master terminal, in the *CICS Supplied Transactions* manual for more information about the CEMT command.

Chapter 14. The EXCI user-replaceable module

This section contains Product-sensitive Programming Interface information.

The external CICS interface provides a user-replaceable module, DFHXCURM. The load module is supplied in CICSTS42.CICS.SDFHEXCI, and the source in CICSTS42.CICS.SDFHSAMP. You can find information about assembling and link-editing user-replaceable programs in the *CICS Customization Guide*.

DFHXCURM is invoked by the external CICS interface in the non-CICS region during the processing of Allocate_Pipe commands, and after the occurrence of any retryable error. The retryable responses are:

- The target CICS region is not available
- There are no pipes available on the target CICS region
- There has been no IRC activity since the MVS IPL.

To retry after a retryable error, issue the EXCI call again.

As supplied, DFHXCURM is effectively a dummy program because of a branch instruction that bypasses the sample logic and returns control to the external CICS interface caller. To use the sample logic, remove the branch instruction and assemble and link-edit the module. Customizing DFHXCURM allows you to do the following:

- When invoked during Allocate_Pipe processing, you can change the specified CICS APPLID, in order to route the request to another CICS system.
- When invoked after a retryable error you can store information regarding CICS availability. You can then use this information on the next invocation of DFHXCURM for Allocate_Pipe processing, so that you can decide to which CICS system to route the request.

DFHXCURM is called using standard MVS register conventions, with register 1 containing the address of the parameter list, and register 14 the return address of the caller. The parameters addressed by register 1 are mapped in the EXCI_URM_PARMS DSECT, which is contained within the DFHXCPLD copybook. The parameters passed to DFHXCURM are as follows:

URMINV

  The address of a fullword that contains the reason for the invocation of DFHXCURM, defined by the following equates:

```
URM_ALLOCATE     EQU 1  This invocation is for an Allocate_Pipe
URM_NO_CICS      EQU 2  The target CICS region is not available
URM_NO_PIPE      EQU 3  There are no pipes available
URM_NO_CICS_IRC  EQU 4  There has been no IRC activity since the MVS IPL
```

URMCICS

  The address of an 8-byte area that contains the generic APPLID of the target CICS system, as specified on the *CICS_applid* parameter of the Allocate_Pipe command, or on the APPLID parameter of the EXEC CICS LINK command.

  When specified by one of these commands, you can change the APPLID to that of a different target CICS region.

  If the *CICS_applid* parameter is omitted from the Allocate_Pipe call, or APPLID is omitted from the EXEC CICS LINK command, the field addressed by this

parameter contains 8 blanks. In this case, you must specify an APPLID in
DFHXCURM before returning control to the caller.

URMAPPL

The address of an 8-byte area that contains the client program's user name as
specified on the *my_name* parameter of the Initialize_User command. Note that
if DFHXCURM is invoked for an EXEC CICS LINK command, this name is
always set to DFHXCEIP.

URMPROG

The address of an 8-byte area that contains the name of the target program (if
available). This name is available only if DFHXCURM is invoked for an EXEC
CICS LINK command. For an external CICS interface Allocate_Pipe command,
the program name is not known until the DPL call is issued.

URMOPTS

The address of a 1-byte area that contains the allocate options, which can be
X'00' or X'80', as specified on the *allocate_opts* parameter. This address is valid
for an Allocate_Pipe request only.

URMANCH

The address of a 4-byte area that is provided for use by DFHXCURM only. A
typical use for this is to store a global anchor address of an area used to save
information across a number of invocations of DFHXCURM. For example, you
can GETMAIN the necessary storage and save the address in the 4-byte area
addressed by this parameter. The initial value of the 4-byte area is set to zero.

There is one URMANCH for each TCB in the address space using EXCI.

Chapter 15. Using the EXCI options table, DFHXCOPT

The EXCI options table, generated by the DFHXCOPT macro, enables you to specify a number of parameters that are required by the external CICS interface.

The format of the DFHXCOPT options table has changed since it was first introduced. In CICS TS for z/OS, Version 3.2 (and, by APAR, also in CICS TS 2.1, CICS TS 2.2, and CICS TS 3.1) the table now includes a version number, which is intended to allow more flexibility for future extensions. You need to be aware of this change if, for example, you plan to migrate a customized DFHXCOPT table from an earlier release of CICS to CICS TS for z/OS, Version 3.2.

To distinguish between the old and newer formats, the new-format table is
link-edited with an alias called "DFHXCOPE". The following sequence is used to
load the options table:
1. CICS tries to load the DFHXCOPT table using its alias name of DFHXCOPE. If
it finds a load module named DFHXCOPE, and successfully loads it, CICS
assumes that the table is in the new format.
2. If CICS does not find a load module named DFHXCOPE (or finds it but fails to
load it), it tries to load the table using its "base" name of DFHXCOPT. In this
case, CICS assumes that the table is in the older format.

CICS provides a default DFHXCOPT table. The source code of the default table, which you can tailor to your own requirements, is supplied in the CICSTS42.CICS.SDFHSAMP library. The load module of the default DFHXCOPT table, with its alias DFHXCOPE, is in the CICSTS42.CICS.SDFHEXCI library.

If you create your own, customized, DFHXCOPT table, ensure that you link-edit it using the DFHXCOPE alias. (Using the standard DFHAUPLE procedure, described below, ensures that this happens.) If you reassemble and link-edit your table without the alias, CICS will load the default table (found by means of its DFHXCOPE alias), rather than your customized table.

You must assemble and link-edit your customized DFHXCOPT table into a suitable library in the STEPLIB concatenation of the job that runs the MVS client program. You can use your own version of the CICS DFHAUPLE procedure to do this. The DFHAUPLE procedure is supplied in CICSTS42.CICS.SDFHINST.

Unlike the tables you specify for CICS regions, the DFHXCOPT table cannot be suffixed.

Table 22 shows the format of the DFHXCOPT macro and its parameters.

Table 22. The DFHXCOPT macro parameters

	DFHXCO	TYPE={**CSECT** | DSECT}
		[,ABENDBKOUT={**NO** | YES}]
		[,CICSSVC={**0** | *number*}]
		[,CONFDATA={**SHOW** | HIDETC}]
		[,DURETRY={**30** | *number-of-seconds*}]
		[,GTF={**OFF** | ON}]
		[,MSGCASE={**MIXED** | UPPER}]
		[,SURROGCHK={**YES** | NO}]

Table 22. The DFHXCOPT macro parameters (continued)

		[,TIMEOUT={**0**|*number*}] [,TRACE={**OFF**|1|2}] [,TRACESZE={**16**|*number-of-kilobytes*}] [,TRAP={**OFF**|ON}] [,XCFGROUP={**DFHIR000**|name}] You must terminate your parameters with the following END statement.
	END	DFHXCOPT

TYPE={CSECT|DSECT}
> Indicates the type of table to be generated.
>
> **CSECT**
>> A regular control section that is normally used.
>
> **DSECT**
>> A dummy control section.

ABENDBKOUT={NO|YES}
> Specifies whether a task that abends within the CICS server is to trigger an automatic rollback of the global unit of work. A global unit of work exists when an EXCI client program is controlling resource recovery through MVS RRS (that is, SYNCONRETURN is *not* specified on the DPL request). In this case you may well want the global unit of work to be marked for rollback if the CICS server program abends.
>
> **Note:** ABENDBKOUT has no effect when SYNCONRETURN is specified on the DPL request.
>
> **NO** The global unit of work is not marked for rollback.
>
> **YES**
>> When processing the abend of the server program, the CICS mirror program marks the global unit of work for backout.
>
> In both cases the EXCI client program receives a return code of 422, SERVER_ABENDED, on the EXCI DPL request.

CICSSVC={0|*number*}
> Specifies the CICS type 3 SVC number being used for MRO communication.
>
> The external CICS interface must use the same SVC number that is in use by the CICS MRO regions that reside in the MVS image in which the client program is running.
>
> If you do not specify a specific CICS SVC number, the external CICS interface determines the SVC in use for MRO by means of an MVS VERIFY command.
>
> **0** Specify zero to indicate that the external CICS interface is to obtain the CICS SVC number from MVS. This is the default.
>
>> You should only specify 0 when you are sure that at least one CICS region has logged on to DFHIRP during the life of the MVS IPL.
>
> *number*
>> Specify the CICS SVC number, in the range 200—255, that is in use for

CICS interregion communications. This must be the SVC number that is installed in the MVS image in which the client program is running (the local MVS).

If no MRO CICS regions have ever logged on to DFHIRP in the local MVS during the life of the IPL, you must specify the SVC number. If you allow this parameter to default, and the external CICS interface requests the SVC from MVS, the request will fail if no CICS region has logged on to DFHIRP.

This parameter is required in those MVS images that do not run any CICS regions, and the client program is issuing DPL requests to a server CICS region that resides in another MVS. In these circumstances, the client program logs on to the local DFHIRP using the locally defined SVC, and communicates with the remote CICS region using XCF/MRO.

Note: All CICS regions using MRO within the same MVS image must use the highest level of both DFHIRP and the CICS SVC, DFHCSVC. If your MRO CICSplex consists of CICS regions at different release levels, the DFHIRP and DFHCSVC installed in the LPA must be from highest release level of CICS within the CICSplex.

MVS client programs using the external CICS interface can communicate only with server regions running under CICS for MVS/ESA 4.1 or later.

CONFDATA={SHOW|HIDETC}

Code this parameter to indicate whether the external CICS interface is to suppress (hide) user data that might otherwise appear in EXCI trace entries output to GTF or in EXCI dumps. This option applies to the tracing of the COMMAREA flowing between the EXCI client program and the CICS server program.

SHOW

Data suppression is not in effect. User data is traced.

HIDETC

This specifies that you want EXCI to 'hide' user COMMAREA data from trace entries. Instead of the COMMAREA data, the trace entry contains a character string stating that the data has been suppressed.

DURETRY={30|*number-of-seconds*|0}

Specifies the total time, in seconds, that the external CICS interface is to continue trying to obtain an MVS system dump using the SDUMP macro.

DURETRY allows you to control whether, and for how long, the external CICS interface is to reissue the SDUMP if another address space in the same MVS system is already taking an SDUMP when the external CICS interface issues an SDUMP request.

In the event of an SDUMP failure, the external CICS interface reacts as follows:

* If MVS is already taking an SDUMP for another address space, and the DURETRY parameter is nonzero, the external CICS interface issues an MVS STIMERM macro to wait for five seconds, before retrying the SDUMP macro. The external CICS interface issues a message to say that it will retry the SDUMP every five seconds until the DURETRY time limit.
* If the SDUMP fails for any other reason such as:
  – There are no SYS1.DUMP data sets available, or
  – There are I/O errors preventing completion of the dump, or
  – The DURETRY limit expires while retrying SDUMP

the external CICS interface issues a message to inform you that the SDUMP has failed, giving the reason why.

30 30 seconds allows the external CICS interface to retry up to six times (once every five seconds).

number-of-seconds
Code the total number of seconds (up to 32767 seconds) during which you want the external CICS interface to continue retrying the SDUMP macro. The external CICS interface retries the SDUMP, once every five seconds, until successful or until retries have been made over a period equal to or greater than the DURETRY value.

0 Code a zero value if you do not want CICS to retry the SDUMP.

GTF={OFF|ON}
Specifies whether all trace entries normally written to the external CICS interface trace table are also to be written to an MVS generalized trace facility (GTF) data set (if GTF tracing is active).

OFF
Code this if trace entries are not to be written to GTF.

ON Code this if trace entries are to be written to GTF.

MSGCASE={MIXED|UPPER}
Specifies whether the DFHEX*xxxx* messages are to be issued in mixed or uppercase.

MIXED
Code this if messages are to be issued in mixed case.

UPPER
Code this if messages are to be issued in uppercase.

SURROGCHK={YES|NO}
Specifies whether the external CICS interface is to perform surrogate user checks against the client job user id (the user ID under which the EXCI job is running).

YES
The external CICS interface is to perform a surrogate user check to verify that the user ID under which the EXCI client job is running is authorized as a surrogate for the user ID specified on a DPL call. The check is made only when the client user ID is different from the user ID on the DPL call.

The client user ID must be authorized to the appropriate profile in the SURROGAT general resource class. You do this by giving the client user ID READ authority to a profile named *userid*.DFHEXCI in the SURROGAT general resource class, where *userid* is the user ID specified on the DPL call.

NO Surrogate user checks are not to be performed.

TIMEOUT={0|*number*}
Specifies the time interval, in hundredths of a second, during which the external CICS interface waits for a DPL command to complete.

0 Specifies that you do not want any time limit applied, and that the external CICS interface is to wait indefinitely for a DPL command to complete.

number
Specifies the time interval, in hundredths of a second, that the external CICS interface is to wait for a DPL command to complete. The number represents hundredths of a second, from 1 up to a maximum of 2 147 483 647. For example:

6000　　Represents a timeout value of one minute

30000　　Represents a timeout value of five minutes

60000　　Represents a timeout value of ten minutes.

TRACE={OFF|1|2}
Specifies whether you want internal tracing for the external CICS interface, and at what level.

OFF
Internal tracing for the external CICS interface is not required. However, even with normal tracing switched off, exception trace entries are always written to the external CICS interface trace table in the CICS region.

1　　Exception and level-1 trace entries are written to the external CICS interface trace table.

2　　Exception, level-1, and level-2 trace entries are written to the external CICS interface trace table.

TRACESZE={16|*number-of-kilobytes*}
Specifies the size in kilobytes of the trace table that is used by the external CICS interface. This trace table is allocated in 31-bit storage (above the line) in the CICS region. .

16　16KB is the default size of the trace table, and also the minimum size.

number-of-kilobytes
The number of kilobytes of storage to be allocated for the trace table, in the range 16 KB through 1 048 576 KB. Subpool 1 is used for the trace table storage, which exists for the duration of the job step TCB. The table is page-aligned and occupies a whole number of pages. If the value specified is not a multiple of the page size (4 KB), it is rounded up to the next multiple of 4 KB.

TRAP={OFF|ON}
Specifies whether the service trap module, DFHXCTRA, is to be used. DFHXCTRA is supplied as a user-replaceable module, in which IBM service personnel can add code to trap errors.

OFF
Code this if you do not want to use DFHXCTRA.

ON　Code this if you require DFHXCTRA.

XCFGROUP={DFHIR000|name}
Specifies the name of the cross-system coupling facility (XCF) group to be joined by this client program.

Note: XCF groups allow CICS regions in different MVS images within the same sysplex to communicate with each other across multi-region operation (MRO) connections. For introductory information about XCF/MRO, and instructions on how to set up XCF groups, see Cross-system multiregion operation (XCF/MRO) in the *CICS Intercommunication Guide*.

Each client program can join a maximum of one XCF group.

DFHIR000
> The default XCF group name.

name
> The group name must be eight characters long, padded on the right with blanks if necessary. The valid characters are A-Z 0-9 and the national characters $ # and @. To avoid using the names IBM uses for its XCF groups, do not begin group names with the letters A through C, E through I, or the character string "SYS". Also, do not use the name "UNDESIG", which is reserved for use by the system programmer in your installation.
>
> It is recommended that you use a group name beginning with the letters "DFHIR".

Chapter 16. Compiling and link-editing EXCI client programs

All programs that use the external CICS interface to pass DPL requests to a CICS server region must include the CICS-supplied program stub, DFHXCSTB.

The stub intercepts all external CICS interface commands, whether they are EXCI CALL interface commands, or EXEC CICS LINK commands, and ensures they are passed to the appropriate external CICS interface routine for processing.

DFHXCSTB is a common stub, designed for inclusion in programs written in all the supported languages. It is supplied in the CICSTS42.CICS.SDFHEXCI library.

Note: The CICSTS42.CICS.SDFHEXCI also contains entries for DFHXCIE and DFHXCIS, which are aliases for DFHXCSTB.

To help you ensure that the stub is included, CICS provides a number of procedures, one for each language, which you can use for translating, compiling, and link-editing.

You must specify AMODE(31) for your EXCI client program.

The CICS-supplied procedures for compiling and link-editing client programs include the following parameters on the PARM statement of the linkage editor job step:

```
LNKPARM='AMODE(31),LIST,XREF'
```

The rest of this chapter describes:
- "Job control language to run an EXCI client program"
- "EXCI programming considerations" on page 177
- "Using EXCI sample application programs" on page 178

Job control language to run an EXCI client program

An EXCI client program runs in an MVS address space, for example, as a batch job.

Note the following requirements when writing the JCL for your client program:
- Include in the STEPLIB concatenation those libraries that contain the CICS-supplied external CICS interface modules and also the client program. The external CICS interface modules are supplied in CICSTS42.CICS.SDFHEXCI, which contains the following:

  DFH$AXCC
  DFHMEBMX
  DFHMET4E
  DFHXCEIX
  DFHXCIE   (alias of DFHXCSTB)
  DFHXCIS   (alias of DFHXCTSB)
  DFHXCOPT
  DFHXCPRX

```
                        DFHXCSTB
                        DFHXCTRA
                        DFHXCURM
```

- You are recommended to include a DD statement for SYSMDUMP. The external CICS interface uses SYSMDUMP for some error conditions.

- The REGION parameter must specify a large enough region size to allow for the size of the internal trace table specified by the TRACESZE parameter in the DFHXCOPT options table.

- Include a SYSPRINT or equivalent DD statement for any output from the client program.

Figure 22 shows a sample job that you can use or modify to start a client program.

```
//EXCI    JOB (accounting_information),CLASS=A,TIME=1440,
//         USER=userid,PASSWORD=pswd,REGION=100M
//*===============================================================*
//*    JCL to execute an external CICS interface client program   *
//*===============================================================*
//          EXEC  PGM=pgmname
//STEPLIB  DD    DSN=CICSTS42.CICS.EXCI.LOADLIB,DISP=SHR
//         DD    DSN=CICSTS42.CICS.SDFHEXCI,DISP=SHR
//SYSPRINT DD    SYSOUT=A
//SYSDUMP  DD    DSN=SYS1.SYSMDP00,VOL=SER=volid,SPACE=(CYL,(1,1)),
                 DISP=OLD,UNIT=3390
```

Figure 22. Sample job for starting an EXCI client program

Note:

1. The job user ID, specified on the USER parameter, must be defined to RACF , or an equivalent external security manager (ESM).

2. In addition to being used for job step initiation security, the job user ID is also used for MRO logon and bind-time security checking.

   See Chapter 17, "EXCI security," on page 187 for information about security when using the external CICS interface.

3. See "Installing the EXCI sample definitions" on page 183 for information about modifying the sample connection definitions before you run the sample application programs in an environment that does not have RACF, or an equivalent external security manager (ESM), installed and active.

CICS-supplied procedures for the EXCI

CICS provides seven procedures to enable you to translate, compile, and link-edit your client programs.

Four of these are for use with specific language compilers or assembler, the other three being for use with Language Environment. These procedures, with the four language-specific procedures shown first, are:

DFHEXTAL
> The assembler procedure for assembler versions of client programs

DFHYXTEL
> The procedure for C++ versions of client programs running under Language Environment.

DFHYXTDL
> The procedure for C versions of client programs running under Language Environment.

DFHYXTPL

The procedure for PL/I versions of client programs running under Language Environment

DFHYXTVL

The procedure for COBOL versions of client programs running under Language Environment.

To ensure that the EXCI stub is included with your client program, all these procedures include a step, COPYLINK, that unloads the stub into a temporary data set defined with a block length suitable for the linkage-editor. This temporary data set is then concatenated with the temporary data set containing your object program on the SYSLIN DD statement in the LKED step.

These procedures are supplied in the CICSTS42.CICS.SDFHPROC library. You are recommended to copy these to SYS1.PROCLIB or another suitable procedure library.

EXCI programming considerations

There are some language requirements that apply to writing an MVS client program that uses the external CICS interface. These affect programs written in PL/I and C. Also, for all languages, consider how you handle return codes before terminating your MVS client program.

PL/I considerations

PL/I programs written to the external CICS interface must provide their parameters on the CALL to DFHXCIS in the form of an assembler-style parameter list.

The EXCI copybook for PL/I, DFHXCPLL, contains the necessary definition of the DFHXCIS entry point, as follows:

```
 DCL DFHXCIS  ENTRY        OPTIONS(INTER ASSEMBLER);
```

The same rule applies for the EXCI LINK command, and in this case the CICS translator ensures that the correct parameter list is built.

For an example of an EXCI client program written in PL/I, see the source of the sample program, DFH$PXCC.

C considerations

C programs written to the external CICS interface must provide their parameters on the CALL to DFHXCIS in the form of an assembler-style parameter list. You ensure this by declaring the entry point to DFHXCIS with OS LINKAGE.

The EXCI copybook for PL/I, DFHXCPLH, contains the necessary definition of the DFHXCIS entry point, as follows:

```
  #pragma linkage(dfhxcis,OS)
```

The same rule applies for the EXCI LINK command, and in this case the CICS translator ensures that the correct parameter list is built.

For an example of an EXCI client program written in C, see the source of the sample program, DFH$DXCC.

Setting the return code (R15) at termination

The external CICS interface does not clear register 15 at termination, regardless of whether your client program executes normally or not. Therefore, even if your MVS client program terminates normally after successfully using the external CICS interface, the job step could end with an undefined return code.

To ensure a meaningful return code is given at termination, set the job step return code before terminating your program. The sample client programs illustrate how you can do this, using the saved response code from last call to the external CICS interface. For example, the COBOL sample DFH0CXCC program moves SAVED-RESPONSE to special register RETURN-CODE before terminating.

Using EXCI sample application programs

CICS provides a number of sample programs that are designed to help you in writing your own application programs. To help with writing programs that use the external CICS interface, CICS provides sample MVS client programs and a sample CICS server program.

The samples show you how to code client applications that use both the EXCI CALL interface and EXEC CICS LINK command.

```
*==================== TEXCI Sample Batch Client Program =====================*
*
*   EXEC Level Processor.
*      Setting up the EXEC level call.
*      The Link Request has successfully completed.
*      Server Response:
*         The file is set to a browsable state.
*
*   CALL Level Processor.
*      Initialise_User call complete.
*      Allocate_Pipe call complete.
*      Open_Pipe call complete.
*      The connection has been successful.
*         The changed target file follows:
*
*========================= Top of File ===================================*
000100W. DAVIS          SURREY, ENGLAND      3215677826 11 81$0100.11COMMITTED
000102F. ALDSON         WARWICK, ENGLAND     9835618326 11 81$1111.11COMMITTED
000104S. BOWLER         LONDON,ENGLAND       1284629326 11 81$0999.99COMMITTED
000106B. ADAMS          CROYDON, ENGLAND     1948567326 11 81$0087.71COMMITTED
000111GENE BARLOWE      SARATOGA,CALIFORNIA 4612075301 02 74$0111.11COMMITTED
000762GEORGE BURROW     SAN JOSE,CALIFORNIA 2231212101 06 74$0000.00COMMITTED
000983H. L. L. CALL     WASHINGTON, DC       3451212021 04 75$9999.99COMMITTED
 003210B.CREPIN         NICE, FRANCE         1234567026 11 81$3349.99COMMITTED
003214HUBERT C HERBERT  SUNNYVALE, CAL.      3411212000 06 73$0009.99COMMITTED
003890PHILIPPE SMITH, JR NICE, FRANCE        0000000028 05 74$0009.99COMMITTED
004004STAN SMITH        DUBLIN, IRELAND      7111212102 11 73$1259.99COMMITTED
004445S. GALSON         SOUTH BEND, S.DAK.   6121212026 11 81$0009.99COMMITTED
004878D.C. CURRENT      SUNNYVALE, CALIF.    3221212010 06 73$5399.99COMMITTED
005005J. S. LAVERENCE   SAN FRANCISCO, CA.   0000000101 08 73$0009.99COMMITTED
005444JEAN LAWRENCE     SARATOGA, CALIF.     6771212020 10 74$0809.99COMMITTED
005581JOHN ALDEN III    BOSTON, MASS.        4131212011 04 74$0259.99COMMITTED
006016DR W. T. KAR      NEW DELHI, INDIA     7033121121 05 74$0009.88COMMITTED
006670WILLIAM KAPP      NEW YORK, N.Y.       2121212031 01 75$3509.88COMMITTED
006968D. CONRAD         WARWICK, ENGLAND     5671382126 11 81$0009.88COMMITTED
007248B. C. WILLIAMSON  REDWOOD CITY, CALF.  3331212111 10 75$0009.88COMMITTED
007779MRS. W. WELCH     SAN JOSE, CALIF.     4151212003 01 75$0009.88COMMITTED
100000G. NEADS          TORONTO, ONTARIO     0341512126 11 81$0010.00COMMITTED
111111C. MEARS          OTTAWA, ONTARIO      5121200326 11 81$0011.00COMMITTED
200000A. BONFIELD       GLASGOW,  SCOTLAND   6373829026 11 81$0020.00COMMITTED
300000K. TRENCHARD      NEW YORK, U.S.       6473980126 11 81$0030.00COMMITTED
333333D. MYRING         CARDIFF, WALES       7849302026 11 81$0033.00COMMITTED
400000W. TANNER         MILAN, ITALY         2536373826 11 81$0040.00COMMITTED
444444A. FISHER         CALGARY, ALBERTA     7788982026 11 81$0044.00COMMITTED
500000J. DENFORD        MADRID, SPAIN        4445464026 11 81$0000.00COMMITTED
555555C. JARDINE        KINGSTON, N.Y.       3994442026 11 81$0005.00BACKEDOUT
600000F. HUGHES         DUBLIN, IRELAND      1239878026 11 81$0010.00BACKEDOUT
666666A. BROOKMAN       LA HULPE, BRUSSELS   4298384026 11 81$0016.00BACKEDOUT
700000A. MACALLA        DALLAS, TEXAS        5798432026 11 81$0002.00BACKEDOUT
777777D. PRYKE          WILLIAMSBURG, VIRG.  9187613126 11 81$0027.00BACKEDOUT
800000H. BRISTOW        WESTEND, LONDON      2423338926 11 81$0030.00BACKEDOUT
888888B. HOWARD         NORTHAMPTON, ENG.    2369163926 11 81$0038.00BACKEDOUT
900000D. WOODSON        TAMPA, FLA.          3566812026 11 81$0040.00BACKEDOUT
*========================= End of File ===================================*
*
*      Closing Dpl Request has been attempted.
*      Close_Pipe call complete.
*      Deallocate_Pipe call complete.
*
*=================== End of TEXCI Sample Batch Client Program =================*
```

Figure 23. Output from DFH$ATXC

```
*==================== EXCI Sample Batch Client Program =====================*
*
*  EXEC Level Processor.
*    Setting up the EXEC level call.
*    The Link Request has successfully completed.
*    Server Response:
*      The file is set to a browsable state.
*
*  CALL Level Processor.
*    Initialise_User call complete.
*    Allocate_Pipe call complete.
*    Open_Pipe call complete.
*    The connection has been successful.
*      The target file follows:
*
*========================== Top of File ====================================*
000100W. DAVIS          SURREY, ENGLAND    3215677826 11 81$0100.11COMMITTED
000102F. ALDSON         WARWICK, ENGLAND   9835618326 11 81$1111.11COMMITTED
000104S. BOWLER         LONDON,ENGLAND     1284629326 11 81$0999.99COMMITTED
000106B. ADAMS          CROYDON, ENGLAND   1948567326 11 81$0087.71COMMITTED
000111GENE BARLOWE      SARATOGA,CALIFORNIA 4612075301 02 74$0111.11COMMITTED
000762GEORGE BURROW     SAN JOSE,CALIFORNIA 2231212101 06 74$0000.00COMMITTED
000983H. L. L. CALL     WASHINGTON, DC     3451212021 04 75$9999.99COMMITTED
003210B.CREPIN          NICE, FRANCE       1234567026 11 81$3349.99COMMITTED
003214HUBERT C HERBERT  SUNNYVALE, CAL.    3411212000 06 73$0009.99COMMITTED
003890PHILIPPE SMITH, JR NICE, FRANCE      0000000028 05 74$0009.99COMMITTED
004004STAN SMITH        DUBLIN, IRELAND    7111212102 11 73$1259.99COMMITTED
004445S. GALSON         SOUTH BEND, S.DAK. 6121212026 11 81$0009.99COMMITTED
004878D.C. CURRENT      SUNNYVALE, CALIF.  3221212010 06 73$5399.99COMMITTED
005005J. S. LAVERENCE   SAN FRANCISCO, CA. 0000000101 08 73$0009.99COMMITTED
005444JEAN LAWRENCE     SARATOGA, CALIF.   6771212020 10 74$0809.99COMMITTED
05581JOHN ALDEN III     BOSTON, MASS.      4131212011 04 74$0259.99COMMITTED
06016DR W. T. KAR       NEW DELHI, INDIA   7033121121 05 74$0009.88COMMITTED
006670WILLIAM KAPP      NEW YORK, N.Y.     2121212031 01 75$3509.88COMMITTED
006968D. CONRAD         WARWICK, ENGLAND   5671382126 11 81$0009.88COMMITTED
007248B. C. WILLIAMSON  REDWOOD CITY, CALF. 3331212111 10 75$0009.88COMMITTED
007779MRS. W. WELCH     SAN JOSE, CALIF.   4151212003 01 75$0009.88COMMITTED
100000G. NEADS          TORONTO, ONTARIO   0341512126 11 81$0010.00COMMITTED
111111C. MEARS          OTTAWA, ONTARIO    5121200326 11 81$0011.00COMMITTED
200000A. BONFIELD       GLASGOW,  SCOTLAND 6373829026 11 81$0020.00COMMITTED
300000K. TRENCHARD      NEW YORK, U.S.     6473980126 11 81$0030.00COMMITTED
333333D. MYRING         CARDIFF, WALES     7849302026 11 81$0033.00COMMITTED
400000W. TANNER         MILAN, ITALY       2536373826 11 81$0040.00COMMITTED
444444A. FISHER         CALGARY, ALBERTA   7788982026 11 81$0044.00COMMITTED
500000J. DENFORD        MADRID, SPAIN      4445464026 11 81$0000.00COMMITTED
555555C. JARDINE        KINGSTON, N.Y.     3994442026 11 81$0005.00Y00010500
600000F. HUGHES         DUBLIN, IRELAND    1239878026 11 81$0010.00Y00010600
666666A. BROOKMAN       LA HULPE, BRUSSELS 4298384026 11 81$0016.00Y00010700
700000A. MACALLA        DALLAS, TEXAS      5798432026 11 81$0002.00Y00010800
777777D. PRYKE          WILLIAMSBURG, VIRG. 9187613126 11 81$0027.00Y00010900
800000H. BRISTOW        WESTEND, LONDON    2423338926 11 81$0030.00Y00011000
888888B. HOWARD         NORTHAMPTON, ENG.  2369163926 11 81$0038.00Y00011100
900000D. WOODSON        TAMPA, FLA.        3566812026 11 81$0040.00Y00011200
*========================== End of File ====================================*
*
*    Closing Dpl Request has been attempted.
*    Close_Pipe call complete.
*    Deallocate_Pipe call complete.
*
*=================== End of EXCI Sample Batch Client Program =================*
```

Figure 24. Successful execution of DFH$AXCC after DFH$ATXC has been successfully executed.

Description of the sample applications

The sample external CICS interface programs are included on the CICS Transaction Server for z/OS distribution tape.

Two sample MVS client programs are supplied. One is provided in assembler language, COBOL, C, and PL/I. The other is only provided in assembler. The sample CICS server program is provided in assembler only. Assembler language programs are in source and executable form. COBOL, PL/I, and C programs are provided in source form only. Each version of the client program has basically the same function, but programming methods vary somewhat according to the language used.

The sample programs, shown in Table 23, are supplied in source form in CICSTS42.CICS.SDFHSAMP. The sample assembler server program is also supplied in executable form in CICSTS42.CICS.SDFHLOAD. The assembler client program is supplied in CICSTS42.CICS.SDFHEXCI.

Note:
1. The assembler versions of the client program use BSAM, which requires the programs to be link-edited in RMODE(24), as a switch to AMODE(24) is made around the BSAM call. The assembler source code includes the required RMODE(24) statement. Normally, EXCI client programs run AMODE(31),RMODE(ANY).
2. Because of this, the assembler versions of these client programs are unsuitable for use as Language Environment MAIN programs.
3. SDFHEXCI and SDFHDLL1 are downwardly compatible with all supported releases of CICS TS.

Table 23. The external CICS interface sample programs

Language	Name	Type of program
Assembler	DFH$AXCC	Client program
Assembler	DFH$ATXC	Client program
Assembler	DFH$AXCS	Server program
COBOL	DFH0CXCC	Client program
PL/I	DFH$PXCC	Client program
C	DFH$DXCC	Client program

The sample client programs show you how to code a simple MVS client application using the EXCI CALL interface and the EXEC CICS LINK command.

Each version of the client is divided into three separate sections as follows:
1. The first section issues a single **EXEC CICS LINK** command to inquire on the state of the sample VSAM file, FILEA, in the target CICS system.

   If the file is in a suitable state, processing continues to sections two and three, which together provide complete examples of the use of the EXCI CALL interface.
2. The second section initiates a specific MRO connection to the target CICS system and, once the pipe is open, performs a series of calls that each retrieve a single sequential record from the sample VSAM file, until no more records are available.

3. The third section is a simple routine to close the target sample file once processing of the data is complete. It also terminates the MRO connection now that the link is no longer required.

Some of the parameters used on the **EXCI CALL** and **EXEC CICS LINK** commands in the client program need to be tailored for your own target CICS server region. Change these as required, then retranslate, compile (or assemble), and link-edit the program.

The variables and their values specified in the sample programs are given in Table 24.

Table 24. Parameters used in the sample client programs

Variable name in sample program	Default value
TARGET_FILE	FILEA
TARGET_TRANSID	EXCI
TARGET_SYSTEM	DBDCCICS (applid)
TARGET_PROGRAM	DFH$AXCS
TARGET_USERID	Defaults to batch region's user ID
APPLICATION	BATCHCLI

The assembler versions of the client programs are supplied pregenerated in an executable form. All versions of the program accept two run-time parameters, as follows:

1. The first (TARGET_SYSTEM) specifies the server region APPLID.

   For the pregenerated assembler versions this avoids you having to reassemble the programs to specify the applid of your own CICS server region. You can also use the sample client programs with different CICS regions without needing to modify the programs each time.

2. The second specifies the user ID to be used on the call interface DPL_request.

You specify these positional parameters on the PARM statement, separated by a comma.

Using the COMMAREA in the sample programs

Data is passed between the sample client and server programs using a standard CICS communications area (COMMAREA) for passing data between programs. The definitions of this COMMAREA are identical on each side of the EXCI link to ensure that data is mapped correctly.

The sample client program DFH$AXCC minimizes data transmission by specifying a specific data length to avoid sending the whole COMMAREA over the link. The data length option specifies a binary value that is the length of a contiguous area of storage, from the start of the COMMAREA, to be passed to the server program. This is because the amount of data being passed to the server is small, but the COMMAREA itself is much larger so that the server can return the requested records from FILEA. Specifying a data length when the amount of data being passed is smaller than the COMMAREA improves performance.

Sample program DFH$ATXC passes the full COMMAREA to the server program because it makes changes to the file records and the whole of the changed record needs to be passed to the server.

The first and third sections of the sample client programs define the COMMAREA as only 18 bytes (no data is requested from the server in these sections). For the second section, the sample client program defines the COMMAREA as 98 bytes, the extra 80 bytes being required for the sample server program to return a record from FILEA. In all sections, the data length is defined as 18 bytes. The COMMAREA structure is defined in the sample programs as follows:

Bytes	Data type	Field description
0-3	Fullword	Call type code.
4-11	Char(8)	Target file name.
12-17	Char(6)	Ridfield identifier.
18-97	Char(80)	FILEA record data area.

Note that, although the COMMAREA structure is described in both the client and server programs, the actual size of the COMMAREA made available by CICS to the server is determined by the client program. If you modify the sample programs to work with one of your own application programs, make sure you specify a COMMAREA large enough to handle the maximum amount of data the server is expected to return to the client. The server must not attempt to return data that is larger than the COMMAREA specified by the client.

For more information about using a COMMAREA for passing data between CICS programs, see Sharing data across transactions, in the *CICS Application Programming Guide*.

Installing the EXCI sample definitions

Resource definitions that support the EXCI sample programs are included in the CICS system definition file (CSD) in groups DFH$EXCI and DFH$FILA.

Note that the sample definitions, while included in the CSD, are not included in the IBM-defined group list DFHLIST. Thus, if CICS is initialized with GRPLIST=DFHLIST, you must install the EXCI resource definition groups before using the samples. Alternatively, you can add the sample groups to your startup group list, so that they are installed automatically at system initialization.

Transactions that are to be linked to from the batch program need to specify the mirror program (DFHMIRS) as the program name in their transaction definitions.

The resource definition groups that must be installed are as follows:

DFH$EXCI

This contains definitions for the sample server transaction, server program, EXCI connections, and sessions.

Only one server program is included—in assembler language, called DFH$AXCS.

The sample application is designed to run the transaction EXCI, which is defined to invoke the DFHMIRS mirror program and references profile DFHCICSA. The required transaction definition for EXCI is included in the group.

Sample CONNECTION and SESSIONS definitions for specific and generic connections are included.

Note: Both the generic and specific connection definitions supplied in the sample group DFH$EXCI specify ATTACHSEC(IDENTIFY). This security option causes the server program DFH$EXCS to fail with an ATCY abend if you run the sample programs in an environment that does not have RACF, or an equivalent external security manager (ESM), installed and active.

If you want to run the external CICS interface sample programs without any security active, you must alter the connection resource definitions to specify ATTACHSEC(LOCAL).

DFH$FILA

This contains the definition for the supplied sample VSAM file, FILEA, which is referenced by the EXCI sample programs.

Once these are installed, you must ensure that interregion communication (IRC) is open. If IRC is not opened during CICS initialization, set it open using the CEMT SET IRC OPEN command.

Running the EXCI sample applications

If you want to use the COBOL, PL/I, or C version of the EXCI client program, you must translate, compile, and link-edit the program into a suitable library.

You can use the sample JCL shown in Figure 22 on page 176 as a basis for creating your own batch job to run the client program.

If you use the pregenerated assembler version, specify the APPLID of your target CICS server region as a parameter on the EXEC statement for the client program, as follows:

```
//*===============================================================*
//ASM       EXEC  PGM=DFH$AXCC,PARM='applid,userid'
```

Where: *applid* is the applid of the CICS server region *userid* is the userid for the DPL_request call. Note: If you omit *applid*, you must keep the comma preceding the userid.

Change PGM=DFH$AXCC to PGM=DFH$ATXC to run the other client sample program.

Results of running the EXCI sample applications

This is an example of the output produced by successful execution of the pregenerated assembler version of the client program, DFH$AXCC.

```
*==================== EXCI Sample Client Program ===========================*
*                                                                          *
*  EXEC Level Processor.                                                   *
*    Setting up the EXEC level call.                                       *
*    The Link Request has successfully completed.                          *
*    Server Response:                                                      *
*      The file is set to a browsable state.                               *
*                                                                          *
*  CALL Level Processor.                                                   *
*    Initialize_User call complete.                                        *
*    Allocate_Pipe call complete.                                          *
*    Open_Pipe call complete.                                              *
*    The connection has been successful.                                   *
*      The target file follows:                                            *
*                                                                          *
*========================== Top of File ====================================*
000102F. ALDSON          WARWICK, ENGLAND    9835618326 11 81$1111.11Y00007300
000104S. BOWLER          LONDON,ENGLAND      1284629326 11 81$0999.99Y00007400
000106B. ADAMS           CROYDON, ENGLAND    1948567326 11 81$0087.71Y00007500
000111GENE BARLOWE       SARATOGA,CALIFORNIA 4612075301 02 74$0111.11Y00007600
000762GEORGE BURROW      SAN JOSE,CALIFORNIA 2231212101 06 74$0000.00Y00007700
000983H. L. L. CALL      WASHINGTON, DC      3451212021 04 75$9999.99Y00007800
003210B.CREPIN           NICE, FRANCE        1234567026 11 81$3349.99Y00008100
003214HUBERT C HERBERT   SUNNYVALE, CAL.     3411212000 06 73$0009.99N00008200
003890PHILIPPE SMITH, JR NICE, FRANCE        0000000028 05 74$0009.99N00008300
004004STAN SMITH         DUBLIN, IRELAND     7111212102 11 73$1259.99N00008400
004445S. GALSON          SOUTH BEND, S.DAK.  6121212026 11 81$0009.99N00008500
004878D.C. CURRENT       SUNNYVALE, CALIF.   3221212010 06 73$5399.99N00008600
005005J. S. LAVERENCE    SAN FRANCISCO, CA.  0000000101 08 73$0009.99N00008700
005444JEAN LAWRENCE      SARATOGA, CALIF.    6771212020 10 74$0809.99N00008800
005581JOHN ALDEN III     BOSTON, MASS.       4131212011 04 74$0259.99N00008900
006016DR W. T. KAR       NEW DELHI, INDIA    7033121121 05 74$0009.88Y00009000
006670WILLIAM KAPP       NEW YORK, N.Y.      2121212031 01 75$3509.88N00009100
06968D. CONRAD           WARWICK, ENGLAND    5671382126 11 81$0009.88Y00009200
007248B. C. WILLIAMSON   REDWOOD CITY, CALF. 3331212111 10 75$0009.88N00009400
007779MRS. W. WELCH      SAN JOSE, CALIF.    4151212003 01 75$0009.88Y00009500
100000G. NEADS           TORONTO, ONTARIO    0341512126 11 81$0010.00Y00009600
111111C. MEARS           OTTAWA, ONTARIO     5121200326 11 81$0011.00Y00009700
200000A. BONFIELD        GLASCOW, SCOTLAND   6373829026 11 81$0020.00Y00009900
300000K. TRENCHARD       NEW YORK, U.S.      6473980126 11 81$0030.00Y00010000
333333D. MYRING          CARDIFF, WALES      7849302026 11 81$0033.00Y00010100
400000W. TANNER          MILAN, ITALY        2536373826 11 81$0040.00Y00010200
444444A. FISHER          CALGARY, ALBERTA    7788982026 11 81$0044.00Y00010300
500000J. DENFORD         MADRID, SPAIN       4445464026 11 81$0000.00Y00010400
555555C. JARDINE         KINGSTON, N.Y.      3994442026 11 81$0005.00Y00010500
600000F. HUGHES          DUBLIN, IRELAND     1239878026 11 81$0010.00Y00010600
666666A. BROOKMAN        LA HULPE, BRUSSELS  4298384026 11 81$0016.00Y00010700
700000A. MACALLA         DALLAS, TEXAS       5798432026 11 81$0002.00Y00010800
777777D. PRYKE           WILLIAMSBURG, VIRG. 9187613126 11 81$0027.00Y00010900
800000H. BRISTOW         WESTEND, LONDON     2423338926 11 81$0030.00Y00011000
888888B. HOWARD          NORTHAMPTON, ENG.   2369163926 11 81$0038.00Y00011100
900000D. WOODSON         TAMPA, FLA.         3566812026 11 81$0040.00Y00011200
999999R. JACKSON         RALEIGH, N.Y.       8459163926 11 81$0049.00Y00011300
 *========================== End of File ====================================*
 *                                                                          *
 *    Closing Dpl Request has been attempted.                               *
 *    Close_Pipe call complete.                                             *
 *    Deallocate_Pipe call complete.                                        *
 *                                                                          *
 *=================== End of EXCI Sample Client Program =====================*
```

Figure 25. Successful execution

```
*==================== EXCI Sample Client Program ============================*
*                                                                          *
*  EXEC Level Processor.                                                   *
*    Setting up the EXEC level call.                                       *
*    The Link Request has failed.  Return codes are;                       *
*        Resp = 00000088  Resp2 = 00000203  Abend Code:                    *
*    >>>> Aborting further processing <<<<                                 *
*                                                                          *
*=================== End of EXCI Sample Client Program =====================*
```

Figure 26. No CICS return code. The target CICS region specified by the client program is not found, or IRC was not opened.

```
*==================== EXCI Sample Client Program ============================*
*                                                                          *
*  EXEC Level Processor.                                                   *
*    Setting up the EXEC level call.                                       *
*    The Link Request has successfully completed.                          *
*    Server Response:                                                      *
*      The file could not be found.                                        *
*    >>>> Aborting further processing <<<<                                 *
*                                                                          *
*=================== End of EXCI Sample Client Program =====================*
```

Figure 27. No file found. The target file name to the server program was not found on the target CICS system.

```
*==================== EXCI Sample Client Program ============================*
*                                                                          *
*  EXEC Level Processor.                                                   *
*    Setting up the EXEC level call.                                       *
*    The Link Request has failed.  Return codes are;                       *
*        Resp = 00000088  Resp2 = 00000414  Abend Code:                    *
*    A message was received from the target CICS system:                   *
*                                                                          *
 DFHAC2001 04/29/93 16:43:03 IYAHZCAZ Transaction 'BAD_' is unrecognized.  Check
 that the transaction name is correct.
*                                                                          *
*    >>>> Aborting further processing <<<<                                 *
*                                                                          *
*=================== End of EXCI Sample Client Program =====================*
```

Figure 28. Incorrect transaction identifier. The target transid passed in the external CICS interface call is not defined on the target CICS system. Note the message received from the target CICS system.

If an error occurs while running the application, then, assuming the error is not severe, messages are written to the SYSPRINT output log displaying the reasons and/or return codes that cause processing to be aborted. Several examples of error-invoked output are shown in Figure 26, Figure 27, and Figure 28.

An example of the output produced by a successful execution of the pregenerated assembler version of the client program DFH$ATXC is shown in Figure 23 on page 179. If an error occurs while running the application, errors are produced as for DFH$AXCC.

After running DFH$ATXC, program DFH$AXCC should be re-run. The output should be as shown in Figure 24 on page 180. The changes up to and including records with RIDFLD values of 500000 were committed using an SRRCMIT call to tell RRS to tell CICS to perform commit processing. Later changes were backed out by issuing an SRRBACK call. This caused RRS to tell CICS to perform a ROLLBACK of these changes.

Clearly, FILEA is changed as a result of running DFH$ATXC. It should be restored to its original state by running the LOADFILE step of DFHDEFDS.

Chapter 17. EXCI security

CICS applies security checks in a number of ways against requests received from an MVS client program.

These are described in the following topics:
- "Using MRO logon and bind-time security"
- "Link security" on page 188
- "User security" on page 189
- "Surrogate user checking" on page 189

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. This user ID is referred to as the batch region's user ID.

To enable your client program to log on successfully to IRP, and to connect to the target server region, first ensure that you define the batch region's user ID in a user profile to RACF. When you have defined the batch region's user ID to RACF, you can then give the batch region the appropriate logon and bind-time authorizations.

1. **Logon authorization**

   Authorize the batch region's user ID 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 batch region's user ID 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 188 for information about FACILITY class profiles for an EXCI client program.

2. **Bind-time authorization**

   Authorize the batch region's user ID to the DFHAPPL.*applid* RACF FACILITY class profile of the target CICS server region, with READ authority.

   Failure to authorize the batch region's user ID to the CICS server region's DFHAPPL.*applid* profile 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 the *CICS RACF Security Guide* 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 the *CICS RACF Security Guide* 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, is authorized as a surrogate of the user ID specified on the DPL_Request call).

EXCI client jobs are subject to surrogate user checking if SURROGCHK=YES (the default) is specified in the EXCI options table, DFHXCOPT. If you specify SURROGCHK=YES (or allow it to default) 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 *userid*.DFHEXCI in the SURROGAT general resource class (where*userid* is the user ID specified on the DPL call). For example, the following commands define a surrogate profile for a DPL userid, and grant READ access to the EXCI batch region:

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

If surrogate user checking is enabled (SURROGCHK=YES), but no 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 "Example of EXCI CALLs with null parameters" on page 146 for information about the correct way to specify a null pointer when omitting an EXCI call parameter.

If you don't want surrogate user security checking, specify SURROGCHK=NO in the DFHXCOPT options table (note that SURROGCHK=YES is the default).

Surrogate user checking is useful when the batch region's user ID is the same as the CICS server region user ID, in which case the link security check (see "Link security" on page 188) is bypassed. In this case, a surrogate user check is

recommended, because the user ID specified on the DPL_Request call is not an authenticated user ID (no password is passed).

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 the RACF security overview in the RACF Security Guide.

Chapter 18. Problem determination for the external CICS interface (EXCI)

Important: This information contains Diagnosis, Modification or Tuning information.

This topic describes some of the aids to problem determination provided by the external CICS interface.

It covers:
- Trace
- System dumps
- MVS 04xx abends for the external CICS interface
- The EXCI service trap, DFHXCTRA
- EXCI trace entry points

Details of the external CICS interface messages and abend codes are given in Chapter 20, "Messages and codes," on page 243.

Trace

The external CICS interface writes trace data to two destinations: an internal trace table and an external MVS GTF data set. The internal trace table resides in the non-CICS MVS address space. Trace data is formatted and included in any dumps produced by the external CICS interface.

Trace entries are issued by the external CICS interface destined for the internal trace table, an MVS GTF data set, or both. They are listed in "EXCI trace entry points" on page 201.

To use GTF for external CICS interface tracing, GTF user tracing must be active, GTF must be started in the MVS image, and you must specify GTF=ON in the DFHXCOPT options table.

If you use GTF trace for both the CICS server region and the external CICS interface region, the trace entries are interleaved, which can help you with problem determination in the CICS–EXCI environment.

Note: The external CICS interface maintains a separate trace table for each user TCB in an external CICS interface application program.

The external CICS interface does not support any form of auxiliary trace.

Formatting GTF trace

To format external CICS interface trace entries written to GTF, you can use the standard CICS DFHTR670 trace formatting routine.

To format external CICS interface trace entries you use the same FID and ID as for CICS (that is, FID=X'EF', and ID=X'F6C').

Using System dumps

The external CICS interface produces MVS SYSMDUMPs for some error conditions and MVS SDUMPs for other, more serious conditions. These dumps contain all the external CICS interface control blocks, as well as trace entries.

Formatting system dumps

You can use the CICS IPCS verb exit, DFHPD670, to format the system dumps.

The following keywords are available for use when formatting an external CICS interface dump using DFHPD670:

KE Formats PSW and registers, and all external CICS interface control blocks.

LD Formats a load map of where the external CICS interface modules are loaded in the address space, and gives their PTF level.

MRO
Formats the MRO control blocks for the external CICS interface address space, including common control blocks that reside in the MVS common service area (CSA). This option also formats some MRO blocks that reside in the CICS address space for pipes connected to CICS.

TR Formats the external CICS interface trace table. You can format the trace table in abbreviated and full forms (TR=1 gives you the abbreviated trace).

SU Produces a dump summary.

Multiple TCBs
If the external CICS interface takes a system dump when there is more than one TCB in use, it dumps only the control blocks and trace table for the TCB that requested the dump.

If you take a dump of the external CICS address space using a console command, the CICS verb exit routine, DFHPD670 formats the control blocks and trace tables for every TCB it finds in the dump.

Capturing SYSMDUMPs

To capture SYSMDUMPs produced by the external CICS interface, ensure you always include a DD statement for the SYSMDUMP data set in the client application program's JCL.

Using the MVS DUMP command at the console for dumps

In addition to the dumps taken automatically by the external CICS interface, you can also force a dump of an address space running a client application program by entering the **MVS DUMP** command at the console.

You can use the CICS IPCS verb exit routine DFHPD670 to format dumps taken in this way. You can also issue the DUMP command from TSO, SDSF, or NetView®.

You can also use the **DUMP** command to dump the CICS server address space as well as the client address space. Use the CICS IPCS verb exit routine DFHPD670 to format the dump that contains both address spaces.

MVS 04xx abends for the external CICS interface

These MVS 04xx abends can occur when you are running an external CICS interface job.

0401

An external CICS interface (EXCI) request was issued using the CALL API or the EXEC API, and the EXCI stub DFHXCSTB link-edited with the application detected that it was running in AMODE 24. The external CICS interface only supports calls made in AMODE 31.

Explanation

System Action

The application terminates abnormally.

User Response

Change the application so that EXCI calls are made in AMODE 31, or relink-edit the application AMODE 31.

Module

DFHXCSTB

0402

The external CICS interface module DFHXCPRH issued an MVS ESTAE macro to establish a recovery environment, but a nonzero return code was returned from MVS.

Explanation

System Action

The application terminates abnormally with a dump.

User Response

Examine the dump and any associated MVS messages produced to determine why the MVS ESTAE request failed.

If the error occurred while processing an INITIALIZE_USER request on behalf of the application, an attempt to format the dump using the CICS IPCS dump formatter does not produce any formatted output. This is because the error occurred too early in EXCI initialization for there to be any control blocks.

Module

DFHXCPRH

0403

The external CICS interface module DFHXCPRH issued an MVS GETMAIN request to obtain storage for its XCGLOBAL block, but a nonzero return code was returned from MVS.

Explanation

System Action

Module DFHXCPRH issues an MVS abend with abend code 0403 which invokes its ESTAE routine to clear up its environment. A SYSMDUMP is taken before returning control to the application. An application using the EXCI CALL API receives RESPONSE(SYSTEM_ERROR) REASON(XCGLOBAL_GETMAIN_ERROR) in its return area. The subreason1 field of the return area contains the R15 return code from MVS indicating why the GETMAIN failed. An application using the EXCI EXEC API receives RESP(LINKERR) RESP2(602).

User Response

Use the MVS R15 return code obtained from the application or from the dump to determine why the MVS GETMAIN request failed. If the reason is insufficient storage, increase the region size of the batch application.

An attempt to format the SYSMDUMP produced with the CICS IPCS dump formatter does not produce any formatted output for the job because the error occurred too early in EXCI initialization for there to be any control blocks.

Module

DFHXCPRH

0404

The external CICS interface module DFHXCPRH needed to take an MVS SDUMP for an earlier reported problem. However the error has occurred too early in EXCI initialization for EXCI dump services to be available.

Explanation

System Action

Module DFHXCPRH issues an MVS abend with abend code 0404 which invokes its ESTAE routine from which a SYSMDUMP is taken instead of an SDUMP to capture the earlier reported problem.

User Response

Examine the SYSMDUMP to determine the cause of the earlier reported problem.

An attempt to format the SYSMDUMP produced with the CICS IPCS dump formatter does not produce any formatted output for the job because the error occurred too early in EXCI initialization for there to be any control blocks.

Module

DFHXCPRH

0405

The external CICS interface module DFHXCPRH issued an IEFSSREQ SSI verify request to MVS to determine the number of the CICS SVC type 3 SVC to use. The SSI VERIFY request failed.

Explanation

System Action

Module DFHXCPRH issues an MVS abend with abend code 0405 which invokes its ESTAE routine to clear up its environment. A SYSMDUMP is taken before returning control to the application. An application using the EXCI CALL API receives RESPONSE(SYSTEM_ERROR) REASON(SSI_VERIFY_FAILED) in its return area. The subreason1 field of the return area contains the R15 return code from MVS indicating why the SSI verify failed. An application using the EXCI EXEC API receives RESP(LINKERR) RESP2(606).

User Response

Use the MVS R15 return code obtained from the application or from the dump to determine why the SSI VERIFY request failed.

An attempt to format the SYSMDUMP produced with the CICS IPCS dump formatter does not produce any formatted output for the job because the error occurred too early in EXCI initialization for there to be any control blocks.

Module

DFHXCPRH

0406

The external CICS interface module DFHXCPRH called the CICS SVC to initialize the EXCI environment. The CICS SVC call failed.

Explanation

System Action

Module DFHXCPRH issues an MVS abend with abend code 0406 which invokes its ESTAE routine to clear up its environment. A SYSMDUMP is taken before returning control to the application. An application using the EXCI CALL API receives RESPONSE(SYSTEM_ERROR) REASON(CICS_SVC_CALL_FAILURE) in its return area. The subreason1 field of the return area contains the R15 return code from the CICS SVC indicating why it failed. An application using the EXCI EXEC API receives RESP(LINKERR) RESP2(607).

User Response

Use the MVS R15 return code obtained from the application or from the dump to determine why the CICS SVC call failed.

An attempt to format the SYSMDUMP produced with the CICS IPCS dump formatter does not produce any formatted output for the job because the error occurred too early in EXCI initialization for there to be any control blocks.

Module

DFHXCPRH

0407

The external CICS interface module DFHXCPRH issued a call to the CICS SVC to check whether the SVC in use is at the correct level to be used with the external CICS interface. The check failed indicating that the CICS SVC is not at the correct level.

Explanation

System Action

Message DFHEX0100 is output, and module DFHXCPRH issues an MVS abend with abend code 0407 which invokes its ESTAE routine to clear up its environment. A SYSMDUMP is taken before returning control to the application. An application using the EXCI CALL API receives RESPONSE(SYSTEM_ERROR) REASON(INCORRECT_SVC_LEVEL) in its return area. An application using the EXCI EXEC API receives RESP(LINKERR) RESP2(627).

User Response

See the explanation of message DFHEX0100 for guidance.

An attempt to format the SYSMDUMP produced with the CICS IPCS dump formatter does not produce any formatted output for the job because the error occurred too early in EXCI initialization for there to be any control blocks.

Module

DFHXCPRH

0408

The external CICS interface module DFHXCPRH issued an MVS GETMAIN request for its working storage but a nonzero return code was returned from MVS.

Explanation

System Action

Module DFHXCPRH issues an MVS abend with abend code 0408 which invokes its ESTAE routine to clear up its environment. A SYSMDUMP is taken before returning control to the application. An application using the EXCI CALL API receives RESPONSE(SYSTEM_ERROR) REASON(WS_GETMAIN_ERROR) in its return area. The subreason1 field of the return area contains the R15 return code from MVS indicating why the GETMAIN failed. An application using the EXCI EXEC API receives RESP(LINKERR) RESP2(601).

User Response

Use the MVS R15 return code obtained from the application or from the dump to determine why the MVS GETMAIN request failed. If the reason is insufficient storage, increase the region size of the batch application.

An attempt to format the SYSMDUMP produced with the CICS IPCS dump formatter does not produce any formatted output for the job because the error occurred too early in EXCI initialization for there to be any control blocks.

Module

DFHXCPRH

0409

The external CICS interface module DFHXCPRH issued an MVS GETMAIN request for storage required for its SSI VERIFY request, but a nonzero return code was returned from MVS.

Explanation

System Action

Module DFHXCPRH issues an MVS abend with abend code 0409 which invokes its ESTAE routine to clear up its environment. A SYSMDUMP is taken before returning control to the application. An application using the EXCI CALL API receives RESPONSE(SYSTEM_ERROR) REASON(VERIFY_BLOCK_GM_ERROR) in its return area. The subreason1 field of the return area contains the R15 return code from MVS indicating why the GETMAIN failed. An application using the EXCI EXEC API receives RESP(LINKERR) RESP2(605).

User Response

Use the MVS R15 return code obtained from the application or from the dump to determine why the MVS GETMAIN request failed. If the reason is insufficient storage, increase the region size of the batch application.

An attempt to format the SYSMDUMP produced with the CICS IPCS dump formatter does not produce any formatted output for the job because the error occurred too early in EXCI initialization for there to be any control blocks.

Module

DFHXCPRH

0410

The external CICS interface module DFHXCPRH issued an MVS GETMAIN request for an XCUSER block but a nonzero return code was returned from MVS.

Explanation

System Action

Module DFHXCPRH issues an MVS abend with abend code 0410 which invokes its ESTAE routine to clear up its environment. A SYSMDUMP is taken before returning control to the application. An application using the EXCI CALL API receives RESPONSE(SYSTEM_ERROR) REASON(XCUSER_GETMAIN_ERROR) in its return area. The subreason1 field of the return area contains the R15 return code from MVS indicating why the GETMAIN failed. An application using the EXCI EXEC API receives RESP(LINKERR) RESP2(603).

User Response

Use the MVS R15 return code obtained from the application or from the dump to determine why the MVS GETMAIN request failed. If the reason is insufficient storage, increase the region size of the batch application.

Module

DFHXCPRH

0411

The external CICS interface dump module DFHXCDMP was attempting to call the CICS SVC in order for an MVS SDUMP to be taken to capture an earlier problem.

Explanation

DFHXCDMP was unable to call the SVC as no SVC number was available. DFHXCDMP issued an 0411 abend in order that the callers ESTAE routine is invoked which takes a SYSMDUMP instead.

System Action

A SYSMDUMP is taken instead of an SDUMP for an earlier reported problem.

User Response

Use the SYSMDUMP produced to diagnose the earlier reported problem.

An attempt to format the SYSMDUMP produced with the CICS IPCS dump formatter does not produce any formatted output for the job because the error occurred too early in EXCI initialization for there to be any control blocks.

Module

DFHXCDMP

0412

The external CICS interface dump module DFHXCEIP was processing an EXCI EXEC API request and detected that the EXEC parameter list passed to it contained a function that is not supported by the external CICS interface.

Explanation

System Action

The application is abnormally terminated with a dump.

User Response

This error indicates that the parameter list being passed to the EXCI has not been generated by the CICS translator. The translator should always be used. Correct the application to specify the correct EXCI EXEC API command.

An attempt to format the SYSMDUMP produced with the CICS IPCS dump formatter may not produce any formatted output for the job if this was the first EXCI request for this TCB.

Module

DFHXCEIP

0413

The external CICS interface dump module DFHXCEIP was processing an EXCI EXEC API request and detected that the EXEC parameter list passed to it did not require the mandatory RETCODE parameter in which return codes are returned to the application.

Explanation

An attempt to format the SYSMDUMP produced with the CICS IPCS dump formatter may not produce any formatted output for the job if this was the first EXCI request for this TCB.

System Action

The application is abnormally terminated with a dump.

User Response

This error indicates that the parameter list being passed to the EXCI has not been generated by the CICS translator. The translator should always be used. Correct the application to specify RETCODE.

Module

DFHXCEIP

0414

The external CICS interface module DFHXCEIP issued an MVS ESTAE macro to establish a recovery environment but a nonzero return code was returned from MVS.

Explanation

System Action

The application terminates abnormally with a dump.

User Response

Examine the dump and any associated MVS messages to determine why the MVS ESTAE request failed.

An attempt to format the SYSMDUMP produced with the CICS IPCS dump formatter may not produce any formatted output for the job if this was the first EXCI request for this TCB.

Module

DFHXCEIP

0415

The external CICS interface module DFHXCEIP detected an error early in EXCI initialization before EXCI dump services were available. DFHXCEIP issues abend 0415 so that its ESTAE routine is invoked from where an SYSMDUMP is taken instead to capture the error.

Explanation

System Action

The application terminates abnormally with a dump.

User Response

Examine the SYSMDUMP to determine the cause of the earlier reported error.

An attempt to format the SYSMDUMP produced with the CICS IPCS dump formatter does not produce any formatted output for the job because the error occurred too early in EXCI initialization for there to be any control blocks.

Module

DFHXCEIP

The EXCI service trap, DFHXCTRA

A user-replaceable program, DFHXCTRA, is available for use under the guidance of IBM service personnel. It is the equivalent of DFHTRAP used in CICS. It is invoked every time the external CICS interface writes a trace entry.

DFHXCTRA can perform one or all of the following actions:

1. Request the external CICS interface to write a trace entry on its behalf
2. Instruct the external CICS interface to take an SDUMP
3. Instruct the external CICS interface to skip writing the current trace entry to GTF
4. Instruct the external CICS interface to disable DFHXCTRA

The CICS-supplied sample version of DFHXCTRA performs all four of the above functions if it detects a trace entry that indicates that a FREEMAIN error occurred while trying to free an EXCI pipe control block.

The source for DFHXCTRA is supplied in CICSTS42.CICS.SDFHMAC. The parameter list passed to DFHXCTRA is defined in the copybook DFHXCTRD, which is supplied in CICSTS42.CICS.SDFHMAC.DFHXCTRD also defines all the external CICS interface trace points for use by DFHXCTRA.

Problem determination with RRMS

When Recoverable Resource Management Services (RRMS) is used to coordinate DPL requests, you can obtain additional problem determination information from RRMS.

To do this, you can use ISPF dialogs, provided by Resource Recovery Services (RRS), as follows:

- You can browse the RRS log streams.
- You can display information about RRS resource managers.
- you can dsplay information about RRS Units of Recovery.

For information about how to install and use the dialogs, see *z/OS MVS Programming: Resource Recovery*.

EXCI trace entry points

Table 25. External CICS interface trace entries

Point ID	Module	Lvl	Type	Data
EX 0001	DFHXCPRH	Exc	PIPE_ALREADY_OPEN	1. Caller's parameter list 2. Call type 3. Caller's user name 4. Return codes and message pointer 5. Pipe token
EX 0002	DFHXCPRH	Exc	PIPE_ALREADY_CLOSED	1. Caller's parameter list 2. Call type 3. Caller's user name 4. Return codes and message pointer 5. Pipe token
EX 0003	DFHXCPRH	Exc	VERIFY_BLOCK_FM_ERROR	1. Caller's parameter list 2. Call type 3. Caller's user name 4. Return codes and message pointer
EX 0005	DFHXCPRH	Exc	XCPIP_ FM_ERR	1. Caller's parameter list 2. Call type 3. Caller's user name 4. Return codes and message pointer 5. Pipe token
EX 0006	DFHXCPRH	Exc	IRP_IOAREA_FM_ERR	1. Caller's parameter list 2. Call type 3. Caller's user name 4. Return codes and message pointer
EX 0008	DFHXCPRH	Exc	XFRASTG1_FM_ERR	1. Caller's parameter list 2. Call type 3. Caller's user name 4. Return codes and message pointer 5. Pipe token

Table 25. External CICS interface trace entries (continued)

Point ID	Module	Lvl	Type	Data
EX 0201	DFHXCPRH	Exc	NO_CICS_IRC_STARTED	1. Caller's parameter list 2. Call type 3. Caller's user name 4. Return codes and message pointer
EX 0202	DFHXCPRH	Exc	NO_PIPE	1. Caller's parameter list 2. Call type 3. Caller's user name 4. Return codes and message pointer 5. Pipe token 6. Target CICS applid
EX 0203	DFHXCPRH	Exc	NO_CICS_ON_OPEN	1. Caller's parameter list 2. Call type 3. Caller's user name 4. Return codes and message pointer 5. Pipe token 6. Target CICS applid
EX 0204	DFHXCPRH	Exc	NO_CICS_ON_DPL_1	1. Caller's parameter list 2. Call type 3. Caller's user name 4. Return codes and message pointer 5. Pipe token 6. Target CICS applid
EX 0205	DFHXCPRH	Exc	NO_CICS_ON_DPL_2	1. Caller's parameter list 2. Call type 3. Caller's user name 4. Return codes and message pointer 5. Pipe token 6. Target CICS applid
EX 0206	DFHXCPRH	Exc	NO_CICS_ON_DPL_3	1. Caller's parameter list 2. Call type 3. Caller's user name 4. Return codes and message pointer 5. Pipe token 6. Target CICS applid

Table 25. External CICS interface trace entries (continued)

Point ID	Module	Lvl	Type	Data
EX 0403	DFHXCPRH	Exc	INVALID_APPL_NAME	1. Caller's parameter list 2. Call type 3. Caller's user name 4. Return codes and message pointer
EX 0405	DFHXCPRH	Exc	PIPE_NOT_CLOSED	1. Caller's parameter list 2. Call type 3. Caller's user name 4. Return codes and message pointer 5. Pipe token
EX 0406	DFHXCPRH	Exc	PIPE_NOT_OPEN	1. Caller's parameter list 2. Call type 3. Caller's user name 4. Return codes and message pointer 5. Pipe token
EX 0407	DFHXCPRH	Exc	INVALID_USERID	1. Caller's parameter list 2. Call type 3. Caller's user name
EX 0408	DFHXCPRH	Exc	INVALID_UOWID	1. Caller's parameter list 2. Call type 3. Caller's user name 4. UOWID
EX 0409	DFHXCPRH	Exc	INVALID_TRANSID	1. Caller's parameter list 2. Call type 3. Caller's user name
EX 0414	DFHXCPRH	Exc	ABORT_RECEIVED	1. Caller's parameter list 2. Call type 3. Caller's user name 4. Target CICS applid 5. Message to be returned
EX 0415	DFHXCPRH	Exc	INVALID_CONNECTION	1. Caller's parameter list 2. Call type 3. Caller's user name 4. Connection name 5. Target CICS applid
EX 0416	DFHXCPRH	Exc	INVALID_CICS_RELEASE	1. Caller's parameter list 2. Call type 3. Caller's user name 4. Target CICS applid

Table 25. External CICS interface trace entries (continued)

Point ID	Module	Lvl	Type	Data
EX 0417	DFHXCPRH	Exc	PIPE_MUST_CLOSE	1. Caller's parameter list 2. Call type 3. Caller's user name 4. Pipe token
EX 0418	DFHXCPRH	Exc	INVALID_PIPE_TOKEN	1. Caller's parameter list 2. Call type 3. Caller's user name 4. Pipe token
EX 0422	DFHXCPRH	Exc	SERVER_ABENDED	1. Caller's parameter list 2. Call type 3. Caller's user name 4. Return codes and message pointer 5. DPL return area
EX 0423	DFHXCPRH	Exc	SURROGATE_CHECK_FAILED	1. Caller's parameter list 2. Call type 3. Caller's user name 4. Job user ID 5. Surrogate resource name 6. ESM return code and reason code
EX 0426	DFHXCPRH	Exc	INVALID_TRANSID2	1. Caller's parameter list 2. Call type 3. Caller's user name
EX 0427	DFHXCPRH	Exc	INVALID_CCSID	1. Caller's parameter list 2. Call type 3. Caller's user name
EX 0428	DFHXCPRH	Exc	INVALID_ENDIAN	1. Caller's parameter list 2. Call type 3. Caller's user name
EX 0603	DFHXCPRH	Exc	XCUSER_GM_ERROR	1. Caller's parameter list 2. Call type 3. Caller's user name 4. Return codes and message pointer
EX 0604	DFHXCPRH	Exc	XCPIPE_GM_ERROR	1. Caller's parameter list 2. Call type 3. Caller's user name 4. Return codes and message pointer

Table 25. External CICS interface trace entries (continued)

Point ID	Module	Lvl	Type	Data
EX 0605	DFHXCPRH	Exc	VERIFY_BLOCK_GM_ERROR	1. Caller's parameter list 2. Call type 3. Caller's user name 4. Return codes and message pointer
EX 0606	DFHXCPRH	Exc	SSI_VERIFY_FAILED	1. Caller's parameter list 2. Call type 3. Caller's user name 4. Return codes and message pointer
EX 0607	DFHXCPRH	Exc	SVC_CALL_FAILED	1. Caller's parameter list 2. Call type 3. Caller's user name 4. Return codes and message pointer
EX 0608	DFHXCPRH	Exc	IRP_LOGON_FAILURE	1. Caller's parameter list 2. Call type 3. Caller's user name 4. Return codes and message pointer 5. Target CICS applid 6. Logon name
EX 0609	DFHXCPRH	Exc	IRP_CONNECT_FAIL	1. Caller's parameter list 2. Call type 3. Caller's user name 4. Return codes and message pointer 5. Pipe token 6. Target CICS applid
EX 0610	DFHXCPRH	Exc	IRP_DISC_FAIL	1. Caller's parameter list 2. Call type 3. Caller's user name 4. Return codes and message pointer 5. Target CICS applid 6. Pipe token
EX 0611	DFHXCPRH	Exc	IRP_LOGOFF_FAILED	1. Caller's parameter list 2. Call type 3. Caller's user name 4. Return codes and message pointer 5. Target CICS applid 6. Pipe token

Table 25. External CICS interface trace entries (continued)

Point ID	Module	Lvl	Type	Data
EX 0612	DFHXCPRH	Exc	TRANSFORM_1_ERROR	1. Caller's parameter list 2. Call type 3. Caller's user name 4. Return codes and message pointer
EX 0613	DFHXCPRH	Exc	TRANSFORM_4_ERROR	1. Caller's parameter list 2. Call type 3. Caller's user name 4. Return codes and message pointer
EX 0614	DFHXCPRH	Exc	IRP_NULL_DATA	1. Caller's parameter list 2. Call type 3. Caller's user name 4. Return codes and message pointer 5. Target CICS applid
EX 0615	DFHXCPRH	Exc	IRP_NEG_RESPONSE	1. Caller's parameter list 2. Call type 3. Caller's user name 4. Return codes and message pointer 5. Target CICS applid
EX 0616	DFHXCPRH	Exc	IRP_SWITCH_PULL_ERR	1. Caller's parameter list 2. Call type 3. Caller's user name 4. Return codes and message pointer 5. Target CICS applid 6. Pipe token
EX 0617	DFHXCPRH	Exc	IRP_IOAREA_GM_ERR	1. Caller's parameter list 2. Call type 3. Caller's user name 4. Return codes and message pointer
EX 0619	DFHXCPRH	Exc	IRP_BAD_IOAREA	1. Caller's parameter list 2. Call type 3. Caller's user name 4. IOAREA address
EX 0620	DFHXCPRH	Exc	IRP_PROTOCOL_ERR	1. Caller's parameter list 2. Call type 3. Caller's user name 4. Target CICS applid 5. Pipe token

Table 25. External CICS interface trace entries (continued)

Point ID	Module	Lvl	Type	Data
EX 0621	DFHXCPRH	Exc	PIPE_RECOVERY_FAILURE	1. Caller's parameter list 2. Call type 3. Caller's user name 4. Target CICS applid 5. Pipe token
EX 0622	DFHXCPRH	Exc	ESTAE_SETUP_FAIL	1. Caller's parameter list 2. Call type 3. Caller's user name 4. Return codes and message pointer
EX 0623	DFHXCPRH	Exc	ESTAE_INVOKED	1. Caller's parameter list 2. Call type 3. Caller's user name 4. Return codes and message pointer 5. MVS abend code
EX 0624	DFHXCPRH	Exc	TIMEDOUT	1. Caller's parameter list 2. Call type 3. Caller's user name 4. Server program name 5. Target CICS applid
EX 0625	DFHXCPRH	Exc	STIMER_SETUP_FAIL	1. Caller's parameter list 2. Call type 3. Caller's user name 4. Return codes and message pointer
EX 0626	DFHXCPRH	Exc	STIMER_CANCEL_FAIL	1. Caller's parameter list 2. Call type 3. Caller's user name 4. Return codes and message pointer
EX 0627	DFHXCPRH	Exc	INCORRECT_SVC_LEVEL	1. Caller's parameter list 2. Call type 3. Caller's user name 4. SVC instruction
EX 0800	DFHXCPRH	Exc	RESP shows LENGERR	1. Caller's parameter list 2. Call type 3. Caller's user name 4. Return codes and message pointer 5. COMMAREA length 6. Data length

Table 25. External CICS interface trace entries (continued)

Point ID	Module	Lvl	Type	Data
EX 0801	DFHXCPRH	Exc	RESP shows INVREQ	1. Caller's parameter list 2. Call type 3. Caller's user name 4. Return codes and message pointer 5. DPL options specified
EX 0802	DFHXCPRH	Exc	RESP shows PGMIDERR	1. Caller's parameter list 2. Call type 3. Caller's user name 4. Program name 5. Target CICS applid
EX 0803	DFHXCPRH	Exc	RESP shows ROLLEDBACK	1. Caller's parameter list 2. Call type 3. Caller's user name 4. Program name 5. Target CICS applid
EX 0804	DFHXCPRH	Exc	RESP shows NOTAUTH	1. Caller's parameter list 2. Call type 3. Caller's user name 4. Program name 5. Target CICS applid
EX 0805	DFHXCPRH	Exc	RESP shows SYSIDERR	1. Caller's parameter list 2. Call type 3. Caller's user name 4. Program name 5. Target CICS applid 6. DPL_Retarea
EX 0806	DFHXCPRH	Exc	RESP shows TERMERR	1. Caller's parameter list 2. Call type 3. Caller's user name 4. Program name 5. Target CICS applid
EX 0904	DFHXCTRP	Exc	Overlength trace data field	1. XCTRP parameter list
EX 0905	DFHXCTRA	Exc	DFHXCTRA trace entry	1. User specified data

Table 25. External CICS interface trace entries (continued)

Point ID	Module	Lvl	Type	Data
EX 1000	DFHXCPRH	EX 1	Entry	For INIT_USER commands: 1. Caller's parameter list 2. Call type 3. Caller's user name 4. Caller's register 14 For Allocate_Pipe requests: 1. Caller's parameter list 2. Call type 3. Caller's user name 4. CICS name 5. Allocate options 6. Caller's register 14 For Open, Close, and Deallocate requests: 1. Caller's parameter list 2. Call type 3. Caller's user name 4. CICS name 5. Pipe token 6. Caller's register 14 For DPL requests: 1. Caller's parameter list 2. Call type 3. Caller's user name 4. CICS name 5. Pipe token 6. Program name 7. Caller's register 14

Table 25. External CICS interface trace entries (continued)

Point ID	Module	Lvl	Type	Data
EX 1001	DFHXCPRH	EX 1	Exit	For INIT_USER, OPEN, CLOSE, and DEALLOCATE requests: 1. Caller's parameter list 2. Call type 3. Caller's user name 4. Return codes and message pointer 5. Caller's register14 For Allocate requests: 1. Caller's parameter list 2. Call type 3. Caller's user name 4. Return codes and message pointer 5. Pipe token 6. Caller's register 14 For DPL requests: 1. Caller's parameter list 2. Call type 3. Caller's user name 4. Target CICS system 5. Pipe token
EX 1010	DFHXCEIP	EX 1	Entry	1. Program name 2. Target CICS applid 3. Transaction ID 4. Caller's register 14 5. Up to the first 100 bytes of COMMAREA (if passed) 6. COMMAREA length, if COMMAREA passed 7. Data length, if COMMAREA passed
EX 1011	DFHXCEIP	EX 1	Exit	1. EXEC retarea 2. Program name 3. Target CICS applid 4. Transaction ID 5. Caller's register 14 6. Up to the first 100 bytes of COMMAREA (if passed) 7. COMMAREA length, if COMMAREA passed

Table 25. External CICS interface trace entries (continued)

Point ID	Module	Lvl	Type	Data
EX 2000	DFHXCPRH	EX 2	IRP_LOGON	1. Caller's parameter list 2. Call type 3. Caller's user name 4. Target CICS applid 5. IRP user ID 6. SLCB address 7. Connection name
EX 2001	DFHXCPRH	EX 2	IRP_CONN	1. Caller's parameter list 2. Call type 3. Caller's user name 4. Target CICS applid 5. IRP user ID 6. IRP thread ID 7. SCCB address
EX 2002	DFHXCPRH	EX 2	IRP_DISC	1. Caller's parameter list 2. Call type 3. Caller's user name 4. Target CICS applid 5. Pipe token
EX 2003	DFHXCPRH	EX 2	IRP_LOGOFF	1. Caller's parameter list 2. Call type 3. Caller's user name 4. Pipe token 5. IRP user ID
EX 2004	DFHXCPRH	EX 2	IRP_SWITCH	1. Caller's parameter list 2. Call type 3. Caller's user name 4. Target CICS applid 5. IRP user ID 6. IRP user thread
EX 2005	DFHXCPRH	EX 2	IRP_SWITCH_DATA	1. User's appl name 2. Pipe token 3. Request header 4. Bind data 5. UOWID/USERID FMH 6. Transformed DPL request to CICS (up to 1000 bytes) 7. Final 1000 bytes of transformed DPL request

Table 25. External CICS interface trace entries (continued)

Point ID	Module	Lvl	Type	Data
EX 2006	DFHXCPRH	EX 2	IRP_DATA	1. Caller's parameter list 2. Call type 3. Caller's user name 4. Target CICS applid 5. Length of data returned 6. Data (first 1000 bytes) 7. Data (final 1000 bytes)
EX 2007	DFHXCPRH	EX 2	PRE_URM	1. Caller's parameter list 2. Call type 3. Caller's user name 4. Parameters passed to DFHXCURM 5. URMINV, reason for calling URM 6. URMCICS, target CICS applid 7. URMANCH, URM anchor point address
EX 2008	DFHXCPRH	EX 2	POST_URM	1. Caller's parameter list 2. Call type 3. Caller's user name 4. Parameters passed to DFHXCURM 5. URMINV, reason for calling URM 6. URMCICS, target CICS applid 7. URMANCH, URM anchor point address
EX 2009	DFHXCPRH	EX 2	PRE-RACROUTE	1. Caller's parameter list 2. Call type 3. Caller's user name 4. Userid 5. Surrogate resource name 6. RACROUTE parameter list
EX 200A	DFHXCPRH	EX 2	POST-RACROUTE	1. Caller's parameter list 2. Call type 3. Caller's user name 4. Userid 5. Surrogate resource name 6. RACROUTE parameter list
EX 3000	DFHXCEIP	Exc	ESTAE_SETUP_ERROR	1. Return area (20 bytes) 2. MVS return code
EX 3001	DFHXCEIP	Exc	ESTAE_INVOKED	1. Return area (20 bytes)
EX 3002	DFHXCEIP	Exc	INV_CTYPE_ON_INIT	1. Return area (20 bytes) 2. Call type

Table 25. External CICS interface trace entries (continued)

Point ID	Module	Lvl	Type	Data
EX 3003	DFHXCEIP	Exc	INV_VNUM_ON_INIT	1. Return area (20 bytes) 2. Version number
EX 3004	DFHXCEIP	Exc	INV_APPL_NAME_ON_INIT	1. Return area (20 bytes) 2. User name
EX 3005	DFHXCEIP	Exc	INV_CTYPE_ON_ALLOC	1. Return area (20 bytes) 2. Call type
EX 3006	DFHXCEIP	Exc	INV_VNUM_ON_ALLOC	1. Return area (20 bytes) 2. Version number
EX 3007	DFHXCEIP	Exc	INV_UTOKEN_ON_ALLOC	1. Return area (20 bytes) 2. User token
EX 3008	DFHXCEIP	Exc	INV_CTYPE_ON_OPEN	1. Return area (20 bytes) 2. Call type
EX 3009	DFHXCEIP	Exc	INV_VNUM_ON_OPEN	1. Return area (20 bytes) 2. Version number
EX 3010	DFHXCEIP	Exc	INV_UTOKEN_ON_OPEN	1. Return area (20 bytes) 2. User token
EX 3011	DFHXCEIP	Exc	INV_PTOKEN_ON_OPEN	1. Return area (20 bytes) 2. Pipe token
EX 3012	DFHXCEIP	Exc	INV_CTYPE_ON_DPL	1. Return area (20 bytes) 2. Call type
EX 3013	DFHXCEIP	Exc	INV_VNUM_ON_DPL	1. Return area (20 bytes) 2. Version number
EX 3014	DFHXCEIP	Exc	INV_UTOKEN_ON_DPL	1. Return area (20 bytes) 2. User token
EX 3015	DFHXCEIP	Exc	INV_PTOKEN_ON_DPL	1. Return area (20 bytes) 2. Pipe token
EX 3017	DFHXCEIP	Exc	INV_USERID_ON_DPL	1. Return area (20 bytes)
EX 3018	DFHXCEIP	Exc	PIPE_NOT_OPEN_ON_DPL	1. Return area (20 bytes) 2. Pipe token
EX 3019	DFHXCEIP	Exc	PIPE_MUST_CLOSE_ON_DPL	1. Return area (20 bytes) 2. Pipe token
EX 3020	DFHXCEIP	Exc	INV_CTYPE_ON_CLOSE	1. Return area (20 bytes) 2. Call type
EX 3021	DFHXCEIP	Exc	INV_VNUM_ON_CLOSE	1. Return area (20 bytes) 2. Version number
EX 3022	DFHXCEIP	Exc	INV_UTOKEN_ON_CLOSE	1. Return area (20 bytes) 2. User token

Table 25. External CICS interface trace entries (continued)

Point ID	Module	Lvl	Type	Data
EX 3023	DFHXCEIP	Exc	INV_PTOKEN_ON_CLOSE	1. Return area (20 bytes) 2. Pipe token
EX 3024	DFHXCEIP	Exc	INV_CTYPE_ON_DEALL	1. Return area (20 bytes) 2. Call type
EX 3025	DFHXCEIP	Exc	INV_VNUM_ON_DEALL	1. Return area (20 bytes) 2. Version number
EX 3026	DFHXCEIP	Exc	INV_UTOKEN_ON_DEALL	1. Return area (20 bytes) 2. User token
EX 3027	DFHXCEIP	Exc	INV_PTOKEN_ON_DEALL	1. Return area (20 bytes) 2. Pipe token
EX 3028	DFHXCEIP	Exc	PIPE_NOT_CLOSED_ON_DEALL	1. Return area (20 bytes) 2. Pipe token
EX 3029	DFHXCEIP	Exc	XCEIP_RETRYING	1. Return area (20 bytes)
EX 3030	DFHXCEIP	Exc	SURROGATE_CHECK_FAILED	1. Return area (20 bytes)
EX 4000	DFHXCGUR	EX 1	Entry	1. DFHXCGUR parameter list
EX 4001	DFHXCGUR	EX 2	Exit	1. DFHXCGUR parameter list
EX 4002	DFHXCGUR	EX 1	PRE_SVC1	1. SVC parameter list
EX 4003	DFHXCGUR	EX 1	POST_SVC	1. SVC parameter list
EX 4004	DFHXCGUR	Exc	RRMS_NOT_SUPPORTED	1. None
EX 4005	DFHXCGUR	Exc	RRMS_ERROR	1. None
EX 4006	DFHXCGUR	Exc	SVC_EXCEPTION	1. SVC return code

Chapter 19. Response and reason codes returned on EXCI calls

This section gives details of the reason codes for the responses returned on the EXCI call interface.

Note: All numeric response and reason code values shown are in decimal.

Reason code for response: OK

0: NORMAL

Call completed normally.

Explanation

Reason codes for response: WARNING

1: PIPE_ALREADY_OPEN

An Open_Pipe request has been issued for a pipe that is already open.

Explanation

System Action

None. The pipe remains open.

User Response

If this response is unexpected, investigate whether an incorrect pipe token has been used on the Open_Pipe call.

2: PIPE_ALREADY_CLOSED

A Close_Pipe request has been issued for a pipe that is already closed.

Explanation

System Action

The external CICS interface ignores the request and the pipe remains closed.

User Response

If the response is unexpected, check that the Close_Pipe call is specifying the correct pipe token.

3: VERIFY_BLOCK_FM_ERROR

Initialize_User processing requires storage below 16MB to build the parameter list for the SSI Verify call, and an error has occurred during the FREEMAIN for this area.

Explanation

System Action

The return code from the FREEMAIN is returned in the EXCI subreason field-1. The Initialize_User request continues unaffected.

User Response

If the problem persists, take a dump of the batch region and use the dump, together with the return code from the MVS FREEMAIN, to determine why the FREEMAIN is failing.

4: WS_FREEMAIN_ERROR

An attempt to FREEMAIN working storage has resulted in an MVS FREEMAIN error.

Explanation

System Action

The return code from the FREEMAIN is returned in the EXCI subreason field-1. The Initialize_User request continues unaffected.

User Response

If the problem persists, take a dump of the batch region and use the dump, together with the return code from the MVS FREEMAIN to determine why the FREEMAIN is failing.

5: XCPIPE_FREEMAIN_ERROR

An attempt to FREEMAIN pipe storage has resulted in an MVS FREEMAIN error.

Explanation

System Action

The return code from the FREEMAIN is returned in the EXCI subreason field-1. However, the external CICS interface continues processing the Deallocate_Pipe request. If the request fails with another error, this reason code is overwritten.

User Response

If the problem persists, take a dump of the client application program address space, and use the dump, with the return code from the MVS FREEMAIN to determine why the FREEMAIN is failing.

6: IRP_IOAREA_FM_FAILURE

An attempt to FREEMAIN an MRO I/O area has resulted in an MVS FREEMAIN error.

Explanation

System Action

The return code from the FREEMAIN is returned in the EXCI subreason field-1, but the DPL request continued to completion. Reason IRP_IOAREA_FM_FAILURE is returned to your application only if the DPL request completes, otherwise it is overwritten by subsequent response and reason codes.

User Response

If the problem persists, take a dump of the batch region and use it with the return code from the MVS FREEMAIN to determine why the FREEMAIN is failing.

7: SERVER_TERMINATED

The CICS session, on which the server program has been executing, has been freed by CICS.

Explanation

System Action

The CICS application server program has been detached at some point in its processing, and control is returned to the external CICS interface, which writes a trace entry for this error.

User Response

The most likely reason for this error is that the server program has caused CICS to terminate, perhaps by an EXEC CICS PERFORM SHUTDOWN command. During shutdown, CICS frees EXCI sessions so that shutdown can complete.

8: XFRASTG1_FM_FAILURE

An attempt to FREEMAIN the transmission area has resulted in an MVS FREEMAIN error.

Explanation

System Action

The return code from the FREEMAIN is returned in the EXCI subreason field-1 but the DPL request continued to completion. Reason XFRASTG1_FM_FAILURE is returned to your application only if the DPL request completes, otherwise it is overwritten by subsequent response and reason codes.

User Response

If the problem persists, take a dump of the batch region and use it with the return code from the MVS FREEMAIN to determine why the FREEMAIN is failing.

Reason codes for response: RETRYABLE

201: NO_CICS_IRC_STARTED

An Initialize_User command has been issued on an MVS image that has had no IRC activity since the previous IPL, and the external CICS interface cannot determine the CICS SVC number.

Explanation

System Action

The Initialize_User call fails, and the external CICS interface invokes the user-replaceable module, DFHXCURM.

User Response

Ensure that a CICS region in the MVS image has logged on to IRC (that is, has started up with the system initialization parameter IRCSTRT=YES or has started IRC dynamically with an OPEN IRC command). Alternatively, if there is no local CICS region in the MVS image, you must specify the SVC parameter that the external CICS interface is to use, by coding a CICSSVC parameter in the DFHXCOPT table. This situation can occur if you are using XCF to communicate to a CICS region in another MVS image. Once the problem has been resolved, re-issue the Initialize_User request.

202: NO_PIPE

An attempt has been made to open a pipe, but the target CICS system associated with the pipe has no free receive sessions.

Explanation

System Action

The Open_pipe call fails, and the external CICS interface invokes the user-replaceable module, DFHXCURM.

User Response

This situation can occur even if the client application program has allocated (using Allocate_Pipe calls) no more pipes than the number of receive sessions defined on the target connection. This is because CICS can be in the process of cleaning up a pipe from a Close_Pipe request. For this reason, you are recommended to specify a larger RECEIVECOUNT value than is theoretically necessary when defining the SESSIONS resource definition to CICS. The application program can reissue the Open_Pipe request.

203 (on Open_Pipe call): NO_CICS

An attempt has been made to open a pipe but the target CICS system is not available, or hasn't yet opened IRC, or the target connection is out of service, or the relevant EXCI connection definition is not installed in the target CICS.

Explanation

System Action

The open pipe request fails, and the external CICS interface invokes the user-replaceable module, DFHXCURM.

User Response

If subreason field-1 is non-zero (the IRP response code (R15)), subreason field-2 contains the IRP reason code. For an explanation of the IRP return codes, see the

interregion control blocks in the *CICS Data Areas* manual. The IRP return codes are in the DFHIRSPS copybook, listed under the heading IRC.

When you have corrected the problem, your client application program can reissue the Open_Pipe call.

204: WRONG_MVS_FOR_RRMS

A DPL request omitting the SYNCONRETURN option has been made specifying a CICS region that is on a different MVS system from the batch program. Because the Recoverable Resource Management Services (RRMS) context is not recognised in the target system, the request is rejected.

Explanation

System Action

The DPL request fails, and the external CICS interface invokes the user-replacable module, DFHXCURM.

User Response

Ensure that the batch program that issued the DPL request and the CICS region it was sent to are on the same MVS system.

205: RRMS_NOT_AVAILABLE

A DPL request omitting the SYNCONRETURN option has been made when the Resource Recovery Services (RRS) is not available.

Explanation

There are two cases:
- When Resource Recovery Services (RRS) is not available.
- When Resource Recovery Services has restarted since the last DPL request omitting the SYNCONRETURN option, and there has been no intervening syncpoint.

- -

Note: RRS is a part of Recoverable Resource Management Services (RRMS).

System Action

The DPL request fails, and the external CICS interface invokes the user-replacable module, DFHXCURM.

User Response

Retry the DPL request when Resource Recovery Services has restarted since the last DPL request omitting the SYNCONRETURN option, and there has been no intervening syncpoint.

Reason codes for response: USER_ERROR

401: INVALID_CALL_TYPE

An invalid *call-type* parameter value is specified on this EXCI request.

Explanation

System Action

The request is rejected.

User Response

Check your EXCI client program and ensure the *call_type* parameter specifies the appropriate value for the EXCI call, as follows.

1   Initialize_User

2   Allocate_Pipe

3   Open_Pipe

4   Close_Pipe

5   Deallocate_Pipe

6   DPL

402: INVALID_VERSION_NUMBER

The *version_number* parameter does not specify a value of 1 or 2.

Explanation

System Action

The request is rejected.

User Response

Check the client application program and ensure that all EXCI calls specify the value of 1 or 2 for the version number.

403: INVALID_APPL_NAME

The *user_name* parameter consists of all blank characters (X'40').

Explanation

System Action

The call is rejected.

User Response

Change the application program to specify a valid, non-blank user name.

404: INVALID_USER_TOKEN

The client application program has issued an EXCI request using a user token that is unknown to the external CICS interface.

Explanation

System Action

The request is rejected.

User Response

The Initialize_User call returns a 4-byte token that must be used on *all* further requests for the that user. Check the client application program and correct the error to ensure that the correct token is passed.

405: PIPE_NOT_CLOSED

A Deallocate_Pipe request has been issued against a pipe that has not yet been closed.

Explanation

System Action

The external CICS interface ignores the request and the pipe remains open.

User Response

Check the client application program, and ensure that the Deallocate_Pipe request is intended. If so, issue a Close_Pipe request for the pipe before issuing the Deallocate_Pipe request.

406: PIPE_NOT_OPEN

A DPL call has been issued on a pipe that is not open.

Explanation

System Action

The external CICS interface rejects the DPL request.

User Response

Check the client application program, and ensure that an Open_Pipe request is issued before using the pipe on a DPL request. If an Open_Pipe has been issued by the application program, check that it has not been closed inadvertently before all the DPL requests have been made.

407: INVALID_USERID

A DPL request has been issued with a USERID parameter that consists of all blanks.

Explanation

System Action

The DPL request is rejected.

User Response

Check the EXCI client program and ensure that the DPL request passes a valid USERID parameter. If you don't want to specify a userid, code the call parameter list with a null address for *userid*. If you pass a null address, the external CICS interface passes the userid under which the client application program is running (the batch region's userid).

408: INVALID_UOWID

A DPL request has been issued with a *uowid* parameter that has invalid length fields.

Explanation

System Action

The DPL request is rejected.

User Response

Check the client application program and ensure that the DPL request passes a valid *uowid* parameter. If you don't want to specify a unit of work id, code the call parameter list with a null address for *uowid*, in which case the external CICS interface generates a unit of work id for you.

409: INVALID_TRANSID

A DPL request has been issued with a *transid* parameter that consists of all blanks.

Explanation

System Action

The DPL request is rejected.

User Response

Check the client application program and ensure that the *transid* parameter is specified correctly or has not been overwritten in some way. If you don't want to specify your own transid, code the call parameter list with a null address for *transid*, in which case the external CICS interface uses the default CICS mirror transaction, CSMI.

410: DFHMEBM_LOAD_FAILED

During Initialize_User processing, the external CICS interface attempted to load the main message module in preparation for issuing external CICS interface messages, and the load of this module failed.

Explanation

System Action

The Initialize_User call is rejected. The return code from the MVS load macro (R15) is returned in the subreason field-1. The external CICS interface handles the error, and returns the abend (R0) that would have occurred in the subreason field-2.

User Response

Using the MVS return code, determine why the load failed. The most likely reason is that the message module, DFHMEBMX, is not in any library included in the STEPLIB of the batch job. Ensure the CICSTS42.CICS.SDFHEXCI library is included in the STEPLIB concatenation, and restart the client application program.

411: DFHMET4E_LOAD_FAILED

The load of message module, DFHMET4E, has failed. During Initialize_User processing, the external CICS interface attempted to load its message table in preparation for issuing messages. The load of this module failed.

Explanation

System Action

The Initialize_User call is rejected. The return code from the MVS load macro (R15) is returned in the subreason field-1. The external CICS interface handles the error, and returns the abend (R0) that would have occurred in the subreason field-2.

User Response

Using the MVS reason code, determine why the load failed. The most likely reason is that the message table, DFHMET4E, is not in any library included in the STEPLIB of the batch job. Ensure the CICSTS42.CICS.SDFHEXCI library is included in the STEPLIB concatenation, and restart the client application program.

412: DFHXCURM_LOAD_FAILED

During Initialize_User processing, the external CICS interface attempted to load the user-replaceable module, DFHXCURM. The load of this module failed.

Explanation

System Action

The Initialize_User call is rejected. The return code from the MVS load macro (R15) is returned in the subreason field-1. The external CICS interface handles the error, and returns the abend (R0) that would have occurred in the subreason field-2.

User Response

Using the MVS reason code, determine why the load failed. The most likely reason is that module DFHXCURM is not in any library included in the STEPLIB of the batch job. Ensure the library containing the module is included in the STEPLIB concatenation, and restart the client application program.

413: DFHXCTRA_LOAD_FAILED

During Initialize_User processing, the external CICS interface attempted to load the trap module (DFHXCTRA). The load of this module has failed.

Explanation

System Action

The Initialize_User call is rejected. The return code from the MVS load macro (R15) is returned in the subreason field-1. The external CICS interface handles the error, and returns the abend (R0) that would have occurred in the subreason field-2.

User Response

Using the MVS reason code, determine why the load failed. The most likely reason is that DFHXCTRA is not in any library included in the STEPLIB of the batch job. Ensure the library containing the module is included in the STEPLIB concatenation, and restart the client application program.

414: IRP_ABORT_RECEIVED

While processing a DPL request, an error occurred in the CICS server region, resulting in an abort FMH7 flow being returned to the external CICS interface.

Explanation

System Action

A message is returned to the client application program. This is the message that would have been issued to the terminal if the server program had been initiated from a terminal. A pointer to the message is returned to the client application program in the message pointer field of the EXCI return area. See the description of the EXCI return areas for the exact definition of the message format. The pipe is put into a "must close" state.

User Response

Use the message to determine the cause of the error. A typical example is where the server transaction cannot be attached, either because is disabled, or it has not been defined, or because of a security failure. Correct the problem, close and reopen the pipe, and reissue the DPL request.

415: INVALID_CONNECTION_DEFN

A DPL request has been rejected by CICS because the target connection is not defined for use by an external CICS client application program.

Explanation

System Action

The DPL request is rejected and the pipe is put into a "must close" state.

User Response

The most likely reason for this is that the connection definition in the CICS server region has been defined incorrectly as a CICS-to-CICS MRO connection, instead of an EXCI connection. Ensure that PROTOCOL(EXCI) is specified on the appropriate CONNECTION and SESSIONS resource definitions. You must close and reopen the pipe before reissuing the DPL request.

416: INVALID_CICS_RELEASE

A DPL request has been rejected by the target CICS server region because it doesn't recognize the request.

Explanation

System Action

The DPL call is rejected and the pipe is put into a "must close" state.

User Response

The most likely reason for this is that the client application program has specified a target CICS server region that does not support the external CICS interface. CICS regions earlier than CICS for MVS/ESA 4.1 do not recognize EXCI call requests. Correct the problem, close and reopen the pipe and then reissue the DPL request.

417: PIPE_MUST_CLOSE

A DPL request has been issued on a pipe that is in a "must close" state.

Explanation

System Action

The DPL request is rejected.

User Response

Some EXCI errors are serious enough to require that the pipe be closed and reopened in order to restore the pipe to a point where it can be used for further DPL requests. Others, more minor errors, allow further calls without closing and reopening the pipe. A previous error on this pipe has been of the more serious variety and the pipe is now in a "must close" state. Close and reopen the pipe and reissue the DPL request.

418: INVALID_PIPE_TOKEN

An Open_Pipe, Close_Pipe, Deallocate_Pipe, or DPL request has been issued, but the pipe token passed on the call is either not a valid pipe, or is not a valid pipe allocated for this user (that is, there is mismatch between the user token and the pipe token).

Explanation

System Action

The call is rejected.

User Response

Ensure that the pipe token has not been overwritten and is being passed correctly on the call. Also ensure there is no mismatch between the user token and the pipe token.

419: CICS_AFCB_PRESENT

An Initialize_User request has been issued on a TCB that has already been used by CICS or by CICS batch shared database. The external CICS interface cannot share a TCB with CICS, ensuring that a CICS application program cannot issue EXCI requests.

Explanation

System Action

The Initialize_User request is rejected.

User Response

To use the external CICS interface, you must create a new TCB (or daughter TCB), and issue the EXCI calls under that unique TCB.

420: DFHXCOPT_LOAD_FAILED

During Initialize_User processing, the external CICS interface attempted to load its options module, DFHXCOPT. The load of this module failed.

Explanation

System Action

The Initialize_User call is rejected. The return code from the MVS load macro (R15) is returned in the subreason field-1. The external CICS interface handles the error, and returns the abend (R0) that would have occurred in the subreason field-2.

User Response

Using the MVS reason code, determine why the load failed. The most likely reason is that DFHXCOPT is not in any library included in the STEPLIB of the batch job. Correct the problem and restart the client application program.

421: RUNNING_UNDER_AN_IRB

The EXCI call is issued under an MVS IRB, which is not permitted.

Explanation

System Action

The call is rejected.

User Response

Determine why the call was issued under an IRB and change the client application program.

422: SERVER_ABENDED

While processing a DPL request, the CICS server application program abended without handling the error.

Explanation

System Action

The server application program is abended and backout out. The abend code is returned in the abend code field of the EXCI return area.

User Response

Determine why the server program abended and fix the problem.

423: SURROGATE_CHECK_FAILED

A DPL request has been issued specifying a USERID parameter.

Explanation

The userid specified was subject to a surrogate user security because SURROGCHK=YES is specified in the EXCI options table, DFHXCOPT. The surrogate user check failed. The surrogate security check verifies whether the EXCI batch region's userid is authorized as a surrogate of the userid specified on the DPL call.

System Action

The DPL call is rejected. The MVS external security manager's return code and reason code are returned in subreason field-1 and field-2. For RACF, these documented in the *External Security Interface (RACROUTE) Macro Reference for MVS, GC23-3733*.

User Response

If you want surrogate user checking, ensure that the EXCI batch region's userid has READ access to the profile *userid*.DFHEXCI in the SURROGAT general resource class, where *userid* is the userid specified on the DPL call.

If you don't want surrogate user security checking, specify SURROGCHK=NO in the DFHXCOPT options table.

See "Surrogate user checking" on page 189 for more information.

424: RRMS_NOT_SUPPORTED

A DPL request omitting the SYNCONRETURN option has been made on a system that is not running z/OS Release 5 (or a later, upward-compatible, release).

Explanation

System Action

The call is rejected.

User Response

Ensure that the batch program is run on a system that is running the correct level of z/OS.

425: UOWID_NOT_ALLOWED

A DPL request omitted the SYNCONRETURN option, but specified a value of UOWID. This combination of parameters is not permitted on a DPL request.

Explanation

System Action

The DPL_Request is rejected.

User Response

Check the client application program and ensure that the correct combination of parameters is used on the DPL call.

426: INVALID_TRANSID2

A DPL request has been issued with a *transid2* parameter that consists of all blanks.

Explanation

System Action

The DPL request is rejected.

User Response

Check the client application program and ensure that the *transid2* parameter is specified correctly or has not been overwritten in some way.

427: INVALID_CCSID

A DPL request has been issued with a *ccsid* parameter that specifies an invalid value.

Explanation

System Action

The DPL request is rejected.

User Response

Check the client application program and ensure that the *ccsid* parameter is specified correctly or has not been overwritten in some way.

428: INVALID_ENDIAN

A DPL request has been issued with a *endian* parameter that specifies an invalid value.

Explanation

System Action

The DPL request is rejected.

User Response

Check the client application program and ensure that the *endian* parameter is specified correctly or has not been overwritten in some way.

429: DFHXCEIX_LOAD_FAILED

During processing of an **EXEC CICS LINK** call, the external CICS interface attempted to load the module (DFHXCEIX). The load of this module has failed.

Explanation

System Action

The **EXEC CICS LINK** call is rejected.

User Response

The most likely reason is that DFHXCEIX is not in any library included in the STEPLIB of the batch job. Ensure the library containing the module is included in the STEPLIB concatenation, and restart the client application program.

430: DFHXCPRX_LOAD_FAILED

During Initialize_User processing, the external CICS interface attempted to load the module (DFHXCPRX). The load of this module has failed.

Explanation

System Action

The Initialize_User call is rejected. The return code from the MVS load macro (R15) is returned in the subreason field-1. The external CICS interface handles the error, and returns the abend (R1) that would have occurred in the subreason field-2.

User Response

Using the MVS reason code, determine why the load failed. The most likely reason is that DFHXCPRX is not in any library included in the STEPLIB of the batch job. Ensure the library containing the module is included in the STEPLIB concatenation, and restart the client application program.

Reason codes for response: SYSTEM_ERROR

601: WS_GETMAIN_ERROR

During Initialize_User processing, a GETMAIN for working storage failed.

Explanation

System Action

Processing cannot continue without working storage, so the request is terminated. At this point the external CICS interface trace and dump services are not available to provide diagnostic information, therefore EXCI issues an MVS abend (U0408) to force a SYSMDUMP. The return code from the MVS GETMAIN request is returned in the return area.

User Response

Locate the GETMAIN return code in the dump, and use this and the rest of the dump to determine why the GETMAIN failed. A possible reason for this is that the region size specified for the job is too small. If this is the case, increase the region size and restart the client application program.

602: XCGLOBAL_GETMAIN_ERROR

During Initialize_User processing, a GETMAIN failed for a critical control block (XCGLOBAL).

Explanation

System Action

Processing cannot continue without this control block, and the request is terminated. At this point the external CICS interface trace and dump services are not available to provide diagnostic information, therefore EXCI issues an MVS abend (U0403) to force a SYSMDUMP. The return code from the MVS GETMAIN request is returned in the return area.

User Response

Locate the GETMAIN return code in the dump, and use this and the rest of the dump to determine why the GETMAIN failed. A possible reason for this is that the region size specified for the job is too small. If this is the case, increase the region size and restart the client application program.

603: XCUSER_GETMAIN_ERROR

During Initialize_User processing, a GETMAIN request failed for the user control block (XCUSER).

Explanation

System Action

Initialize_User processing is terminated. The return code from the GETMAIN is returned in subreason field-1 of the return area. The external CICS interface issues message DFHEX0003 and issues an MVS user abend (0410) to force a SYSMDUMP.

User Response

Use the return code from the GETMAIN, with the dump, to determine why the GETMAIN failed. A possible reason for this is that the region size of the job is too small. If this is the case, increase the region size and restart the client application program.

604: XCPIPE_GETMAIN_ERROR

During Allocate_Pipe processing, a GETMAIN request for the pipe control block (XCPIPE) failed.

Explanation

System Action

Allocate_Pipe processing is terminated. The return code from the GETMAIN is returned in subreason field-1 of the EXCI return area. The external CICS interface issues message DFHEX0003, and takes a system dump.

User Response

Use the return code from the GETMAIN, and the dump, to determine why the GETMAIN failed. A possible reason for this is that the region size for the job is too small. If this is the case, increase the region size and restart the client application program.

605: VERIFY_BLOCK_GM_ERROR

During Initialize_User processing, a GETMAIN failed for an EXCI internal control block.

Explanation

System Action

Initialize_User processing is terminated. The return code from the GETMAIN is returned in the subreason field-1 of the EXCI return area. This error occurs before EXCI dumping services are initialized, Therefore EXCI issues an MVS abend (U0409) to force a SYSMDUMP The return code from the MVS GETMAIN request is returned in the return area.

User Response

Locate the GETMAIN return code in the dump, and use this and the rest of the dump to determine why the GETMAIN failed. A possible reason for this is that the region size specified for the job is too small. If this is the case, increase the region size and restart the client application program.

606: SSI_VERIFY_FAILED

A VERIFY call to the MVS subsystem interface (SSI) to obtain the current CICS SVC number failed.

Explanation

System Action

The Initialize_User request is terminated. The return code from the SSI call is returned in subreason field-1 of the return area. This error occurs before the external CICS interface dump services are initialized, therefore EXCI issues an MVS user abend (0405) to force a SYSMDUMP.

User Response

Locate the return code in the dump, and use this with the rest of the dump and SSI documentation to determine why the VERIFY request failed. When the problem is resolved, restart the client application program.

607: CICS_SVC_CALL_FAILURE

During Initialize_User processing, a call to the currently installed CICS SVC failed.

Explanation

System Action

The return code from the CICS SVC is returned in the subreason field-1 of the EXCI return area. This error occurs before the external CICS interface dump services are initialized, therefore EXCI issues an MVS user abend (0406) to force a SYSMDUMP.

User Response

Contact your IBM support center for assistance, with the return code and the dump available.

608: IRC_LOGON_FAILURE

During Allocate_Pipe processing, an attempt by the external CICS interface to LOGON to DFHIRP failed.

Explanation

System Action

The Allocate_Pipe request fails. DFHIRP returns a R15 value to subreason field-1 and a R0 value (the reason code) to subreason field-2. The first two bytes of subreason field-1 are the return code qualifier and the last two bytes are the return code itself.

User Response

For an explanation of the IRP return codes, see the interregion control blocks in the *CICS Data Areas* manual. The IRP return codes are in the DFHIRSPS copybook, listed under the heading IRC. Use the return codes to determine why the logon failed, or contact your IBM support personal with details of the failure.

609: IRC_CONNECT_FAILURE

During Open_Pipe processing, an attempt to connect to the target CICS system failed.

Explanation

System Action

The Open_Pipe request fails. DFHIRP returns a R15 value to subreason field-1 and a R0 value (the reason code) to subreason field-2. The first two bytes of subreason field-1 are the return code qualifier and the last two bytes are the return code itself.

User Response

For an explanation of the IRP return codes, see the interregion control blocks in the *CICS Data Areas* manual. The IRP return codes are in the DFHIRSPS copybook, listed under the heading IRC.

Use the return code to determine why the logon failed, and reissue the open pipe request.

Note: This error is not caused by the target CICS being unavailable, which is returned as a RETRYABLE condition (NO_CICS).

610: IRC_DISCONNECT_FAILURE

During Close_Pipe processing, CICS issued a DFHIRP disconnect call to terminate the connection to CICS. This request has failed.

Explanation

System Action

The call fails and the pipe is left open. DFHIRP returns a R15 value to subreason field-1 and a R0 value (the reason code) to subreason field-2. The first two bytes of subreason field-1 are the return code qualifier and the last two bytes are the return code itself.. The external CICS interface takes a system dump.

Although the disconnect failed, it is possible that the pipe is still connected to CICS. However, all connections are automatically disconnected at the end of the batch program.

User Response

For an explanation of the IRP return codes, see the interregion control blocks in the *CICS Data Areas* manual. The IRP return codes are in the DFHIRSPS copybook, listed under the heading IRC. Use the return code and the dump to determine the cause of the error.

611: IRC_LOGOFF_FAILURE

During Deallocate_Pipe processing, CICS issued a DFHIRP logoff call. This request failed.

Explanation

System Action

The Deallocate_Pipe call fails and the pipe remains allocated. DFHIRP returns a R15 value to subreason field-1 and a R0 value (the reason code) to subreason field-2. The first two bytes of subreason field-1 are the return code qualifier and the last two bytes are the return code itself. The external CICS interface takes a system dump.

Note: Because it remains allocated, the pipe is available for further calls. Any storage associated with the pipe is not freed. However, this storage is freed at the end of the client application program.

User Response

For an explanation of the IRP return codes, see the interregion control blocks in the *CICS Data Areas* manual. The IRP return codes are in the DFHIRSPS copybook, listed under the heading IRC. Use the return code and the dump to determine the cause of the error.

612: TRANSFORM_1_ERROR

During DPL processing, while processing the data in preparation for sending to CICS, an internal call to program DFHXFQ resulted in an error.

Explanation

System Action

The DPL request is terminated.

User Response

The return code from the call is returned in the EXCI subreason field-1, and the external CICS interface takes a system dump.

This is an external CICS interface error. Contact your IBM support center with details of the return code and the dump.

613: TRANSFORM_4_ERROR

During DPL processing, while processing the data returned by the CICS server region, an internal call to module DFHXFQ resulted in an error.

Explanation

System Action

The DPL request is terminated. Note that the server application program has executed. The return code from the call to DFHXFQ is returned in the EXCI subreason field-1. This return code corresponds to any EIBRCODE information that was available. The external CICS interface takes a system dump.

User Response

This is an external CICS interface error. Contact your IBM support center with details of the return code and the dump.

614: IRP_NULL_DATA_RECEIVED

During DPL processing, a request has been sent to the target CICS and this target CICS has replied without returning any data.

Explanation

System Action

The DPL processing is terminated and the external CICS interface takes a system dump.

User Response

This is an internal protocol error. Contact your IBM support center with details of the dump.

615: IRP_NEGATIVE_RESPONSE

An internal protocol error has occurred while trying to communicate with the target CICS region.

Explanation

System Action

The DPL request fails, the pipe is put into a "must close" state, and the external CICS interface takes a system dump.

User Response

This is an external CICS interface error. Keep the dump and contact your IBM support center.

Note: The pipe is in a "must close" state. Before attempting further calls, the pipe must first be closed and reopened.

616: IRP_SWITCH_PULL_FAILURE

An internal protocol error has occurred while trying to communicate with the target CICS region.

Explanation

System Action

The DPL request fails, the pipe is put into a "must close" state, and the external CICS interface takes a system dump. The IRP return code (R15) and reason code if any (R0) are returned in the EXCI subreason field-1 and subreason field-2.

User Response

This is an external CICS interface error. Keep the dump and contact your IBM support center.

Note: The pipe is in a "must close" state, and before attempting further DPL calls, the pipe must first be closed and reopened.

617: IRP_IOAREA_GM_FAILURE

During DPL processing, an MVS GETMAIN request for an internal control block failed.

Explanation

System Action

The DPL request is terminated. The return code from the GETMAIN is returned in the EXCI subreason field-1.

Note: This error occurs while processing the data returned by CICS, after the server application program has completed execution. This error results in the pipe being put into a "must close" state.

User Response

Use the return code to determine why the GETMAIN failed. A possible reason for this is that the region size of the job is too small. If this is the case, increase the region size and restart the batch job.

619: IRP_BAD_IOAREA

During a DPL request, an I/O area has been supplied to DFHIRP that could not be used.

Explanation

System Action

The DPL request is terminated, the pipe is forced into a "must close" state, and the external CICS interface takes a system dump.

User Response

This is an external CICS interface error. Contact the IBM support center with details of the return code and the dump.

Note: The pipe is in a "must close" state after this error, and before attempting further calls must first be closed and reopened.

620: IRP_PROTOCOL_ERROR

An internal protocol error has occurred while trying to communicate with the target CICS system.

Explanation

System Action

The DPL request is terminated, the pipe is forced into a "must close" state, and the external CICS interface takes a system dump.

User Response

This is an external CICS interface error. Keep the dump and contact your IBM support center.

Note: The pipe is in a "must close" state after this error, and before attempting further calls must first be closed and reopened.

621: PIPE_RECOVERY_FAILURE

An error has occurred during an open pipe request. The external CICS interface attempts to recover by disconnecting the pipe again. During this disconnection, further errors have occurred.

Explanation

System Action

The Open_Pipe call is terminated and the pipe is placed in a "must close" state. The return code from DFHIRP is returned in the EXCI subreason field-1, and a system dump is taken.

User Response

For an explanation of the IRP return codes, see the interregion control blocks in the *CICS Data Areas* manual. The IRP return codes are in the DFHIRSPS copybook, listed under the heading IRC. Use the dump and IRP return codes to determine why the disconnect failed. You may also want to use the EXCI trace to determine the earlier error that caused the open pipe recovery routine to be invoked.

Note: The pipe is now in a "must close" state and if further calls are to be issued, the pipe must be closed and reopened again first.

622: ESTAE_SETUP_FAILURE

In order to protect itself from possible program checks the external CICS interface establishes an MVS ESTAE. In this case, the MVS ESTAE macro has failed.

Explanation

System Action

The call terminated, and the return code from the MVS ESTAE command is returned in the EXCI subreason field-1. This error may occur before EXCI dump services are initialized, therefore an EXCI issues an MVS abend (U0402) to force a SYSMDUMP.

User Response

Use the return code and the dump to determine why the ESTAE command failed. This may be an internal EXCI error and if the problem persists, contact your IBM support center.

623: ESTAE_INVOKED

A program check is encountered during call processing, and the ESTAE is invoked.

Explanation

System Action

The program check is handled by the EXCI ESTAE and an attempt is made to recover to a state that can support further EXCI calls. The MVS abend code is returned in the EXCI subreason field-1 of the return area. To aid further diagnosis, a SYSMDUMP is taken.

User Response

Use the return code and the dump to determine why a program check occurred in the external CICS interface. The most likely reason for this is that the EXCI code abended while trying to access the client program's parameters. Use the EXCI trace to determine if any of the parameters might have caused this error. If this is not

the case, this may be an error in the external CICS interface. Keep the dump and contact your IBM support center.

624: SERVER_TIMEDOUT

A DPL request has been issued and the target server program has executed in the CICS server region. However, the server program has been executing for longer than the time-out value specified in the DFHXCOPT table.

Explanation

System Action

The external CICS interface stops waiting for the server program to complete. Because the server program might complete some time after the time-out, and try to respond to the DPL call, the pipe is forced into a "must close" state.

User Response

Determine why the server application program timed out. Either there is a problem with the server program itself (for example, it might be in a loop), or the timeout value is too low.

625: STIMER_SETUP_FAILURE

In order to provide a TIMEOUT mechanism, the external CICS interface issues an MVS STIMERM macro call. This call has failed.

Explanation

System Action

The return code from the call is returned in the subreason field-1 of the EXCI return area. The DPL request is terminated and the external CICS interface takes a system dump. The pipe is placed in a "must close" state.

User Response

Use the MVS return code and the dump to determine why the call failed. This could be an external CICS interface error. Contact your IBM support center with details of the dump.

Note: The pipe is in a "must close" state after this error, and before attempting further calls must first be closed and reopened.

626: STIMER_CANCEL_FAILURE

On successful completion of a DPL request, the cancel of an STIMERM request issued to check the TIMEOUT value has failed with an error.

Explanation

System Action

The return code from the STIMERM CANCEL is returned in the subreason field-1 of the EXCI return area. The pipe is placed in a "must close" state, and the external CICS interface takes a system dump.

User Response

Use the return code and the dump to determine why the MVS STIMERM CANCEL command failed. This could be an external CICS interface error. Contact your IBM support center with details of the dump.

Note: The pipe is in a "must close" state after this error, and before attempting further calls must first be closed and reopened.

627: INCORRECT_SVC_LEVEL

The release level of the CICS SVC (DFHCSVC) is not the same (or higher) than the release level of the external CICS interface.

Explanation

System Action

The Initialize_User request is terminated. This error occurs before the external CICS interface SDUMP facilities are initialized, therefore EXCI issues an MVS abend (U0407) to force a SYSMDUMP.

User Response

Determine the level of the CICS SVC being used and ensure it is the same release level as the external CICS interface, or higher. If the SVC number is allowed to default (CICSSVC=0 in DFHXCOPT), the SVC number being used is the SVC first used by a CICS region on the MVS image. That is, the SVC used by the first CICS region to open the CICS interregion communications (IRC). If the SVC number is specified on CICSSVC in DFHXCOPT, the SVC number specified is at an incorrect level. For more information, see the description of the CICSSVC parameter in Chapter 15, "Using the EXCI options table, DFHXCOPT," on page 169.

628: IRP_LEVEL_CHECK_FAILURE

The release level of the module DFHIRP is not at the same, or higher, level than the release level of the external CICS interface.

Explanation

System Action

The Allocate_pipe request is terminated. The IRP return code (R15) is returned in the EXCI subreason field-1, and the function level of DFHIRP being used is returned in the EXCI subreason field-2. Subreason field-2 is only meaningful if subreason field-1 is zero. The external CICS interface takes a system dump.

User Response

Check the level of the DFHIRP module installed in the LPA. Ensure that it is at least the same as the external CICS interface. The installed level of DFHIRP must be the highest level of CICS or external CICS interface in use in the MVS image. For more details about installing DFHIRP, see the .

629: SERVER_PROTOCOL_ERROR

A response to a DPL request has been returned by CICS but the external CICS interface does not understand the response.

Explanation

System Action

The DPL request is terminated and the external CICS interface takes a system dump.

User Response

Use the dump to determine why the response was in error. The most likely reason for this is that the CICS application server program was not running under the control of a CICS mirror task. This can happen if the transaction definition named by the transid parameter on the DPL call does not specify DFHMIRS as the program name. This would cause unidentified responses being sent from the CICS server region.

630: RRMS_ERROR

An unexpected return code was received from Recoverable Resource Management Services (RRMS) while processing a DPL_Request.

Explanation

System Action

DPL_Request processing is terminated.

The value in subreason field-1 of the return area indicates which RRMS interface returned the unexpected return code:

1   CTXRCC

2   ATRRURD

3   CTXSDTA

The return code from the RRMS request is returned in subreason field-2.

The external CICS interface issues message DFHEX0002, and takes a system dump.

User Response

Use the return code from the RRMS request and the dump, to determine why the request failed. This may be an internal EXCI error or a problem with RRMS and you may need the assistance of your IBM support center.

631: RRMS_SEVERE_ERROR

During the processing of a DPL_Request, the EXCI code encountered an unexpected error while using its interface with Recoverable Resource Management Services (RRMS).

Explanation

System Action

DPL_Request processing is terminated.

The external CICS interface issues message DFHEX0002, and takes a system dump.

User Response

Use the dump, to determine why the request failed. This may be an internal EXCI error and you may need the assistance of your IBM support center.

632: XCGUR_GETMAIN_ERROR

During DPL_Request processing, a GETMAIN request for working storage for module DFHXCGUR failed.

Explanation

System Action

DPL_Request processing is terminated.

The return code from the GETMAIN is returned in subreason field-1 of the return area. The external CICS interface issues message DFHEX0003, and takes a system dump.

User Response

Use the return code from the GETMAIN, and the dump, to determine why the GETMAIN failed. A possible reason is that the region size of the job is too small. If this is the case, increase the region size and restart the client application program.

Chapter 20. Messages and codes

This section lists messages and abend codes for the external CICS interface.

DFHEX0001: An abend (code *aaa/bbbb*) has occurred in module *modname*.

An unexpected program check or abend *aaa/bbbb* has occurred in module *modname*. This implies that there may be an error in external CICS interface code.

Explanation

Alternatively, unexpected data has been passed on an external CICS interface call or storage has been overwritten.

The code *aaa/bbbb* is, if applicable, a 3-digit hexadecimal MVS system completion code *aaa* (for example, 0C1 or D37). If an MVS code is not applicable, this field is filled with three hyphens. The 4-digit code *bbbb*, which follows *aaa* is, if applicable, a user abend code produced by the external CICS interface. If the user abend code is not applicable, this field is filled with four hyphens.

System Action

An exception entry is made in the external CICS interface internal trace table, and to the GTF trace data set (if GTF is active), and a SYSMDUMP is taken.

The external CICS interface terminates the current request, and attempts to recover to a consistent state so that further EXCI requests can be serviced. For an application using the EXCI CALL API, a response of EXCI_SYSTEM_ERROR with a REASON of ESTAE_INVOKED is returned to the application. For an application using the EXCI EXEC API, an EXEC_RESP of LINKERR is returned to the application, together with an EXEC_RESP2 of ESTAE_INVOKED or EXEC_ESTAE_INVOKED, depending on whether the call level ESTAE routine, or the EXEC level ESTAE routine was invoked.

User Response

Look up the MVS code *aaa*, if there is one, in the relevant MVS codes manual which is detailed in the book list in the front of this manual.

If applicable, see the description of abend code *bbbb* for further guidance.

You may need further assistance from IBM to resolve this problem. See the *CICS Problem Determination Guide* for guidance on how to proceed.

Destination

Console

Module

DFHXCPRH, DFHXCEIP

DFHEX0002: A severe error (code *X'code'*) has occurred in module *modname*.

An error has been detected in module *modname*. The code *code* is the exception trace point ID which uniquely identifies what the error is and where the error was detected.

Explanation

System Action

An exception entry is made in the EXCI internal trace table and to GTF if it is active, (*X'code'* in the message). A system dump is taken.

This is a critical error and the EXCI request is terminated. The external CICS interface attempts to recover to a consistent state so that further EXCI requests can be issued. For applications using the EXCI CALL API, the EXCI_REASON returned to the application indicates the reason for the error. For applications using the EXCI EXEC API, the reason is returned in the EXEC_RESP2 field of the RETCODE area.

User Response

This failure indicates a serious error in the external CICS interface code. For further information about the EXCI exception trace entries, refer to the *CICS Problem Determination Guide*.

Destination

Console

Module

DFHXCPRH, DFHXCEIP

DFHEX0003: A GETMAIN request in module *modname* (code *X'code'*) has failed. Reason *X'rc'*.

An MVS GETMAIN was issued by module *modname*, but it failed with return code *rc*.

Explanation

The code *code* is the exception trace point ID which uniquely identifies the place where the MVS GETMAIN was issued.

System Action

An exception entry is made in the EXCI internal trace table (code *code* in the message). This is a critical error and the EXCI request is terminated. The external CICS interface attempts to recover to a consistent state so that further EXCI requests can be issued.

For applications using the EXCI CALL API, the EXCI_REASON returned to the application indicates the point of failure.

For applications using the EXCI EXEC API, the point of failure is returned in the
EXEC_RESP2 field of the RETCODE area.

For EXCI_REASON and EXCI_RESP of 603, the EXCI module DFHXCPRH also
issues abend 0410 which drives the ESTAE exit. Message DFHEX0001 is issued and
a SYSMDUMP is taken

User Response

Look up the MVS GETMAIN return code *rc* in the relevant MVS codes manual.

If the reason is insufficient storage, try increasing the size of the region for the
batch EXCI job.

You may need further assistance from IBM to resolve this problem. See the *CICS
Problem Determination Guide* for guidance on how to proceed.

Destination

Console

Module

DFHXCPRH, DFHXCTRI

DFHEX0004: JOBNAME: *jobname*, STEPNAME: *stepname*, PROCNAME: *procname*, SYSID IN SMF: *sysid*, APPLID: *applid*.

This message accompanies message DFHEX0001 and provides the jobname,
stepname, procname, Sysid in SMF and applid to which the EXCI job is
connecting.

Explanation

If an insert value is unknown or not specified, then the message insert will show
Unknown. For example, procname and stepname are not mandatory in an EXCI
job. If they are omitted and DFHEX0004 is issued, then the inserts for procname
and stepname show Unknown.

System Action

Follow system action for DFHEX0001.

User Response

Follow user response for DFHEX0001.

Destination

Console

Module

DFHXCPRH, DFHXCEIP

DFHEX0100: The installed level of CICS SVC does not support the EXCI call.

The external CICS interface module DFHXCPRH detected that the level of CICS SVC (DFHCSVC) in use does not support the external CICS interface.

Explanation

System Action

The EXCI request is terminated. An exception trace is made in the EXCI internal trace table, and if GTF is active, in the GTF trace data set. The external CICS interface module DFHXCPRH issues abend 0407 which drives the ESTAE exit. Message DFHEX0001 is issued, and a SYSMDUMP is taken.

User Response

Check the level of DFHCSVC installed in the LPA, which Generally, the latest level of DFHCSVC must be used when running CICS and the external CICS interface. For more information about installing DFHCSVC see Installing the CICS Type 3 SVC, in the *CICS Transaction Server for z/OS Installation Guide*.

Destination

Console

Module

DFHXCPRH

DFHEX0101: Unable to start interregion communication because DFHIRP services are down level.

The version of DFHIRP being used is at a lower level than that of the External CICS Interface (EXCI) module DFHXCPRH.

Explanation

System Action

The EXCI allocate pipe request is rejected, and a return code passed back to the batch application.

User Response

Update the level of the DFHIRP module in the LPA such that it matches the level of the latest CICS version in use.

Destination

Console

Module

DFHXCPRH

DFHEX0110: EXCI SDUMP has been taken. Dumpcode: *dumpcode*, Dumpid: *dumpid*.

This message is issued on successful completion of a MVS SDUMP issued by external CICS interface module DFHXCDMP. An error, signalled by a previous message, caused a call to be made to DFHXCDMP to take a system dump.

Explanation

The dump code *dumpcode* is an 8-character system dump code identifying the external CICS interface problem. A system dump code is the EXCI message number with the DFH prefix removed.

dumpid is the unique 9-character string identifying this dump.

System Action

The EXCI request is terminated.

User Response

See the EXCI message indicated by *dumpcode* for further guidance.

You may need further assistance from IBM to resolve this problem. See the *CICS Problem Determination Guide* for guidance on how to proceed.

Destination

Console

Module

DFHXCDMP.

DFHEX0111: EXCI SDUMP attempted but SDUMP is busy - will retry every five seconds for *nnnn* seconds.

At the time of the MVS SDUMP request issued by DFHXCDMP another address space in the same MVS system was in the process of taking an SDUMP.

Explanation

This causes MVS to reject the new request. A nonzero value for the dump retry parameter in the DFHXCOPT table means that the external CICS interface waits five seconds before retrying the SDUMP request. If necessary, the external CICS interface retries every five seconds for the total time specified on the retry parameter.

System Action

The external CICS interface issues an MVS STIMERM macro which causes it to wait for five seconds. The request is reissued when the delay interval has expired.

User Response

None.

Destination

Console

Module

DFHXCDMP.

DFHEX0112: SDUMP request failed - *reason X'nn'*.

Explanation

An MVS SDUMP request issued from the external CICS interface has failed to complete successfully. The possible reasons, (*reason*) for the failure are as follows:

ONLY PARTIAL DUMP
> The SYS1.DUMP data set to which the dump is written is not large enough to contain all of the dumped storage.

SDUMP BUSY
> At the time of the MVS SDUMP request issued by the EXCI, another address space in the same MVS system was in the process of taking an SDUMP. This causes MVS to reject the new request. If a nonzero value is specified for the dump retry parameter in DFHXOPTS table, the EXCI has retried the SDUMP request every five seconds for the specified period. This message is only issued if SDUMP is still busy after the final retry.

STIMERM FAILED
> In order to delay for five seconds before retrying SDUMP after an SDUMP BUSY condition, the EXCI issues an MVS STIMERM macro request. MVS has indicated that the STIMERM request has failed.

NO DATA SET AVAILABLE
> No SYS1.DUMP data sets were available at the time the SDUMP request was issued.

REJECTED BY MVS, REASON = X'nn'
> MVS has rejected the SDUMP request because of user action (for example, specifying DUMP=NO in the MVS IPL) or because of an I/O error or terminating error in the SDUMP routine. X'nn' is the SDUMP reason code.

NOT AUTHORIZED FOR EXCI
> SDUMP is not authorized for the external CICS interface.

INSUFFICIENT STORAGE
> The EXCI issued an MVS GETMAIN for subpool 253 storage during the processing of the SDUMP request. The GETMAIN has been rejected by MVS.

System Action

The EXCI proceeds as if the dump had been successful.

User Response

The user response depends on the reasons, (*reason*), for the failure.

ONLY PARTIAL DUMP
> Increase the size of the SYS1.DUMP data sets and cause the SDUMP request to be reissued.

SDUMP BUSY
> Cause the SDUMP to be reissued after, if appropriate, increasing the dump retry time in DFHXCOPT.

STIMERM FAILED
> Use MVS problem determination methods to fix the STIMERM failure and then cause the SDUMP request to be reissued.

NO DATA SET AVAILABLE
> Clear a SYS1.DUMP data set and then cause the SDUMP request to be reissued.

REJECTED BY MVS, REASON = X'nn'
> No action is required if the dump is suppressed deliberately. If the dump has failed because of an error in the MVS SDUMP routine, use MVS problem determination methods to fix the error and then cause the SDUMP request to be reissued. See the *z/OS MVS Programming: Assembler Services Reference*, for an explanation of the SDUMP reason code X'nn'.

NOT AUTHORIZED FOR EXCI
> This reason is unlikely because SDUMP is unconditionally authorized during EXCI initialization, and should be authorized throughout the EXCI run. If you do get this reason, the EXCI AFCB (authorized function control block) has probably been accidentally overwritten.

INSUFFICIENT STORAGE
> Ensure sufficient storage is available to MVS for subpool 253 requests.

Destination

Console

Module

DFHXCDMP

DFHEX0113: EXCI trace Initialization has failed.

An attempt to initialize external CICS interface (EXCI) trace facilities during EXCI initialization has failed.

Explanation

System Action

The EXCI request continues without trace facilities. An earlier message identifies the cause of the failure.

User Response

Refer to the earlier message to determine the cause of the failure.

Destination

Console

Module

DFHXCTRI

DFHEX0114: Incorrect data has been passed for EXCI tracing causing a program check in DFHXCTRP.

Some data passed to the external CICS interface (EXCI) trace module DFHXCTRP for addition to the EXCI internal trace table, or GTF trace, caused a program check to occur when an attempt was made to access it.

Explanation

The most likely cause of this error is incorrect data passed on an EXCI CALL API request that the trace program DFHXCTRP is attempting to access.

System Action

The EXCI request is terminated and a SYSMDUMP is taken.

User Response

Examine the dump to determine the source of the incorrect data.

You may need further assistance from IBM to resolve this problem. See the *CICS Problem Determination Guide* for guidance on how to proceed.

Destination

Console

Module

DFHXCTRI

DFHEX0115: EXCI trace services have been disabled due to a previous error.

An error occurred in the external CICS interface (EXCI) trace module DFHXCTRP indicated by message DFHEX0001. In trying to recover from the error, module DFHXCTRI determined that the error was not caused by accessing incorrect data passed to DFHXCTRP, but was due to a program check in DFHXCTRP.

Explanation

System Action

The EXCI trace facilities are disabled to prevent further errors. A SYSMDUMP is taken.

User Response

See the DFHEX0001 message and the SYSMDUMP to determine the cause of the error.

You may need further assistance from IBM to resolve this problem. See the *CICS Problem Determination Guide* for guidance on how to proceed.

Destination

Console

Module

DFHXCTRI

DFHEX0116: Program check occurred within global trap exit - DFHXCTRA now marked unusable.

After making a trace entry, the external CICS interface (EXCI) trace program DFHXCTRP called the EXCI field engineering global trap program DFHXCTRA. A program check occurred during execution of DFHXCTRA.

Explanation

System Action

The EXCI marks the currently active version of DFHXCTRA as unusable and ignores it on subsequent calls to DFHXCTRP for all subsequent calls made under this TCB. The EXCI request is terminated, and a SYSMDUMP is taken.

User Response

Use the dump to find the cause of the program check.

You may need further assistance from IBM to resolve this problem. See the *CICS Problem Determination Guide* for guidance on how to proceed.

You should use the global trap exit only in consultation with an IBM support representative.

Destination

Console

Module

DFHXCTRI

Part 4. CICS ONC RPC support

Client applications can use the Open Network Computing Remote Procedure Call (ONC RPC) format to access CICS programs as remote procedures.

This part contains:

Chapter 21. Introduction to ONC RPC

CICS ONC RPC allows client applications to access CICS programs by calling them as remote procedures using the ONC RPC format.

CICS ONC RPC can be used:

- To allow clients to use existing CICS programs and the transaction processing services they provide
- To allow clients to use newly created CICS programs

TCP/IP for MVS is a prerequisite for CICS ONC RPC; it provides the library code for Sun Microsystems' ONC RPC Version 3.9. Hence, CICS ONC RPC servers work with any remote client compatible with ONC RPC Version 3.9, regardless of operating system or machine type. See the *z/OS Communications Server: Programmer's Reference* for information about the function of ONC RPC Version 3.9 supported by TCP/IP for MVS .

Figure 29 shows how CICS ONC RPC allows a variety of client applications to communicate with CICS programs using ONC RPC.
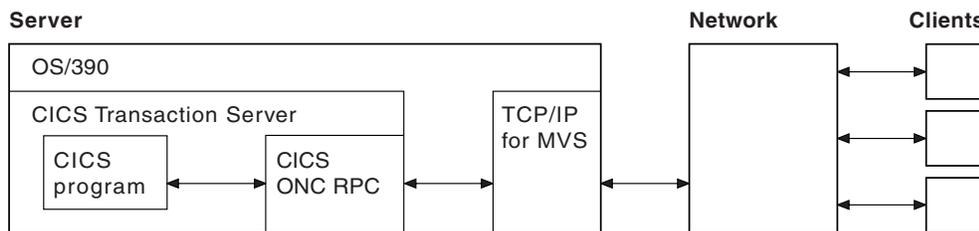
Figure 29. How CICS ONC RPC might be used

The CICS program called to service a client request is executed by a transaction that has no principal facility. It is therefore not allowed to use some commands of the CICS application programming interface:
- Terminal control commands that reference the principal facility
- Options of EXEC CICS ASSIGN that return terminal attributes
- BMS commands
- Sign-on and sign-off commands.

The rest of this chapter describes:
- "ONC RPC concepts"
- "ONC RPC facilities" on page 257
- "ONC RPC naming and routing" on page 259

ONC RPC concepts

This section introduces the basics of ONC RPC operation, its place in TCP/IP networks, and how its main facilities work. It does not cover all aspects of ONC RPC or TCP/IP, only those that relate to CICS ONC RPC.

CICS ONC RPC: In the rest of this information, notes like this point out how CICS ONC RPC implements the area of ONC RPC being described in the text.

RPC

When a process invokes or calls a process on a remote system, that call is a *remote procedure call (RPC)*.

The calling process is a *client* (that is, a process requesting a service); the remote process is a *server* (a process offering a service). As shown in Figure 30, the client sends a request for a procedure to be run, and supplies parameters for that particular run. Once the server has run the procedure, it returns the reply.

Remote procedure   **Client application**

RPC request and parameters

reply returned
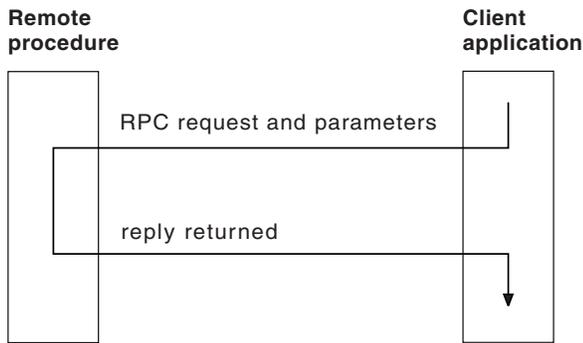
Figure 30. Basic RPC operation

In the RPC model, there is no provision for coordinating changes to recoverable resources in different servers, nor for coordinating changes to recoverable resources in successive calls to the same server. Committing changes to recoverable resources is under the control of the remote procedure, not the client application.

Several RPC implementations have been developed and are now available on a variety of systems. RPC allows a programmer to network an application by distributing the procedures that make up the application across different processors. This is done without the programmer becoming involved with the details of the communication interface required to transmit the parameters to and from the remote procedures.

ONC

ONC is Open Network Computing, a range of software developed by Sun Microsystems.

ONC is Open Network Computing, a range of software developed by Sun Microsystems. As well as the ONC RPC routines, Sun provides XDR (eXternal Data Representation) routines, which are used for data conversion. The ONC RPC and XDR protocols and formats are supported on many different platforms.

CICS ONC RPC: CICS ONC RPC allows users to run only ONC RPC servers under CICS hosts. It does not support client applications running under CICS.

TCP/IP

ONC RPC applications use the TCP/IP family of protocols.

See "TCP/IP protocols" on page 8 for more information about TCP/IP.

ONC RPC facilities

The ONC RPC implementation consists of XDR routines the RPCGEN compiler, and the ONC RPC API library.

XDR routines

Data exchanged between systems engaged in ONC RPC must always flow in a standard format specified by XDR, because different machine architectures have different representations of the same information.

Both client and server use *XDR routines* to convert the input and output parameters between XDR format and the local data format. You either write these yourself, or specify an XDR library function, as described below. In Figure 31, **inproc** and **outproc** are the XDR routines.
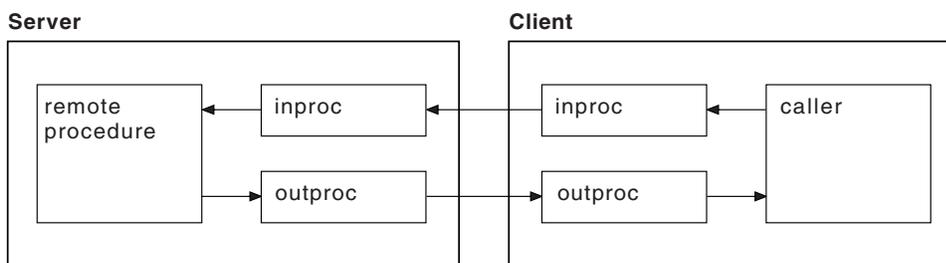
Figure 31. XDR routines used in a remote procedure call

Notice that in Figure 31, the same XDR routine, **inproc**, is used to encode and decode the data as it flows from client to server, and similarly for **outproc** as it flows back to the client. The source for **inproc** is the same in the client and server, but XDR library functions in the routines are compiled to encode or decode as appropriate. Such routines are termed *bidirectional*, and they help to ensure that the encoding and decoding is done symmetrically in the two routines.

Using XDR library functions

XDR library functions are a set of C functions supplied with ONC RPC, which application programmers can use when writing XDR routines.

They can be used as follows, depending on the complexity of the structure pointed to by the call argument and reply parameters.

For parameters that are simple single-field C data types
Use an XDR library function for **inproc** and **outproc**.

For parameters that are C data type vectors, arrays, strings, and so on
Use an XDR library function for **inproc** and **outproc**.

For more complex structures
Write an XDR routine, using XDR library functions as required. Alternatively, use the RPCGEN compiler, described in "RPCGEN compiler" on page 258, to create an XDR routine from an XDR data description.

CICS ONC RPC: CICS ONC RPC supports the use of the XDR library functions that support data conversion.

RPCGEN compiler

To use RPCGEN, you write a program definition in RPCL, a language similar to a subset of C, designed for the definition of ONC RPC distributed programs.

To use RPCGEN, you write a program definition in RPCL, a language similar to a subset of C, designed for the definition of ONC RPC distributed programs. The definition defines the data to be transferred and procedures to be used for both client and server. The client application source program is written as though the remote procedure call were a call to a local program. The code to send the call and get the reply are part of the client stub, which is generated by RPCGEN. Similarly the code the server needs to accept the call and send back the reply are part of the server stub, which is also generated by RPCGEN. Figure 32 illustrates the role of RPCGEN in application development.
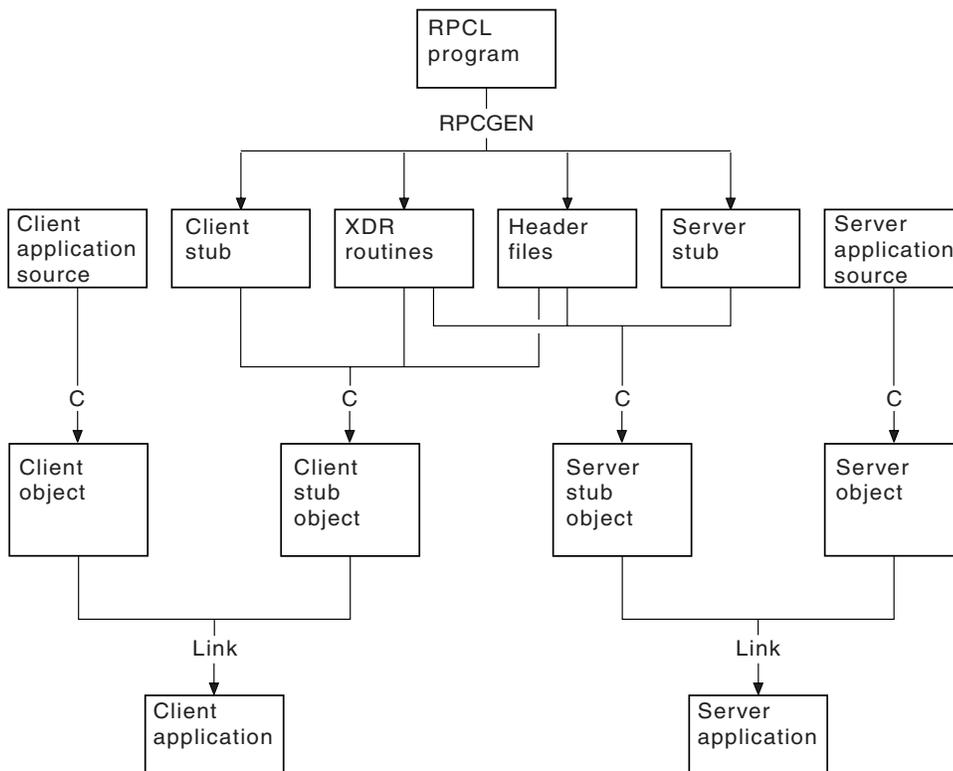
Figure 32. Using the RPCGEN compiler

CICS ONC RPC: RPCGEN may only be used for:
* Generating pairs of XDR routines, as described in the previous section
* Generating a client stub to be linked with the application for the client system
* Generating header files

CICS ONC RPC does not use the server stub generated by RPCGEN.

ONC RPC API library

The ONC RPC API library contains two types of call: high level and low level.

The high-level ONC RPC API can be used only with UDP. It enables users to make remote procedure calls very and with a minimum of library calls, but at a cost of some restriction in available function. The main function of the API is provided by three calls:

registerrpc
> Used in the server to register a procedure to be called as a remote procedure by clients.

svc_run
> Used in the server to see if a request has arrived from a client.

callrpc
> Used in the client to make a remote procedure call.

The low-level ONC RPC API contains many more calls, which give more control and flexibility. For example:

- Low-level calls give the user the choice of transport below ONC RPC, including TCP or UDP.
- With low-level calls, user-written network registration services other than the Portmapper (the Portmapper is described below) can be used.
- Low-level calls allow the variation of ONC RPC time-outs and retry values.
- Low-level calls allow standard ONC RPC authorization to be applied. Only UNIX authorization is available in ONC RPC Version 3.9.

CICS ONC RPC: CICS ONC RPC provides all the server function. You don't specify any server RPC calls.

The client can make its request with the high-level call **callrpc**, or can use low-level calls. CICS ONC RPC is implemented using low-level ONC RPC calls. The implementation allows concurrent dispatching of individual procedures and allows TCP to be supported as well as UDP.

ONC RPC naming and routing

Remote procedures in ONC RPC are identified by the 3-tuple: program number, version number, and procedure number.

It is usual to package several related procedures together into a single program. When changes are made to the procedures, a new version of the program is created, but the new version usually contains the same procedure numbers as the previous version.

Procedure zero

Users define procedure numbers for each program, conventionally starting at 1 and proceeding in sequence.

Procedure 0 is usually defined as a procedure with no parameters and no processing that returns an empty reply. This is useful for clients, who can call procedure 0 to see if a particular service exists and to test performance on a null call.

Registration and the Portmapper

Servers on a host need to let clients know their logical addresses and which services they offer.

In ONC RPC, servers generally do this by *registering* with a utility service called the *Portmapper*. This maintains a list of mappings from program/version numbers (also qualified by protocol used) to TCP/IP port numbers on a host.

The Portmapper itself can always be located by clients because it is always on well-known port 111 on a given host. If using low-level calls, the client first asks the Portmapper for the port number for the particular remote procedure, and then calls that port directly. The high-level call, **callrpc**, performs the same function transparently to the user.

CICS ONC RPC: Registration is done by CICS ONC RPC automatically, or under operator control.

Routing

Before calling a procedure, a client asks the Portmapper at the host for the port number of the program and version that the client wants to call.

Before calling a procedure, a client asks the Portmapper at the host for the port number of the program and version that the client wants to call. (The protocol is determined when the connection between TCP/IP systems is set up.) In the remote procedure call, the client supplies only the IP address, port number, and procedure number. Figure 33 shows how the IP address, port number, and procedure number identify the server procedure.
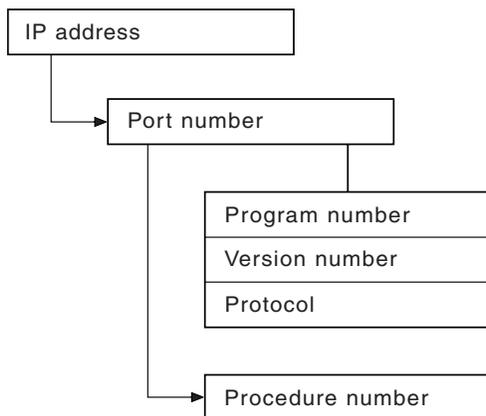
Figure 33. TCP/IP and RPC routing

Types of remote procedure call

These are the five types of remote procedure call.

Synchronous
> This is the normal method of operation. The client makes a call and does not continue until the server returns the reply.

Nonblocking
> The client makes a call and continues with its own processing. The server does not reply.

Batching
> This is a facility for sending several client nonblocking calls in one batch.

Broadcast RPC
> RPC clients have a broadcast facility, that is, they can send messages to many servers and then receive all the consequent replies.

Callback RPC

The client makes a nonblocking client/server call, and the server signals completion by calling a procedure associated with the client.

CICS ONC RPC: CICS ONC RPC cannot support callback RPC, because callback requires that both ends contain both client and server procedures.

Chapter 22. CICS ONC RPC concepts

This section describes the CICS ONC RPC components and control flow.

It describes:
- "ONC RPC remote procedures and CICS programs"
- "CICS ONC RPC transactions" on page 264
- "CICS ONC RPC user-replaceable programs" on page 265
- "CICS ONC RPC control flow" on page 266
- "CICS ONC RPC data flow" on page 268

ONC RPC remote procedures and CICS programs

In CICS ONC RPC, the CICS programs are identified by a 4-tuple.
- Program number—same as the ONC RPC program number
- Version number—same as the ONC RPC version number
- Procedure number—same as the ONC RPC procedure number
- Protocol—determined by the protocol used to communicate between the client system and z/OS Communications Server.

When a client request arrives, the CICS program chosen to service it is the one associated with the 4-tuple just described. Figure 34 shows a state of CICS ONC RPC in which five 4-tuples are associated with three CICS programs.

4-tuples   **CICS programs**

24127AC0	5	1	U

PROGA

24127AC0	5	1	T

CE00457F	3	1	U

PROGB

CE00457F	3	1	T

PROGC

CE00457F	3	2	T
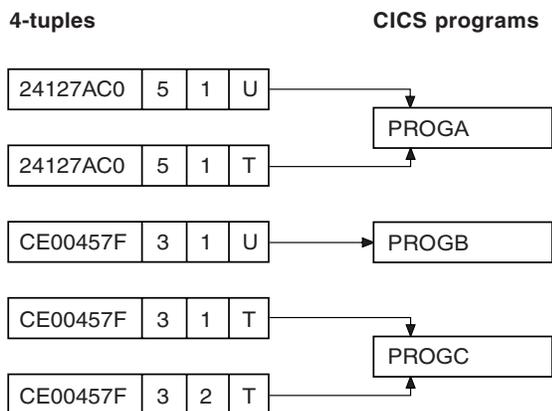
Figure 34. Remote procedures and CICS programs

The program numbers are given in hexadecimal. The protocols are U for UDP and T for TCP.
- If a client request arrives for program 24127AC0, version 5, procedure 1, the CICS program PROGA is used to service it whether the protocol is TCP or UDP.
- If a request arrives for program CE00457F, version 3, procedure 1, and the protocol is UDP, the CICS program PROGB is used to service it. But if the same request arrives and the protocol is TCP, PROGC is used to service it.

  It is, however, usual to use the same program, version, and procedure irrespective of the protocol used to transmit the request.
- The CICS program PROGC is also used for procedure 2 of the same program and version if the protocol is TCP.

    263

How you set up and control the relationship between 4-tuples and CICS programs is described in Chapter 24, "Configuring CICS ONC RPC using the connection manager," on page 279.

Where the CICS program might be

The CICS program might be in one of three places.

- In the same CICS region as CICS ONC RPC
- In a different CICS region on the same host
- On a different host that supports CICS and inbound DPL

The CICS programs can reside on any CICS system accessible with DPL from the CICS region running CICS ONC RPC. DPL operation is described in the Overview of DPL.

CICS ONC RPC transactions

Three CICS transactions are supplied with CICS ONC RPC: connection manager, server controller, and alias.

Connection manager (CRPC)

The connection manager is a transaction that allows you to enable and disable CICS ONC RPC, and configure and inquire on it.

You run the connection manager transaction as required, and several instances of it can be active at the same time. The connection manager is described in Chapter 24, "Configuring CICS ONC RPC using the connection manager," on page 279.

Server controller (CRPM)

The server controller monitors the z/OS Communications Server interface for client requests, and starts instances of the alias transaction, using EXEC CICS START, to service them.

The server controller is a transaction of long duration. It is started by the connection manager when CICS ONC RPC is enabled, and stopped when CICS ONC RPC is disabled. Only one instance of the server controller can be active in a CICS system.

Alias (CRPA)

CICS ONC RPC supplies one alias *program*. Multiple instances of the alias *transaction* can be run in parallel, each in response to a client request.

An alias is started by the server controller for each client request that arrives to be processed, as shown in Figure 35 on page 265. This allows CICS ONC RPC to process many client requests concurrently.

The alias program uses EXEC CICS LINK to transfer control to the CICS program.

```
Server          ◄─── Client requests
controller      ◄─── from
                ◄─── TCP/IP
        │   │   EXEC CICS START
        │   │
        ▼   ▼
    ┌───────┐
    │   ┌───────┐
    │   │   ┌───────┐
    ▲   │   │  Alias
    │   │   │  transactions
    │   │   │
    │   │   │  EXEC CICS LINK
    │   ▼   │
    │ ┌───────┐
    └─│   ┌───────┐
      │   │   ┌───────┐
      │   │   │  CICS
          │   │  programs
```
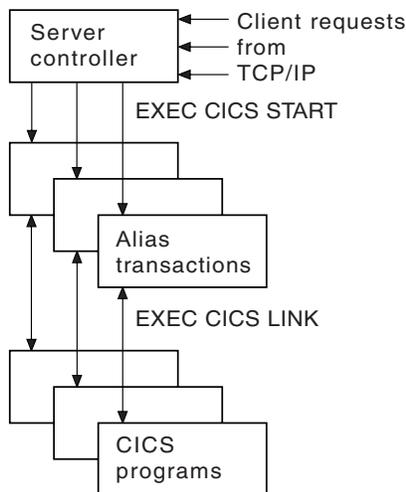
Figure 35. The server controller and alias transactions

CICS ONC RPC user-replaceable programs

Servicing a client request involves not only a CICS program, but a converter
program and XDR routines. For compatibility with earlier releases of CICS you can
use a resource checker program to validate incoming client requests, or you can
use CICS security facilities.

XDR routines

You need to provide one or two XDR routines for each 4-tuple. You always need
an inbound XDR routine, and unless the client call is nonblocking, you need an
outbound XDR routine as well.

XDR (eXternal Data Representation) is described in "XDR routines" on page 257.

The XDR routines for each 4-tuple are specified by using the connection manager.

Resource checker module

CICS ONC RPC provides an interface to a resource checker (which you write).

Converters

You can also supply a converter for each program-version-procedure-protocol
4-tuple.

Each converter can contain up to three functions.
* **Getlengths function**. The **Getlengths** function might be called by the connection
  manager when a 4-tuple is registered. **Getlengths** can supply the following
  information:
  - The length of the input and output data for the CICS program
  - Whether the output data overlays the input data in the communication area

  Because its processing is done before any client requests are received, It is
  appropriate to use **Getlengths** to provide the values of data lengths that do not

vary from call to call. Refer to "Lengths of the CICS program input and output data" on page 307 for a fuller description of when **Getlengths** should be used for this purpose.

- **Decode function**. The **Decode** function is called by the server controller on receipt of a client request. **Decode** can do the following:
  - Supply the length of the input and output data for the CICS program. If the parameter lengths vary from call to call, **Decode** should return them for the current call.
  - Reconstruct the data from the client as a communication area for the CICS program. "CICS ONC RPC data flow" on page 268 illustrates the kinds of data that **Decode** might have to handle. The incoming data might include pointers, and **Decode** must gather up the incoming data into the communication area.
  - Convert data structures from C format to the format appropriate to the programming language in which the CICS program is written.
  - Process data from the client that is not intended for the CICS program. For example the data from the client might include the name of the CICS program to be called, and **Decode** can feed this information back to the server controller.

- **Encode function**. The **Encode** function is called by the alias when the CICS program ends. **Encode** can do the following:
  - Reconstruct the data from the communication area into the form expected by the client. "CICS ONC RPC data flow" on page 268 illustrates the kinds of data that **Encode** might have to handle. The client might expect to receive data accessed by pointers, and **Encode** must build this structure from the data in the communication area.
  - Convert data structures from the format appropriate to the programming language in which the CICS program is written into C format.

Not all 4-tuples need a converter with all three functions. You use the connection manager to specify the converter and the use of **Getlengths**, **Decode**, and **Encode** for each 4-tuple.

The way that particular language data structures are stored is documented in the appropriate language manuals, and a correspondence between C data types and those in other languages is given in the *z/OS Language Environment Programming Guide*.

For detailed instructions on the writing of converters, refer to "Write the CICS ONC RPC converter" on page 307.

CICS ONC RPC control flow

This shows the components involved in processing a typical client ONC RPC request.
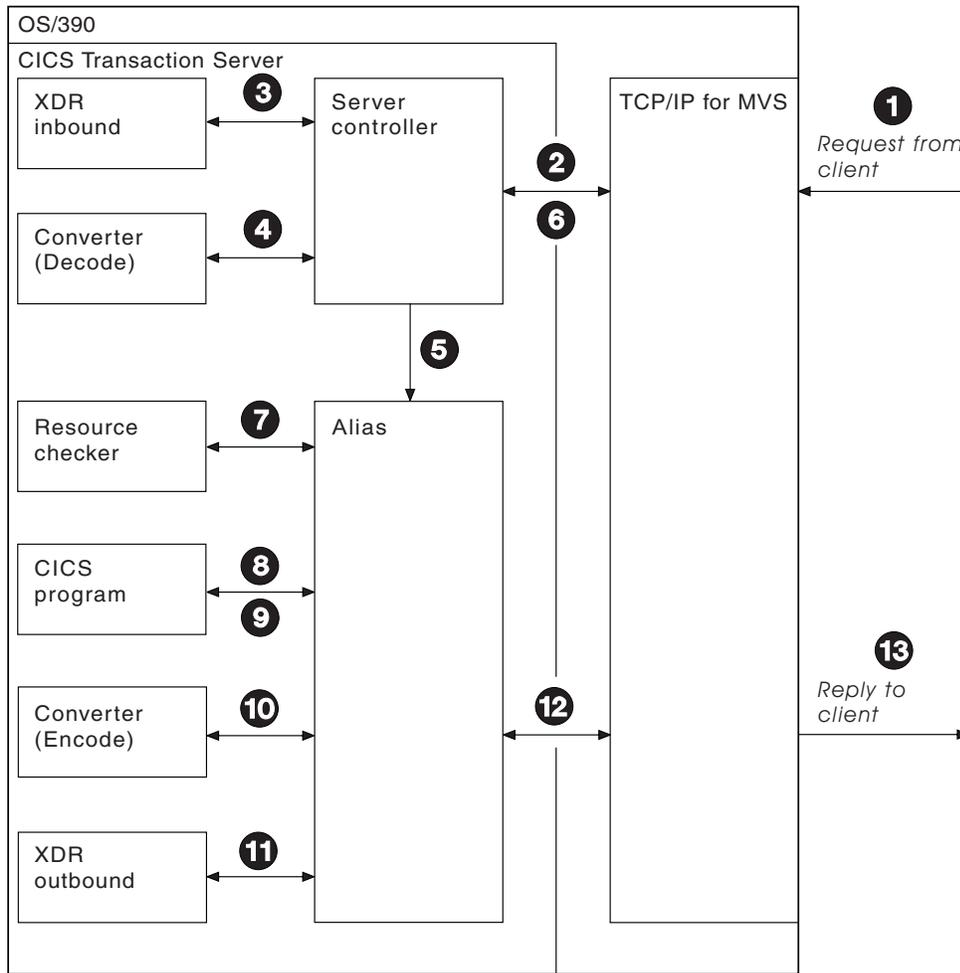
Figure 36. Call processing

Client requests are processed in the following steps:

1. A request from a client arrives in z/OS Communications Server.

2. The server controller monitors the z/OS Communications Server interface for incoming client requests, and the client request is passed to it. (From the 4-tuple for the request, the server controller can find the corresponding XDR routine and converter to call.)

3. The server controller invokes the inbound XDR routine.

4. The server controller calls the converter, requesting the **Decode** function, if it is required for the 4-tuple. If **Decode** is not required, the server controller allocates storage for the CICS program communication area.

5. The server controller then starts an alias to deal with all further processing of the request within CICS.

6. The server controller returns to monitor the z/OS Communications Server interface for client requests.

7. The alias optionally calls a user-written resource checker.

8. The alias issues an EXEC CICS LINK to the CICS program for the 4-tuple. The communication area set up by **Decode** is passed in the LINK command.

9. The CICS program processes the request and returns its output to the alias program in the communication area.

10. The alias calls the **Encode** function, if it is required for the 4-tuple.

11. The alias invokes the outbound XDR routine.
12. The alias returns the reply to z/OS Communications Server, and ends.
13. The reply is sent back to the client.

Updating recoverable resources

After **Decode** processing, the server controller uses **EXEC CICS SYNCPOINT** to commit any changes to recoverable resources that **Decode** might have made.

If the CICS program makes updates to recoverable resources, whether the changes are committed or backed out depends on the location of the CICS program, and on whether it uses the **EXEC CICS SYNCPOINT** command.

- If the CICS program is in a CICS region different from the one in which CICS ONC RPC is operating, the updates are committed when the CICS program returns control to the alias.
- If the CICS program is in the same CICS region as CICS ONC RPC, and it uses **EXEC CICS SYNCPOINT**, the updates are committed when the syncpoint is processed.
- If the CICS program is in the same CICS region as CICS ONC RPC, but it does *not* use **EXEC CICS SYNCPOINT**, the updates are committed when the alias transaction ends normally, or are backed out when the alias transaction abends.

CICS ONC RPC data flow

This section describes data flow from a client to a CICS program, and from a CICS program back to the client.

From client to CICS program

This diagram shows the progress of data from the client to the CICS program during a remote procedure call.

Parameter

| ... int |
| pointer |
| ... double |
| etc |

Data in client

XDR in client (outbound)

Data as transmitted

XDR in host (inbound)

Parameter

| ... int |
| pointer |
| ... double |
| etc |

Data in host

Decode

Communication area
for CICS program
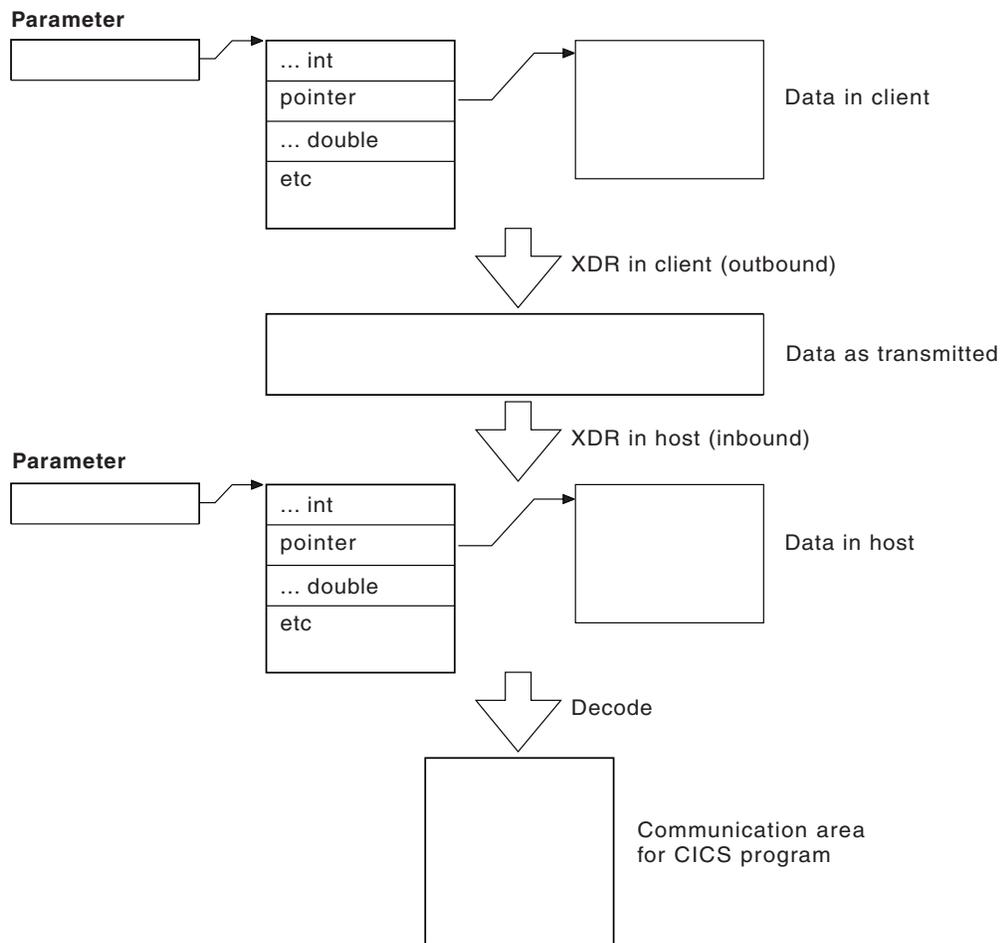
Figure 37. Data flow from client to CICS program

In this example the processing is as follows:

1. The client call has a parameter which includes a pointer to data that is to be passed to the CICS program. The client's outbound XDR routine packages the parameter and the indirect data for transmission to the host.

2. The data is transmitted over the network to the host.

3. In the host, the inbound XDR routine rebuilds the data as it was in the client.

4. The **Decode** function of the converter reorganizes the data into a communication area for the CICS program.

Data format in the CICS program communication area

If the call is a blocking call, the position in the CICS program's communication area of data to be returned to the client has to be specified. The data in the CICS program's communication area can be organized in two ways.

- Contiguous—the data to be returned to the client does not start at the beginning of the communication area, but at some offset into it.

- Overlaid—the data to be returned starts at the beginning of the communication area. The CICS program overwrites the inbound client data in this area with any data to be returned to the client.

Figure 38 on page 270 illustrates these two possibilities.

Contiguous

input output

Overlaid

input

output
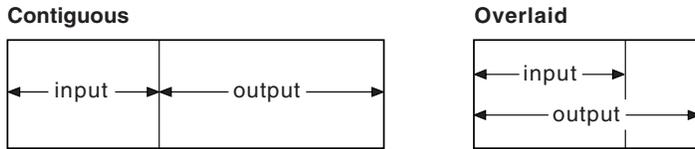
Figure 38. Use of communication area according to data format

From CICS program to client

This shows the progress of data from the CICS program back to the client.

Parameter

Communication area
from CICS program

Encode

Parameter

pointer

pointer

Data in host

XDR in host (outbound)

Data as transmitted

XDR in client (inbound)

Parameter

pointer

Data in client

pointer
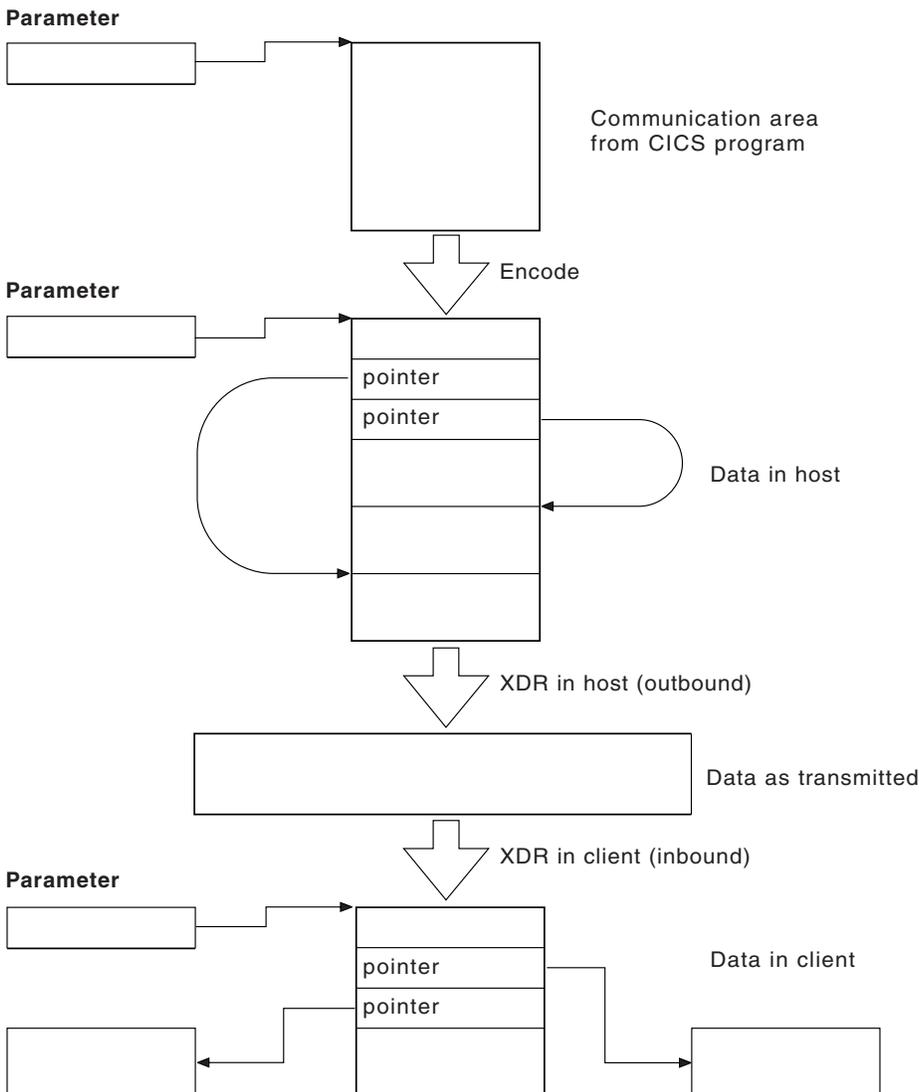
Figure 39. Data flow from CICS program to client

The processing is as follows:

1. The CICS program's output is in the communication area that was created by the **Decode** function. The **Encode** function reorganizes the data in the manner that the client expects. In this case the client is expecting to get back a structure

including two pointers to indirect data. The **Encode** function puts the data in a
single area of storage to simplify storage management processing when the
area is to be freed.

2. The outbound XDR routine packages the data for transmission.
3. The data is transmitted over the network to the client.
4. In the client, the inbound XDR routine rebuilds the data as it was in the host.

Chapter 23. Setting up CICS ONC RPC

Clients

Clients must access servers on CICS ONC RPC over a TCP/IP network.

Client systems must use a library compatible with the library for ONC RPC Version 3.9, as this is the ONC RPC version supported by TCP/IP for MVS (Versions 2.2.1 and 3.1). To communicate over a TCP/IP network, appropriate hardware and software must be in place.

MVS

The following items are prerequisite, that is, must be installed on the MVS system for CICS ONC RPC to run.

- TCP/IP for MVS Version 2.2.1 or above. TCP/IP for MVS ports must be made available for use by the CICS region involved.
- Language Environment. This provides the C runtime libraries that are a prerequisite for running CICS ONC RPC.
- If you are using RPCGEN, or writing your own XDR routines, you need a C compiler to compile RPCGEN output and your XDR routines.

CICS

CICS must be set up for Language Environment support, as described in the *CICS Transaction Server for z/OS Installation Guide* and in the *z/OS Language Environment Customization*.

Note: TCP/IP for MVS CICS Sockets is not a prerequisite for CICS ONC RPC.

TCP/IP for MVS

CICS ONC RPC and TCP/IP for MVS CICS Sockets Version 2.2.1 cannot operate together from one CICS region to one TCP/IP for MVS region. You are advised to run CICS Sockets and CICS ONC RPC in different CICS regions.

TCP/IP for MVS Version 3.1 users do not have this problem; CICS Sockets and CICS ONC RPC can both be run from the same CICS region.

TCP/IP for MVS 2.2.1

There are no prerequisites for running CICS ONC RPC.

Note: CICS ONC RPC and TCP/IP for MVS CICS Sockets Version 2.2.1 cannot operate together from one CICS region to one TCP/IP for MVS region. You are advised to run CICS Sockets and CICS ONC RPC from different CICS regions.

TCP/IP for MVS 3.1

The following PTF is a prerequisite for running CICS ONC RPC:

- A PTF, number UN79963, related to the use of the **xdr_text_char** XDR library function.

Note: CICS ONC RPC and TCP/IP for MVS CICS Sockets Version 2.2.1 cannot operate together from one CICS region to one TCP/IP for MVS region. You are advised to run CICS Sockets and CICS ONC RPC from different CICS regions.

Storage requirements

Except where otherwise noted, the storage used by CICS ONC RPC is obtained from CICS subpools.

When CICS ONC RPC is enabled, its storage requirements are as follows:
- 40 KB base storage
- 100 bytes for each registered 4-tuple.

For each client request being processed the following storage is required:
- MVS-controlled storage used by the inbound XDR routine for internal data structures
- Storage used by the inbound XDR routine for the data structure it builds for the **Decode** function
- Storage for the CICS program communication area
- Storage used by the alias transaction while running the CICS program
- Storage used by the **Encode** function to create a data structure for the outbound XDR routine
- MVS-controlled storage used by the outbound XDR routine

CICS ONC RPC setup tasks

There are tasks associated with the CICS ONC RPC data set, dump formatting, and a warning about migration.

Creating the CICS ONC RCP data set

JCL is provided in the DFHCOMDS job to create the CICS ONC RPC data set.

The data set is defined as a VSAM key-sequenced data set by a DEFINE CLUSTER command like the following:

```
DEFINE CLUSTER (                          -
  NAME( xxxxxxxx.CICSONC.RESOURCE ) -
  CYL ( 2 1  )                            -
  KEYS( 19 0 )                            -
  INDEXED                                 -
  VOLUME ( vvvvvv )                       -
  RECORDSIZE( 150 150 )                   -
  FREESPACE( 5 5 )                        -
  SHAREOPTIONS( 1 )                       -
  )
```

The job to define the data set must be run before you start the connection manager for the first time.

JCL entry for dump formatting

To switch dump formatting on for CICS ONC RPC (and for all running features), change the IPCS VERBEXIT control statement.

```
IPCS VERBEXIT DFHPD670 FT=2
```

The VERBEXIT provides a formatted dump of CICS ONC RPC control blocks.

Migrating between CICS versions

CICS ONC RPC is part of the CICS Transaction Server base. None of the IBM-supplied programs for CICS ONC RPC can be moved to CICS Transaction Server from earlier releases.

Modifying z/OS Communications Server data sets

You can define the CICS Transaction Server region to z/OS Communications Server in the TCPIP.PROFILE. data set to reserve specific ports for ONC RPC applications.

This is described in *z/OS Communication Server: IP Configuration Guide*.

Defining CICS ONC RPC resources to CICS

CICS ONC RPC provides two RDO groups defining CICS resources used by CICS ONC RPC: DFHRP and DFHRPF.

Transaction definitions for CICS ONC RPC transactions

These CICS ONC RPC transactions are defined in the locked group DFHRP.

CRPA Alias

CRPC Connection manager

CRPM
      Server controller

These definitions cannot be changed.

Transaction definitions for extra alias transactions

You may want to use other alias transaction names for various reasons.
* Auditing purposes
* Resource and command checking
* Allocating initiation priorities
* Allocating database plan selection
* Assigning different runaway values for different CICS programs

If you do, you must also define these to CICS, copying the definition from CRPA, and making amendments as necessary. The CRPA definition is as follows:

```
DEFINE  TRANSACTION(CRPA)      GROUP(DFHRP)
        PROGRAM(DFHRPAS)       TWASIZE(0)
        PROFILE(DFHCICST)      STATUS(ENABLED)
        TASKDATALOC(BELOW)     TASKDATAKEY(USER)
        RUNAWAY(SYSTEM)        SHUTDOWN(ENABLED)
        PRIORITY(1)            TRANCLASS(DFHTCL00)
        DTIMOUT(NO)            INDOUBT(BACKOUT)
        SPURGE(YES)            TPURGE(NO)
        RESSEC(NO)             CMDSEC(NO)
```

If you want a CICS program to run under an alias with a name other than CRPA, you can enter this in the connection manager when defining the attributes of the 4-tuple associated with the CICS program, as described in "Defining the attributes of a 4-tuple" on page 287. The name of the alias can also be changed by the **Decode** function, as described in "Changing the alias and CICS program" on page 308.

Changing the CMDSEC and RESSEC values

You might want to define new alias transactions with CMDSEC(YES) or RESSEC(YES) in order to enforce security checking on the programs run under the alias transaction, including the CICS program that services the client request.

None of the IBM-supplied programs used by the alias use any of system programmer interface (SPI) commands, so CMDSEC need not be changed. However, if you want to oversee the use of SPI commands by the CICS program, resource checker, or **Encode** function of the converter, CMDSEC(YES) is required.

Program definitions for CICS ONC RPC programs

All the CICS ONC RPC programs are defined in the locked group DFHRP.

Program definitions for user-written programs

You need to make definitions for: CICS programs, converters, user-written XDR routines, and a resource checker.

LANGUAGE option

User-written XDR routines should be defined with LANGUAGE(C). Converters and CICS programs should be defined with an appropriate LANGUAGE.

CEDF option

Program definitions for CICS programs must include CEDF(YES) if EDF is required for debugging.

If you want to use EDF, you must enter a terminal ID in the connection manager when defining the attributes of the 4-tuple associated with the CICS program, as described in "Defining the attributes of a 4-tuple" on page 287.

EXECKEY option

CICS operates with storage protection only if the SIT parameter STGPROT is set to YES, and the system has the required hardware and software.

Converters and the resource checker should not be regarded as application programs when defining storage. You are recommended to define them as EXECKEY(CICS). This allows them to modify CICS-key storage.

When the **Decode** and **Encode** functions allocate storage to hold the converted data, that storage should be allocated as CICS-key.

User-written XDR routines must be defined as EXECKEY(CICS).

CICS programs should be defined as EXECKEY(USER), unless there is some reason for defining them as CICS-key in your CICS system. Defining programs as EXECKEY(USER) prevents them from overwriting CICS.

If you specify EXECKEY(USER) for the CICS program, ensure that TASKDATAKEY(USER) is specified for the alias. USER is the default TASKDATAKEY setting in the alias definition in the supplied group DFHRP.

If you have CICS programs that need to be specified with EXECKEY(CICS), you are advised to specify TASKDATAKEY(CICS) for the alias that will execute them.

RELOAD option

You should specify RELOAD(YES) for any user-written XDR routines to prevent errors in CICS ONC RPC disable processing.

Definitions for remote CICS programs

If a CICS program that is to service a remote procedure call runs in a different CICS system from CICS ONC RPC, a program definition is required on both the local system and the remote system.

The program resides on the remote system, so its definition there is straightforward. The program definition on the local system:

- Must include a REMOTESYSTEM parameter to specify the system on which the program resides.
- Can optionally include a REMOTENAME parameter if you want the names on the local system and remote system to be different.
- Can optionally include a TRANSID parameter:
  - If TRANSID is not specified, the CICS program runs under the CICS mirror transaction on the remote CICS system.
  - If TRANSID is specified, the program in the remote CICS system runs under the transaction name given. See "Transaction definitions for extra alias transactions" on page 275 for reasons why you may want a different name.

    If the remote transaction ID is specified, you must provide a matching transaction definition in the remote CICS system. This definition must specify the appropriate mirror program for the remote system (DFHMIRS for CICS for MVS/ESA and CICS Transaction Server for z/OS systems).

If a CICS program is running on a CICS platform other than CICS for MVS/ESA or CICS Transaction Server for z/OS similar considerations apply, but you should refer to the DPL details for that platform.

Mapset definition

Mapset definitions are supplied in the group DFHRP for the connection manager mapsets. The definitions cannot be changed.

Transient data definitions

CICS provides a resource definition for the CICS ONC RPC message transient data queue CRPO. The resource definition is in group DFHDCTG, which is part of DFHLIST.

Group DFHDCTG is not protected by a lock, so the definitions it contains can be modified if required. CRPO is defined as an extrapartition queue, but you can make the destination intrapartition or indirect if you prefer.

If you leave CRPO defined as an extrapartition queue, you must add a suitable DD statement for the extrapartition queue in the CICS JCL, for example:

```
//CRPO   DD   SYSOUT=A
```

XLT definitions

The XLT system initialization parameter and its associated transaction list should allow the connection manager, CRPC, to be started during normal CICS shutdown. If CICS ONC RPC is delaying shutdown, the connection manager can be used to force an immediate disable of CICS ONC RPC.

Chapter 24. Configuring CICS ONC RPC using the connection manager

The connection manager has four main functions.
- Enabling CICS ONC RPC
- Disabling CICS ONC RPC
- Controlling the operating options and 4-tuple information stored in the CICS ONC RPC data set
- Controlling the operating options and 4-tuple information in current use when CICS ONC RPC is enabled

The rest of this chapter describes:

Starting the connection manager

You can start the connection manager in various ways.
- From a terminal that supports BMS maps. You can work with the connection manager panels described in this section.
- From a CICS console.
- Using an **EXEC CICS START** command.
- From a sequential terminal.

The effect of starting the connection manager depends on:
- Whether CICS ONC RPC is enabled or disabled
- Whether you start the connection manager from a terminal that permits the use of BMS
- Whether you enter additional data with the transaction name
- Whether the Automatic Enable option in the CICS ONC RPC definition record is set to YES

When CICS ONC RPC is disabled, the effect of entering the transaction name (and optional additional data) on a terminal that supports BMS is as follows:

CRPC

- If Automatic Enable is YES, automatic enable processing occurs.
- If Automatic Enable is NO, a BMS panel (DFHRP01) is shown.
- If there is no CICS ONC RPC definition record yet, a BMS panel (DFHRP01) is shown.

CRPC E A(N)

- A BMS panel (DFHRP01) is shown.

CRPC E A(Y)

- Automatic enable processing occurs. If there is no CICS ONC RPC definition record, one is created using default values for the options, but no 4-tuples are registered.

If you start the connection manager in a way that does not allow panels to be shown (EXEC CICS START, or non-BMS terminal, for example) and the action is to show a panel, error message DFHRP1505 is produced.

When CICS ONC RPC is enabled, the effect of entering the transaction name (and optional additional data) is as follows:

- CRPC displays panel DFHRP04, or produces error message DFHRP1505 if panels cannot be shown.
- CRPC D(N) causes normal disable processing.
- CRPC D(I) causes immediate disable processing.

The forms CRPC E A(N), CRPC E A(Y), CRPC D(N), and CRPC D(I) are called fast-path commands.

z/OS Communications Server should be started before you try to enable CICS ONC RPC with the connection manager, otherwise you cannot register 4-tuples, and you have to reenable CICS ONC RPC after starting z/OS Communications Server.

```
CRPC                    CICS ONC RPC for MVS/ESA                  DFHRP01

Select one of the following. Then press Enter.

_   1. Enable CICS ONC RPC
    2. View or modify the CICS ONC RPC data set













Current Status: Disabled


                                     SYSID= CI41  APPLID= IYK1ZFL1
PF1=Help    PF3=Exit    PF9=Messages    PF12=Return
```

Figure 40. Panel DFHRP01

```
CRPC                     CICS ONC RPC for MVS/ESA                      DFHRP04

Select one of the following. Then press Enter.

_   1. Disable CICS ONC RPC
    2. View or modify the CICS ONC RPC data set
    3. View or modify CICS ONC RPC status









Current Status: Enabled


                                        SYSID= CI41  APPLID= IYK1ZFL1
PF1=Help   PF3=Exit   PF9=Messages
```

Figure 41. Panel DFHRP04

Using the connection manager BMS panels

All leading and trailing blanks are ignored on BMS input.

At the top of all panels is a panel identifier in the right corner (for example, DFHRP02) and CRPC in the left corner.

On the bottom of all panels, the fourth line from the bottom gives the status of CICS ONC RPC, the third line from the bottom is a prompt line, while the bottom line lists the available PF keys, which can include:

PF1   Help information (all panels)

PF2   Delete definition from the CICS ONC RPC data set (only where shown)

PF3   Exit CRPC (you are prompted to confirm by using PF3 again)

PF4   Write fields to the CICS ONC RPC data set (only where shown)

PF7   Scroll up (only where shown)

PF8   Scroll down (only where shown)

PF9   Display messages relating to current input

PF12   Cancel this panel and return to the previous panel

Connection manager error message output

The destination of connection manager messages depends on the nature of the message.

- Severe errors requiring operator intervention are sent to the console. No other messages go to the console.
- Messages relating to invalid input on the panel can be displayed by pressing PF9.
- Messages reporting internal errors are sent to CRPO, and in most cases they can be displayed on the terminal by pressing PF9.

Using PF9 to display messages

During the operation of the connection manager, error messages might be issued.

These are not displayed immediately on the screen, but a prompt appears on the prompt line to say that messages are waiting to be viewed. To see the messages, press PF9. The number and text of the messages is displayed. You can look up the messages in *CICS Messages and Codes* for more information about errors, and for advice about what to do next.

When you have read the messages, you can press Enter, PF3, or PF12 to return to the input panel.

Starting the connection manager when CICS ONC RPC is disabled

If CICS ONC RPC is disabled, panel DFHRP01 is shown.

Select an option, then press Enter.

Option For more information see:

1   "Enabling CICS ONC RPC" on page 284

2   "Updating the CICS ONC RPC data set" on page 296

Starting the connection manager when CICS ONC RPC is enabled

If CICS ONC RPC is enabled, panel DFHRP04 is shown.

Select an option, then press Enter.

Option For more information see:

1   "Disabling CICS ONC RPC" on page 294

2   "Updating the CICS ONC RPC data set" on page 296

3   "Updating CICS ONC RPC status"

Updating CICS ONC RPC status

If you select option 3 on panel DFHRP04, panel DFHRP10 is shown.

```
CRPC                    CICS ONC RPC for MVS/ESA Update Status              DFHRP10


Select one of the following. Then press Enter.

_   1. Change CICS ONC RPC settings
    2. Register procedure(s)
    3. Unregister procedure(s)
    4. View or modify alias list









Current Status: Enabled


                                          SYSID= CI41  APPLID= IYK1ZFL1
PF1=Help    PF3=Exit    PF9=Messages    PF12=Return
```

Figure 42. Panel DFHRP10

Select an option, then press Enter.

Option For more information see:
1   "Changing the CICS ONC RPC status"
2   "Defining, saving, modifying, and deleting 4-tuples" on page 286
3   "Unregistering 4-tuples" on page 292
4   "Processing the alias list" on page 300

Changing the CICS ONC RPC status

If you select option 1 on panel DFHRP10, panel DFHRP16 is shown.

You can type over any of the entries except CRPM Userid to change the values currently used by CICS ONC RPC. CRPM Userid is displayed only for information. CRPM Userid cannot be changed without first disabling CICS ONC RPC.

```
CRPC                     CICS ONC RPC for MVS/ESA Status              DFHRP16


    Trace( STARTED )                       Trace Level( 1   )
    Resource Checker( NO  )                CRPM Userid( CICSUSER )










Current Status: Enabled


                                        SYSID= CI41  APPLID= IYK1ZFL1
PF1=Help   PF3=Exit   PF9=Messages   PF12=Return
```

Figure 43. Panel DFHRP16

Enabling CICS ONC RPC

You can enable CICS ONC RPC in two ways: operator-assisted enable, or automatic enable.

When CICS ONC RPC is disabled, the connection manager allows you to:
- Create or update the CICS ONC RPC definition record in the data set
- Add, delete, and change 4-tuple records in the data set
- Enable CICS ONC RPC

You can use the connection manager to enable CICS ONC RPC in two ways:
- Operator-assisted enable—before you enable CICS ONC RPC, you can:
  – Modify any or all of the options
  – Select which 4-tuples are to be registered
  – Modify the attributes of 4-tuples before registration

  When you enable CICS ONC RPC, options to control its operation come into play, and 4-tuples can be registered.

  The changes you make during an operator-assisted enable can be temporary, lasting only until the next time you disable CICS ONC RPC, or you can store them into the CICS ONC RPC data set, and use them the next time you enable CICS ONC RPC.
- Automatic enable—the contents of the CICS ONC RPC definition record determine the options to control the operation of CICS ONC RPC until the next time you disable it. Some 4-tuples might be registered, depending on an attribute in the 4-tuple definition.

The CICS ONC RPC data set is a store of operating environment information. It contains two kinds of records: the CICS ONC RPC definition record contains the operating options, and 4-tuple records contain the 4-tuple information.

Setting and modifying options

If you start the connection manager when CICS ONC RPC is disabled, and select option 1 on panel DFHRP01, panel DFHRP02 is shown.

```
CRPC                    CICS ONC RPC for MVS/ESA Enable              DFHRP02

Overtype to Modify
                              Choice        Possible Options

Trace                  ===>   STARTED       STArted | STOpped

Trace Level            ===>   1             1 | 2

Resource Checker       ===>   NO            Yes | No

CRPM Userid            ===>   CICSUSER

Automatic Enable       ===>   NO            Yes | No




Current Status: Disabled


                                        SYSID= CI41  APPLID= IYK1ZFL1
PF1=Help   PF3=Exit   PF4=Save   PF9=Messages   PF12=Return
```

Figure 44. Panel DFHRP02

The values displayed in the Choice column are those stored in the CICS ONC RPC data set. The data set is initialized with the values shown in Figure 44, except that the value displayed for CRPM Userid is the default CICS user ID for the CICS system in which CICS ONC RPC is operating.

You can make entries in the fields listed below. Entries may be in lowercase or uppercase. Where entries to a field are restricted (for example, YES or NO) you can enter the whole option (YES) or the minimum (Y). In the panels, the minimum entry is shown in uppercase in the Possible Options column. In the reference material in this manual, the minimum entry is given in parentheses after the full entry.

Trace   Specifies whether CICS ONC RPC tracing is active. STARTED (STA) means it is active, STOPPED (STO) means it is not. The default value is STARTED.

CICS ONC RPC exception trace entries are always written to CICS internal trace whatever the setting of this option. To get non-exception trace entries written, CICS trace must be started, and this option must be set to STARTED.

Trace Level

Specifies the trace level for CICS ONC RPC. The value 1 means that level 1 trace points are traced, and 2 means that both level 1 and 2 are traced. The default value is 1.

Resource Checker

YES (Y) means that CICS ONC RPC is to call the user-written resource-checking module on receipt of every incoming RPC request. NO (N) means the resource checker is not to be called. The default is NO.

CRPM Userid

    Specifies the CICS user ID under which the server controller is to run. The default is the default user ID for the CICS system in which CICS ONC RPC is operating.

Automatic Enable

    Enter YES (Y) or NO (N). If YES is stored in the CICS ONC RPC data set, you can enable CICS ONC RPC by just typing CRPC; all values are defaulted from the CICS ONC RPC data set, CICS ONC RPC becomes enabled without further user input, and all the 4-tuples with YES for their Register from Data Set option are registered. The default value is NO.

    Setting this field has an effect only when you enable CICS ONC RPC. If you use PF4 to save the values to the CICS ONC RPC data set, this value will be effective the next time you enable, unless you override it. A YES in this field in the CICS ONC RPC data set may be overridden by the fast path command CRPC E A(N).

Validating, saving, and activating options

After you have made your changes on panel DFHRP02, press Enter to get them validated by the connection manager.

If you want to save the new values in the CICS ONC RPC data set, press PF4.

If you press Enter a second time, CICS ONC RPC becomes enabled, and panel DFHRP03 is shown, as described in "Defining, saving, modifying, and deleting 4-tuples."

When CICS ONC RPC is enabled

When CICS ONC RPC is enabled, the connection manager allows you to:
- Update the CICS ONC RPC definition record in the data set
- Add, delete, and change 4-tuple records in the data set
- Change the options being used to control the operation of CICS ONC RPC
- Register 4-tuple definitions from the data set
- Create temporary 4-tuple definitions and register them
- Unregister 4-tuple definitions
- Disable CICS ONC RPC

There are two ways of disabling CICS ONC RPC: normal, and immediate. The effects of disable processing are described in "Disabling CICS ONC RPC" on page 294.

Defining, saving, modifying, and deleting 4-tuples

The first panel for defining, saving, modifying, and deleting 4-tuples is DFHRP03.

The first panel for defining, saving, modifying, and deleting 4-tuples is DFHRP03. (See Figure 45 on page 287.) This panel is shown as soon as you have enabled CICS ONC RPC, or if you choose option 2 on panel DFHRP10.

```
CRPC                        CICS ONC RPC for MVS/ESA                   DFHRP03
                          Remote Procedure Registration

Select one of the following. Then press Enter.


_   1. Register procedures from the data set
    2. List procedures sequentially
    3. Register a new procedure
    4. Retrieve a specified procedure from the data set (Enter required data)
          Program Number       ===> _____     0-FFFFFFFF
          Version Number       ===> _____     0-FFFFFFFF
          Procedure Number     ===> _____     1-FFFFFFFF
          Protocol             ===> UDP          Udp | Tcp






Current Status: Enabled


                                        SYSID= CI41  APPLID= IYK1ZFL1
PF1=Help    PF3=Exit    PF9=Messages    PF12=Return
```

Figure 45. Panel DFHRP03

Option For more information see:

1	See below.
2	"Defining the attributes of a 4-tuple"
3	"Unregistering 4-tuples" on page 292
4	See below.

If you select option 1, the 4-tuples in the CICS ONC RPC data set that have YES for their Register from Data Set attribute are all registered.

If you specify a 4-tuple for which there is no definition in the CICS ONC RPC data set, a message is issued when you press Enter, and panel DFHRP03 remains on the screen.

Defining the attributes of a 4-tuple

When you select option 3 or option 4 on panel DFHRP03, panel DFHRP5 is shown. If you chose option 3, some of the fields are empty, but if you chose option 4, the details of the selected 4-tuple are shown. You have to supply more information on panel DFHRP5B.

```
CRPC       CICS ONC RPC for MVS/ESA Remote Procedure Registration       DFHRP5

Overtype to Modify. Then press Enter to Validate

  ONC RPC ATTRIBUTES
   ONC RPC Program Number   ===>  _____        0-FFFFFFFF
   ONC RPC Version Number   ===>  _____        0-FFFFFFFF
   ONC RPC Procedure Number ===>  _____         1-FFFFFFFF
   Protocol                 ===>  UDP             Udp | Tcp
   RPC Call Type            ===>  BLOCKING        Blocking | Nonblocking
   Inbound XDR Routine      ===>  _____
   Outbound XDR Routine     ===>  _____
  CICS ATTRIBUTES
   ALIAS Transaction ID     ===>  CRPA
   EDF Terminal ID          ===>  ____
+  Program Name             ===>  _____



Current Status: Enabled


                                        SYSID= CI41  APPLID= IYK1ZFL1
PF1=Help    PF3=Exit    PF4=Save    PF8=Forward    PF9=Messages    PF12=Return
```

```
CRPC       CICS ONC RPC for MVS/ESA Remote Procedure Registration       DFHRP5B

Overtype to Modify. Then press Enter to Validate

+ CICS ONC RPC ATTRIBUTES
   Converter Program Name   ===>  _____
   Encode                   ===>  NO              Yes | No
   Decode                   ===>  YES             Yes | No
   Getlengths               ===>  YES             Yes | No
    Server Input Length     ===>  _____           0 - 32767 Bytes
    Server Output Length    ===>  _____           0 - 32767 Bytes
    Server Data Format      ===>  CONTIGUOUS      Contiguous | Overlaid
   Register from Data set   ===>  YES             Yes | No





Current Status: Enabled


                                        SYSID= CI41  APPLID= IYK1ZFL1
PF1=Help    PF3=Exit    PF4=Save    PF7=Back    PF9=Messages    PF12=Return
```

Figure 46. Panels DFHRP5 and DFHRP5B

After you have made your modifications to panel DFHRP5, you should press PF8
to move to panel DFHRP5B. From panel DFHRP5B you can press PF7 if you want
to go back to panel DFHRP5. After you have made your modifications to the
panels, you press Enter to get all the modifications validated.

The attributes of a 4-tuple are divided into three categories:
• ONC RPC attributes
• CICS attributes
• CICS ONC RPC attributes

ONC RPC attributes
The first four options establish the 4-tuple whose attributes are being defined.

ONC RPC Program Number

Specifies the program number of the 4-tuple as a hexadecimal string of 1

through 8 characters. You are advised not to use numbers in the range 0 through 1FFFFFFF, as these numbers are reserved for public network services and are allocated by Sun Microsystems.

ONC RPC Version Number
> Specifies the version number of the 4-tuple as a hexadecimal string of 1 through 8 characters.

ONC RPC Procedure Number
> Specifies the procedure number of the 4-tuple as a hexadecimal string of 1 through 8 characters. Procedure 0 is reserved by z/OS Communications Server for a procedure with no parameters and no processing that returns an empty reply.

Protocol
> Specifies the protocol of the 4-tuple. UDP (U) for UDP, or TCP (T) for TCP.

The remaining options specify the attributes of the 4-tuple.

RPC Call Type
> Specifies whether CICS ONC RPC is to treat calls from clients as BLOCKING (B) or NONBLOCKING (N). If NONBLOCKING is specified, the outbound XDR routine cannot be specified, and no reply is sent to the client. The default is BLOCKING.

Inbound XDR Routine
> Specifies the name of the inbound XDR routine. If an XDR library function is used, its full name is specified. See Table 26 on page 305 to find out which library routines can be specified here. If a user-defined routine is used, its name (maximum 8 characters) is specified.

Outbound XDR Routine
> Specifies the name of the outbound XDR routine, if RPC Call Type is BLOCKING. If an XDR library function is used, its full name is specified. See Table 26 on page 305 to find out which library routines can be specified here. If a user-defined routine is used, its name (maximum 8 characters) is specified. A blank input is valid only if RPC Call Type is NONBLOCKING.

CICS attributes

The alias transaction ID, EDF terminal ID, and program name are the attributes you must specify for CICS.

ALIAS Transaction ID
> Specifies the transaction ID to be used for the alias. If this is omitted, and not provided by the **Decode** function, the alias transaction ID is CRPA. For reasons why you might want a different name from CRPA, see "Transaction definitions for extra alias transactions" on page 275.

EDF Terminal ID
> Specifies the terminal ID to be used for the alias. You need a terminal ID only if you want to use execution diagnostic facility (EDF) to debug the resource checker, CICS program, or **Encode** function of the converter. A blank means that you cannot use EDF. EDF setup is described in "Using EDF" on page 338.

Program Name
> Specifies the name of the CICS program that is to be called to service a request for this 4-tuple.

CICS ONC RPC attributes

Converter Program Name
> Specifies the name of the converter program. This name must be specified.

Encode
> YES (Y) means that CICS ONC RPC must call the **Encode** function of the converter when servicing a client request for this 4-tuple; NO (N) means that it must not. The default is NO.

Decode
> YES (Y) means that CICS ONC RPC must call the **Decode** function of the converter when servicing a client request for this 4-tuple; NO (N) means that it must not. The default is YES.

Getlengths
> YES (Y) means that the connection manager must call the **Getlengths** function of the converter before registering this 4-tuple. NO (N) means that it must not. If you specify YES here, you should ignore the next two attributes, but you can set Server Data Format. If you specify NO here, you must specify the next three attributes. The default is YES.

Server Input Length
> For the use of this option, see the description of Server Data Format.
>
> If you specified YES for the Getlengths option, leave this field blank.

Server Output Length
> For the use of this option, see the description of Server Data Format.
>
> If you specified YES for the Getlengths option, leave this field blank.

Server Data Format
> A value that controls:
> - How the input data pointer for **Encode** will be set up
> - How the communication area length to be checked by the connection manager is calculated
>
> The values you can specify are as follows:
>
> **CONTIGUOUS**
> > The value of the data pointer that will be passed to **Encode**, or to the outbound XDR routine if **Encode** is not used for this 4-tuple, is the address of the CICS program communication area plus the value of Server Input Length, though **Decode** can modify this offset.
> >
> > The connection manager calculates a communication area length by adding the values of Server Input Length and Server Output Length. If this length exceeds 32 767 bytes, message DFHRP1965 is issued. If this length is different from the actual length of the communication area passed from **Decode** to the CICS program, errors might occur in the processing of client requests.
>
> **OVERLAID**
> > The value of the data pointer that will be passed to **Encode**, or to the outbound XDR routine if **Encode** is not used for this 4-tuple, is the address of the CICS program communication area.
> >
> > The connection manager calculates a communication area length by taking the larger of the output values of Server Input Length and Server Output Length. If this length is different from the actual

length of the communication area passed to the CICS program, errors might occur in the processing of client requests.

If you specified YES for the Getlengths option, the value in this field is used as an input to the **Getlengths** function of the converter.

Register from Data Set

YES (Y) means that the 4-tuple is to be registered:

- During automatic enable processing
- When option 1 is selected on panel DFHRP03, as described in "Registering the 4-tuples"

NO (N) means that it is not. The default is YES. Entries specified as NO can be stored in the CICS ONC RPC data set and you can register them at any time when CICS ONC RPC is enabled.

Saving new 4-tuple definitions

There are five ways of saving new 4-tuple definitions.

- On panel DFHRP03, select option 3. Complete panels DFHRP5 and DFHRP5B, and validate your input as described in "Defining the attributes of a 4-tuple" on page 287. Press PF4 to save the definition in the CICS ONC RPC data set.
- On panel DFHRP03, select option 4. Modify the panels DFHRP5 and DFHRP5B, and validate your input as described in "Defining the attributes of a 4-tuple" on page 287. Press PF4 to save the definition in the CICS ONC RPC data set.
- On panel DFHRP20, select option 3. Complete panels DFHRP21 and DFHRP2B, and validate your input as described in "Changing the attributes of a 4-tuple" on page 299. Press Enter to save the definition in the CICS ONC RPC data set.
- On panel DFHRP20, select option 4. Modify the panels DFHRP21 and DFHRP2B, and validate your input as described in "Changing the attributes of a 4-tuple" on page 299. Press Enter to save the definition in the CICS ONC RPC data set.
- On panel DFHRP03, select option 2. Then on panel DFHRP14, enter command **M** against a 4-tuple. Modify the panels DFHRP21 and DFHRP2B, and validate your input as described in "Changing the attributes of a 4-tuple" on page 299. Press Enter to save the definition in the CICS ONC RPC data set.

Modifying existing 4-tuple definitions

To change some of the attributes of a 4-tuple that already has a definition in the CICS ONC RPC data set, select option 4 on panel DFHRP03 or panel DFHRP20.

Deleting existing 4-tuple definitions

You can delete existing 4-tuple definitions from the CICS ONC RPC data set in two ways.

- On panel DFHRP03, select option 2. Then on panel DFHRP14 you can enter **D** against 4-tuples in the list, and they are deleted from the data set when you press Enter.
- On panel DFHRP21, by using key PF2, as described in "Changing the attributes of a 4-tuple" on page 299.

Registering the 4-tuples

You can register 4-tuples in any of the following ways.

- You can register all the 4-tuples in the CICS ONC RPC data set that are defined with YES specified for Register from Data Set. To do this, select option 1 on panel DFHRP03, and press Enter. After these 4-tuples have been registered, panel DFHRP03 is still displayed, so you can make other selections.
- You can register 4-tuple definitions one at a time. To do this, you use option 3 or option 4 on panel DFHRP03. Make changes, if you need any, to panels DFHRP5 and DFHRP5B and get them validated as described in "Defining the attributes of a 4-tuple" on page 287. To register the definition, press Enter.
- You can register 4-tuples from a list. See "Working with a list of 4-tuples" on page 298.
- When CICS ONC RPC is disabled, you can register all the 4-tuples in the CICS ONC RPC data set that have YES for their Register from Data Set attribute by initiating automatic enable processing.

When a 4-tuple is registered, two things happen:
- If the program-version-protocol 3-tuple has not yet been registered with TCP/IP for MVS, it is registered. The Portmapper assigns a port number to this combination, and that port number is the one that clients use to request the service represented by this 4-tuple. Procedure 0 for the program, version, and protocol becomes available to callers.
- The resources associated with the 4-tuple become available to service client requests. When a client request arrives in CICS ONC RPC, the resources used to service it are those of the 4-tuple whose program, version, and procedure numbers match those of the request, and whose protocol matches the protocol used to transmit the request from the client to the server.

Limits on registration

CICS ONC RPC makes a total of 252 sockets available for use. One socket is used by each program/version/protocol 3-tuple from the time the first 4-tuple for that program, version and protocol is registered. This socket remains in use until the last 4-tuple with that program and version is unregistered. One socket is used by each TCP call for the duration of the call.

If you register too many 4-tuples, you reduce the service that CICS ONC RPC can give to incoming client requests. If you attempt to register more than 252 program-version-protocol 3-tuples with z/OS Communications Server, the results are unpredictable.

Unregistering 4-tuples

You can unregister 4-tuples that have previously been registered with CICS ONC RPC only when CICS ONC RPC is already enabled.

From panel DFHRP10, if you select option 3, panel DFHRP11 is shown. (See Figure 47 on page 293.)

```
CRPC                       CICS ONC RPC for MVS/ESA                    DFHRP11
                          Remote Procedure Unregister

Select one of the following. Then press Enter.


_   1. Unregister procedures from a list
    2. Unregister a specified procedure (Enter required data)
            Program Number      ===> _____   0-FFFFFFFF
            Version Number      ===> _____   0-FFFFFFFF
            Procedure Number    ===> _____   1-FFFFFFFF
            Protocol            ===> UDP        Udp | Tcp








Current Status: Enabled


                                        SYSID= CI41  APPLID= IYK1ZFL1

PF1=Help    PF3=Exit    PF9=Messages    PF12=Return
```

Figure 47. Panel DFHRP11

Select an option, then press Enter.

Option For more information see:
1   "Unregistering 4-tuples from a list"
2   "Unregistering 4-tuples one by one"

Unregistering 4-tuples one by one

Before you select option 2 on panel DFHRP11, you must supply the program
number, version number, procedure number, and the protocol.

Program Number
        The program number of the 4-tuple to be unregistered.

Version Number
        The version number of the 4-tuple to be unregistered.

Procedure Number
        The procedure number of the 4-tuple to be unregistered.

Protocol
        The protocol of the 4-tuple to be unregistered.

If you specify a 4-tuple that is registered, it is unregistered when you press Enter,
and panel DFHRP11 remains on the screen.

If you specify a 4-tuple that is not registered, a message is issued when you press
Enter, and panel DFHRP11 remains on the screen.

Unregistering 4-tuples from a list

If you select option 1 on panel DFHRP11, the panel DFHRP12 is shown.

This panel presents a list of 4-tuples currently registered with CICS ONC RPC. If
you enter **U** against 4-tuples in the list, they are unregistered when you press
Enter. You can display the attributes of a 4-tuple by entering **?** against it, and
pressing Enter. Panel DFHRP13 is shown. (See Figure 49 on page 294.)

```
CRPC                       CICS ONC RPC for MVS/ESA                    DFHRP12
                          Registered Procedures List

Enter 'U' to Unregister, or '?' to display details of a procedure
_     Prog( 20000002 ) Vers( 00000001 ) Proc( 00000006 ) Prot( UDP )
_     Prog( 20000002 ) Vers( 00000001 ) Proc( 00000007 ) Prot( TCP )
_     Prog( 20000002 ) Vers( 00000001 ) Proc( 00000007 ) Prot( UDP )
_     Prog( 20000002 ) Vers( 00000001 ) Proc( 00000008 ) Prot( TCP )
_     Prog( 20000002 ) Vers( 00000001 ) Proc( 00000009 ) Prot( UDP )
_     Prog( 20000002 ) Vers( 00000001 ) Proc( 0000000A ) Prot( TCP )
_     Prog( 20000002 ) Vers( 00000001 ) Proc( 0000000B ) Prot( TCP )
_     Prog( 20000002 ) Vers( 00000001 ) Proc( 0000000B ) Prot( UDP )
_     Prog( 20000002 ) Vers( 00000001 ) Proc( 0000000C ) Prot( TCP )
_     Prog( 20000002 ) Vers( 00000001 ) Proc( 0000000C ) Prot( UDP )
_     Prog( _____ ) Vers( _____ ) Proc( _____ ) Prot( ___ )
_     Prog( _____ ) Vers( _____ ) Proc( _____ ) Prot( ___ )
_     Prog( _____ ) Vers( _____ ) Proc( _____ ) Prot( ___ )
_     Prog( _____ ) Vers( _____ ) Proc( _____ ) Prot( ___ )
_     Prog( _____ ) Vers( _____ ) Proc( _____ ) Prot( ___ )
_     Prog( _____ ) Vers( _____ ) Proc( _____ ) Prot( ___ )
Current Status: Enabled


                                   SYSID= CI41  APPLID= IYK1ZFL1
PF1=Help PF2=Refresh PF3=Exit PF7=Back PF8=Forward PF9=Messages PF12=Return
```

Figure 48. Panel DFHRP12

```
CRPC                       CICS ONC RPC for MVS/ESA                    DFHRP13
                          Display Registered Procedure

      Program Number( 20000002 )          Version Number( 00000001 )
      Procedure Number( 00000006 )        Protocol( UDP )
      RPC Call Type( Blocking     )       Inbound XDR( XDR_WRAPSTRING    )
      Outbound XDR( XDR_WRAPSTRING    )    Alias Transid( CRPA )
      Alias Termid(      )                Server Program Name( STRING6  )
      Converter Program Name( RINGCVNY )   Getlengths( NO  )
      Decode( YES )                       Encode( NO  )
      Server Input Length( 00001 )        Server Output Length( 00001 )
      Server Data Format( CONTIGUOUS  )




Current Status: Enabled


                                   SYSID= CI41  APPLID= IYK1ZFL1
PF1=Help  PF3=Exit  PF12=Return
```

Figure 49. Panel DFHRP13

Disabling CICS ONC RPC

From panel DFHRP04, select option 1; panel DFHRP06 is shown.

```
CRPC                   CICS ONC RPC for MVS/ESA Disable            DFHRP06

Select the type of disable required. Then press Enter.


   Type of Disable  ===>  _____      Normal | Immediate









Current Status: Enabled


                                          SYSID= CI41  APPLID=  IYK1ZFL1
PF1=Help    PF3=Exit    PF9=Messages    PF12=Return
```
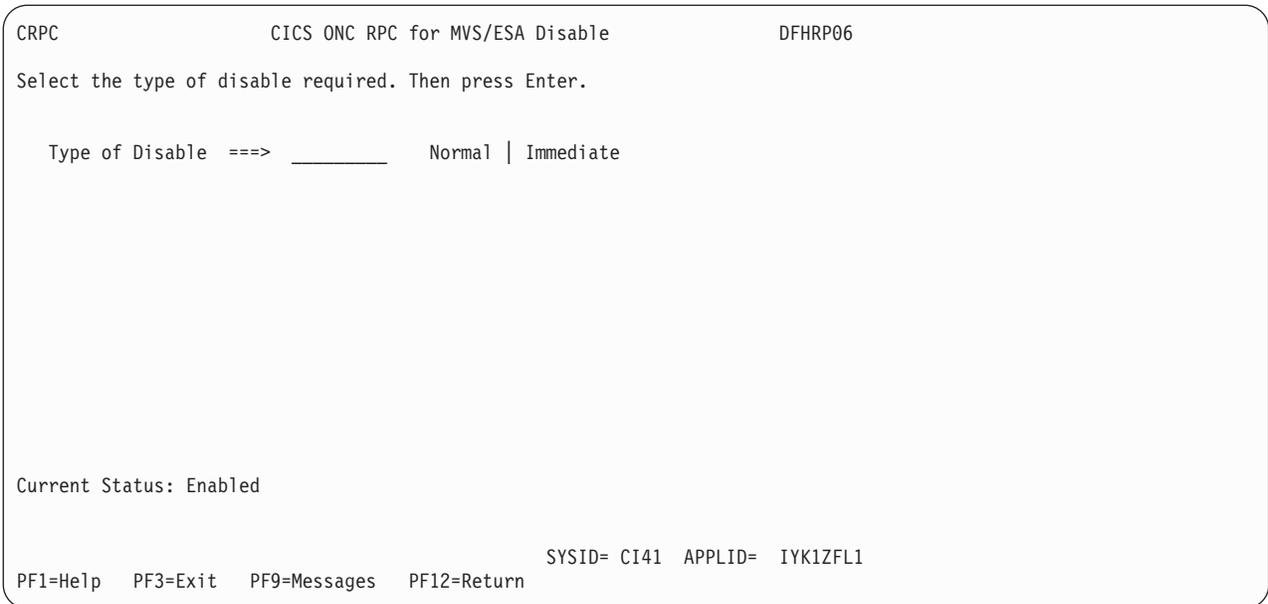
Figure 50. Panel DFHRP06

In this panel there is only one field to enter.

Type of Disable

> **NORMAL (N)**
>> Normal disable processing is started.
>> - All program-version pairs are unregistered from z/OS Communications Server.
>> - All work that has already entered CICS ONC RPC is allowed to run to completion, and replies are sent to the relevant client.
>
> **IMMEDIATE (I)**
>> Immediate disable processing is started.
>> - Aliases not yet started do not start at all.
>> - CICS programs running under aliases are allowed to end, and then the alias abends. If the CICS program ends normally, and was called using DPL, the changes it makes to recoverable resources are committed. If the CICS program is a local program, the changes it makes to recoverable resources are backed out unless the CICS program takes a sync point with EXEC CICS SYNCPOINT.
>> - All the program-version pairs are unregistered from z/OS Communications Server.
>> - No replies are sent to clients, so they do not know whether the CICS program has run or not.

Pressing Enter causes the entry you have made to be validated. Pressing Enter a second time begins disable processing. The Current Status is changed to Disabling or Disabled, depending on the progress of disable processing. When disable processing is complete, pressing Enter changes the Current Status to Disabled.

The panel is displayed until you use PF3 or PF12.

On CICS normal shutdown

CICS normal shutdown starts normal disable processing for CICS ONC RPC.

On CICS immediate shutdown

On CICS immediate shutdown, all transactions are terminated. Clients are not informed of the shutdown or its effects. The program-version-protocol 3-tuples that are registered with z/OS Communications Server might remain registered.

Updating the CICS ONC RPC data set

If you select option 2 on panel DFHRP01, or option 2 on panel DFHRP04, panel DFHRP20 is shown.

```
 CRPC                      CICS ONC RPC for MVS/ESA                 DFHRP20
                          Update CICS ONC RPC Data set

 Select one of the following. Then press Enter.

 _  1. View or modify the CICS ONC RPC definition record
    2. Display a list of remote procedure definitions
    3. Define a new procedure
    4. Retrieve a specified procedure from the data set (Enter required data)
            Program Number     ===> _____     0-FFFFFFFF
            Version Number     ===> _____     0-FFFFFFFF
            Procedure Number   ===> _____     1-FFFFFFFF
            Protocol           ===> UDP          Udp | Tcp




 Current Status:


                                         SYSID= CI41  APPLID= IYK1ZFL1
 PF1=Help   PF3=Exit   PF9=Messages   PF12=Return
```

Figure 51. Panel DFHRP20

The Current Status field in this panel might show Enabled or Disabled, depending on which panel you came from.

Before selecting option 4, you must supply the following information:

Program Number
> The program number of the 4-tuple whose definition is to be retrieved.

Version Number
> The version number of the 4-tuple whose definition is to be retrieved.

Procedure Number
> The procedure number of the 4-tuple whose definition is to be retrieved.

Protocol
> The protocol of the 4-tuple whose definition is to be retrieved.

Select an option, then press Enter.

Option For more information see:
1   "Updating the CICS ONC RPC definition record" on page 297
2   "Working with a list of 4-tuples" on page 298

3 "Changing the attributes of a 4-tuple" on page 299
4 "Changing the attributes of a 4-tuple" on page 299

If you specify a 4-tuple which is not defined in the CICS ONC RPC data set, a message is issued when you press Enter, and panel DFHRP20 remains on the screen.

Updating the CICS ONC RPC definition record

If you select option 1 on panel DFHRP20, panel DFHRP22 is shown.

```
CRPC                    CICS ONC RPC for MVS/ESA                 DFHRP22
                  Update CICS ONC RPC Definition Record
Overtype to Modify
                             Choice       Possible Options

Trace                  ===>   STARTED     STArted | STOpped

Trace Level            ===>   1           1 | 2

Resource Checker       ===>   NO          Yes | No

CRPM Userid            ===>   CICSUSER

Automatic Enable       ===>   NO          Yes | No




Current Status:


                                   SYSID= CI41  APPLID= IYK1ZFL1
PF1=Help   PF3=Exit   PF9=Messages   PF12=Return
```

Figure 52. Panel DFHRP22

The values displayed in the Choice column are those stored in the CICS ONC RPC data set.

After you have made your changes you should press Enter to get them validated. You can then press Enter again to update the CICS ONC RPC data set with the values you have supplied. The next time you start the connection manager, the saved options are used to set up panel DFHRP02

Trace Specifies whether CICS ONC RPC tracing is active. STARTED (STA) means it is active, STOPPED (STO) means it is not. The default value is STARTED.

CICS ONC RPC exception trace entries are always written to CICS internal trace whatever the setting of this option. To get non-exception trace entries written, CICS trace must be started, and this option must be set to STARTED.

Trace Level
Specifies the trace level for CICS ONC RPC. The value 1 means that level 1 trace points are traced, 2 means that both level 1 and level 2 are traced. The default value is 1.

Resource Checker
YES (Y) means that CICS ONC RPC is to call the user-written resource-checking module on receipt of every incoming RPC request. NO (N) means the resource checker is not to be called. The default is NO.

CRPM Userid

Specifies the CICS user ID under which the server controller is to operate. The default is the default user ID for the CICS system in which CICS ONC RPC is operating.

Automatic Enable

Enter YES (Y) or NO (N). If YES is stored in the CICS ONC RPC data set, you can enable CICS ONC RPC by just typing CRPC; all values are defaulted from the CICS ONC RPC data set, CICS ONC RPC becomes enabled without further user input, and all the 4-tuples with YES for their Register from Data Set option are registered. The default value is NO.

Setting this field has an effect only when you enable CICS ONC RPC. If you save the values to the CICS ONC RPC data set, this value will be effective the next time you enable, unless you override it. The value of this field in the CICS ONC RPC data set may be overridden by the fast path command CRPC E A(N).

Working with a list of 4-tuples

If you select option 2 on panel DFHRP03, or option 2 on panel DFHRP20, panel DFHRP14 is shown.

```
CRPC                      CICS ONC RPC for MVS/ESA                  DFHRP14
                        Remote Procedure Definition List

Enter a command (press PF1 to view the list of valid commands).
_     Prog( 20000002 ) Vers( 00000001 ) Proc( 00000006 ) Prot( UDP )
_     Prog( 20000002 ) Vers( 00000001 ) Proc( 00000007 ) Prot( TCP )
_     Prog( 20000002 ) Vers( 00000001 ) Proc( 00000007 ) Prot( UDP )
_     Prog( 20000002 ) Vers( 00000001 ) Proc( 00000008 ) Prot( TCP )
_     Prog( 20000002 ) Vers( 00000001 ) Proc( 00000009 ) Prot( UDP )
_     Prog( 20000002 ) Vers( 00000001 ) Proc( 0000000A ) Prot( TCP )
_     Prog( 20000002 ) Vers( 00000001 ) Proc( 0000000B ) Prot( TCP )
_     Prog( 20000002 ) Vers( 00000001 ) Proc( 0000000B ) Prot( UDP )
_     Prog( 20000002 ) Vers( 00000001 ) Proc( 0000000C ) Prot( TCP )
_     Prog( 20000002 ) Vers( 00000001 ) Proc( 0000000C ) Prot( UDP )
_     Prog( _____ ) Vers( _____ ) Proc( _____ ) Prot( ___ )
_     Prog( _____ ) Vers( _____ ) Proc( _____ ) Prot( ___ )
_     Prog( _____ ) Vers( _____ ) Proc( _____ ) Prot( ___ )
_     Prog( _____ ) Vers( _____ ) Proc( _____ ) Prot( ___ )
_     Prog( _____ ) Vers( _____ ) Proc( _____ ) Prot( ___ )
_     Prog( _____ ) Vers( _____ ) Proc( _____ ) Prot( ___ )
Current Status:

                                SYSID= CI41  APPLID= IYK1ZFL1
PF1=Help PF2=Refresh PF3=Exit PF7=Back PF8=Forward PF9=Messages PF12=Return
```

Figure 53. Panel DFHRP14

This panel presents a list of 4-tuples currently defined in the CICS ONC RPC data set. If CICS ONC RPC is enabled, the 4-tuples that are currently registered are shown highlighted. You can put a command against a 4-tuple, and it takes effect when you press Enter. The following commands can be entered against a 4-tuple:

D   Deletes the definition from the data set.

R   If CICS ONC RPC is enabled, registers the 4-tuple with CICS ONC RPC. If CICS ONC RPC is disabled, this command produces an error message.

M   Shows panel DFHRP21. See "Changing the attributes of a 4-tuple" on page 299 for details.

?   Shows panel DFHRP15, which displays the attributes of a 4-tuple, but does

not allow changes.

```
CRPC                    CICS ONC RPC for MVS/ESA              DFHRP15
                     Display Registered Procedure

     Program Number( 20000002 )          Version Number( 00000001 )
     Procedure Number( 00000006 )        Protocol( UDP )
     RPC Call Type( Blocking     )       Inbound XDR( XDR_WRAPSTRING     )
     Outbound XDR( XDR_WRAPSTRING   )     Alias Transid( CRPA )
     Alias Termid(      )                Server Program Name( STRING6  )
     Converter Program Name( RINGCVNY )  Getlengths( NO  )
     Decode( YES )                       Encode( NO  )
     Server Input Length( 00000 )        Server Output Length( 00000 )
     Server Data Format( CONTIGUOUS  )   Register from Data set( Yes )




Current Status:


                                    SYSID= CI41  APPLID= IYK1ZFL1
 PF1=Help  PF3=Exit  PF12=Return
```

Figure 54. Panel DFHRP15

Changing the attributes of a 4-tuple

If you select option 3 or 4 on panel DFHRP20, or if you enter the **M** command on panel DFHRP14, panel DFHRP21 is shown.

The attributes of a 4-tuple are divided into three categories:
- ONC RPC attributes—see "ONC RPC attributes" on page 288.
- CICS attributes—see "CICS attributes" on page 289.
- CICS ONC RPC attributes—see "CICS ONC RPC attributes" on page 290.

```
CRPC      CICS ONC RPC for MVS/ESA Remote Procedure Definition      DFHRP21

Overtype to Modify. Then press Enter to Validate

  ONC RPC ATTRIBUTES
   ONC RPC Program Number   ===>  _____        0-FFFFFFFF
   ONC RPC Version Number   ===>  _____        0-FFFFFFFF
   ONC RPC Procedure Number ===>  _____         1-FFFFFFFF
   Protocol                 ===>  UDP             Udp | Tcp
   RPC Call Type            ===>  BLOCKING        Blocking | Nonblocking
   Inbound XDR Routine      ===>  _____
   Outbound XDR Routine     ===>  _____
  CICS ATTRIBUTES
   ALIAS Transaction ID     ===>  CRPA
   EDF Terminal ID          ===>  ____
+  Program Name             ===>  _____




Current Status:


                                     SYSID= CI41  APPLID= IYK1ZFL1
PF1=Help   PF2=Delete   PF3=Exit   PF8=Forward   PF9=Messages   PF12=Return
```

```
CRPC      CICS ONC RPC for MVS/ESA Remote Procedure Registration      DFHRP2B

Overtype to Modify. Then press Enter to Validate

+ CICS ONC RPC ATTRIBUTES
   Converter Program Name   ===>  _____
   Encode                   ===>  NO              Yes | No
   Decode                   ===>  YES             Yes | No
   Getlengths               ===>  YES             Yes | No
     Server Input Length    ===>  _____           0 - 32767 Bytes
     Server Output Length   ===>  _____           0 - 32767 Bytes
     Server Data Format     ===>  CONTIGUOUS      Contiguous | Overlaid
   Register from Data set   ===>  YES             Yes | No





Current Status:


                                     SYSID= CI41  APPLID= IYK1ZFL1
PF1=Help PF2=Delete PF3=Exit PF7=Back PF9=Messages PF12=Return
```

Figure 55. Panels DFHRP21 and DFHRP2B

You can use these panels to delete a 4-tuple definition from the CICS ONC RPC data set by pressing PF2.

If you want to modify the 4-tuple definition, you should first make modifications to panel DFHRP21, and then press PF8 to move to panel DFHRP2B. From panel DFHRP2B you can press PF7 if you want to go back to panel DFHRP21. After you have made your modifications to the panels, you should press Enter to get all the modifications validated, and then press Enter again to get the definition changed.

Processing the alias list

If you select option 4 on panel DFHRP10, panel DFHRP17 is shown.

This panel gives a list of the aliases that have been started, or scheduled, by the server controller, but have not yet ended. Each alias has two lines on the panel.

- The first line shows the 4-tuple for the client request.
- The second line shows the CICS task number of the alias that is processing the client request.

If the alias is scheduled, but not yet started, the task number is blank. If the alias has started, a task number is given and the line is highlighted.

You can enter the following commands against an alias:

P   Purges the alias.

?   Shows panel DFHRP18, which displays details of the alias and the associated client request. (See Figure 57 on page 302.)

        If the alias is scheduled, but not yet started, the task number and start time are blank. If the alias has started, a task number and start time are given.

```
CRPC                       CICS ONC RPC for MVS/ESA                 DFHRP17
                                 Alias List

Enter 'P' to Purge, or '?' to display details of an alias task
  _     Prog( 00000103 ) Vers( 00000114 ) Proc( 00000001 ) Prot( UDP )
        Task Number( 00000033 )
  _     Prog( _____ ) Vers( _____ ) Proc( _____ ) Prot( ___ )
        Task Number( _____ )
  _     Prog( _____ ) Vers( _____ ) Proc( _____ ) Prot( ___ )
        Task Number( _____ )
  _     Prog( _____ ) Vers( _____ ) Proc( _____ ) Prot( ___ )
        Task Number( _____ )
  _     Prog( _____ ) Vers( _____ ) Proc( _____ ) Prot( ___ )
        Task Number( _____ )
  _     Prog( _____ ) Vers( _____ ) Proc( _____ ) Prot( ___ )
        Task Number( _____ )
  _     Prog( _____ ) Vers( _____ ) Proc( _____ ) Prot( ___ )
        Task Number( _____ )
  _     Prog( _____ ) Vers( _____ ) Proc( _____ ) Prot( ___ )
        Task Number( _____ )
Current Status: Enabled

                                    SYSID= CI41  APPLID= IYK1ZFL1
PF1=Help PF2=Refresh PF3=Exit PF7=Back PF8=Forward PF9=Messages PF12=Return
```

Figure 56. Panel DFHRP17

```
CRPC                     CICS ONC RPC for MVS/ESA                    DFHRP18
                          Display Alias Task Details

      Program Number( 00000103 )            Version Number( 00000114 )
      Procedure Number( 00000001 )          Protocol( UDP )
      Task Number( 00000033 )               Client IP Addr( 9.20.2.19       )
      CICS Program Name( RPROC103 )         Transid( CRPA )
      Port Number( 000007BC )               Socket Descriptor( 00000003 )
      Task Start Time( 14:38:19 )           Termid(      )








Current Status: Enabled


                                    SYSID= CI41  APPLID= IYK1ZFL1

PF1=Help  PF3=Exit  PF12=Return
```

Figure 57. Panel DFHRP18

Chapter 25. Programming with CICS ONC RPC

Important: This information contains Product-sensitive Programming Interface and Associated Guidance Information.

This section tells you how to write CICS ONC RPC user-replaceable programs. It describes the general process of development, including details of the interfaces to the converter functions.

Developing an ONC RPC application for CICS ONC RPC

ONC RPC applications are always developed as client/server pairs.

The process described in this section takes account of this, but concentrates on the server, because CICS ONC RPC affects this and not the client. For details of the client development process, read the documentation of the ONC RPC system running on the client machine.

The process of developing all the material needed for an ONC RPC application using CICS ONC RPC is summarized in Figure 58, which should be compared with Figure 32 on page 258, which showed the process for ONC RPC without CICS ONC RPC.
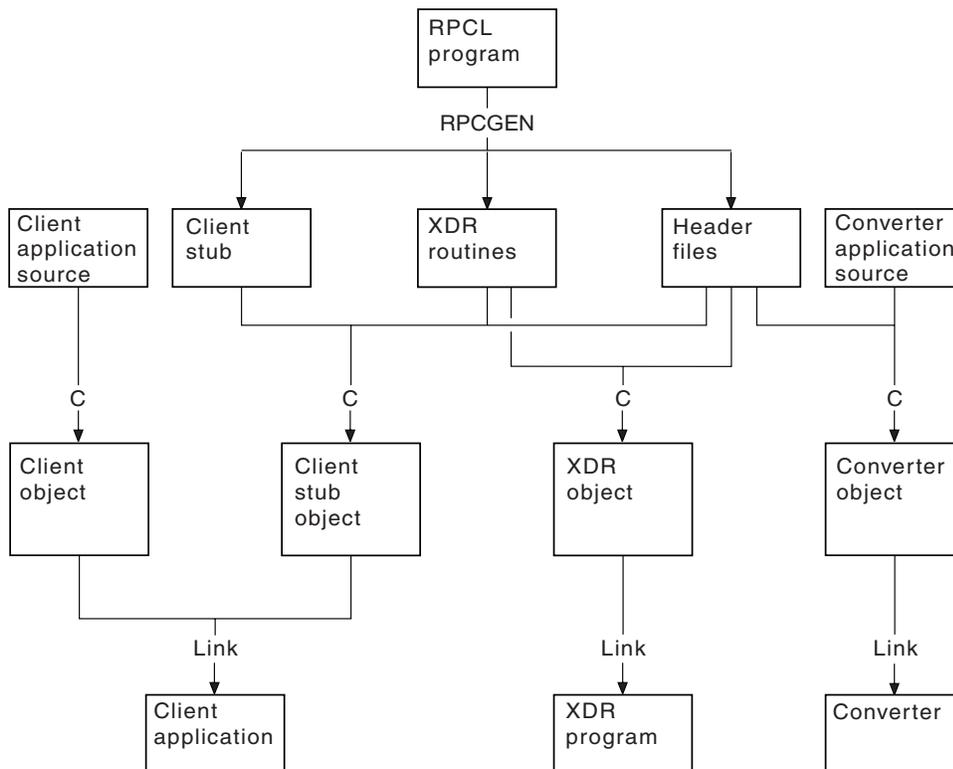
Figure 58. Program development with CICS ONC RPC

The figure shows the development process when RPCGEN is used to create source text from the interface definition in the RPCL program. If you do not use

303

RPCGEN, you must supply some of its output—XDR routines and header files—yourself. The development of the CICS program to service client requests is not shown.

The sequence of development of an ONC RPC application is summarized below. Each step is described in detail in the sections following the summary.

1. Decide what data is to be sent from client to server and what is to be returned. If the data structures the client uses are not simple, you might choose to use RPCGEN to help with managing the data. If you choose to use RPCGEN, some of its output is useful for writing the user-replaceable programs for CICS ONC RPC.

2. Decide the format of the communication area to be used by the CICS program. If the client is to use an existing CICS program, the format is already decided.

3. Write the XDR routines. If the translations you need can be done by an XDR library function supported by the connection manager (see Table 26 on page 305), you do not need to write an XDR routine. If you used RPCGEN, it has generated source for XDR routines. In any other case you must write the XDR routines yourself.

   XDR routines must be written in C.

4. Write the converter. If you used RPCGEN, and you are going to write your converter in C, the header files produced by RPCGEN describe the data structures that **Decode** receives and **Encode** returns. The format of the CICS program communication area is also used by **Decode** and **Encode**.

5. Write the resource checker, if required. You may want to write your own resource checker to validate incoming client requests. Chapter 26, "CICS ONC RPC security," on page 327 tells you about this and other security facilities available for use with CICS ONC RPC. "Writing the resource checker" on page 329 gives you details on writing a resource checker.

6. Compile and link the user-replaceable programs. If you used RPCGEN, the header files are needed for the compilation of the XDR routines and the converter if it is in C.

7. Define the server application set to CICS. This means defining programs for the CICS program, any XDR routines that are not just XDR library functions, and the converter. One or more alias transaction definitions may also be required, see "Defining CICS ONC RPC resources to CICS" on page 275.

8. Use the connection manager to define a 4-tuple and save it in the CICS ONC RPC data set. The definition specifies the CICS program, XDR routines, and converter, as described in "Defining the attributes of a 4-tuple" on page 287.

Step 1—Decide what data is to be sent

This step is outside the scope of this manual.

What you do depends on the nature of the data to be sent with the request and with the reply. Defining data with RPCL and the use of RPCGEN are described in Sun Microsystems' publication *Network Programming*.

Step 2—Decide the format of the communication area

This step is also outside the scope of this manual.

You are reminded that if the CICS program that services a client request is not in the same CICS region as CICS ONC RPC, the maximum communication area length is 35 000 bytes. If the CICS program resides in a server other than CICS Transaction Server for z/OS, other restrictions might also apply.

Step 3—Write the XDR routines

If you used RPCGEN in Step 1, you use the XDR source programs generated by RPCGEN. If the XDR source uses the **xdr_char** or **xdr_u_char** XDR library functions, you must use the C `#define` directive to make the compiler use the **xdr_text_char** function instead.

If the translations you need can be done by an XDR library function supported by the connection manager (see Table 26), you do not need to write an XDR routine. Instead you specify one of the XDR library functions described below when you register a 4-tuple with the connection manager.

If you write your own XDR routine, you need to use the XDR library functions. The full C definitions of these functions are documented in the *z/OS Communications Server: Programmer's Reference* .

CICS ONC RPC supports only the functions listed below. You should use only these functions in your own XDR routines. These functions convert C data types to XDR formats, and XDR formats to C data types.

Some of these function names cannot be used in the connection manager when specifying XDR library functions for the inbound and outbound XDR routines for a 4-tuple. In the column headed **CM**, an asterisk means that the XDR library routine can be specified in the connection manager, while a blank means that it cannot.

Table 26. Supported XDR library functions

XDR library function	CM	C type
xdr_int	*	int
xdr_u_int	*	unsigned int
xdr_long	*	long
xdr_u_long	*	unsigned long
xdr_short	*	short int
xdr_u_short	*	unsigned short int
xdr_float	*	float
xdr_bool	*	bool_t (see note)
xdr_double	*	double
xdr_enum		enum
xdr_void	*	void
xdr_array		variable-length array
xdr_opaque		fixed-length uninterrupted data
xdr_bytes		variable-length array of bytes
xdr_pointer		object references, including null pointers
xdr_reference		object references
xdr_char	*	character
xdr_u_char	*	unsigned character
xdr_text_char	*	text character
xdr_string		null-terminated character arrays
xdr_vector		fixed-length array with arbitrary element size
xdr_wrapstring	*	variable-length null-terminated character arrays
xdr_union		discriminated union

Note: bool_t is not a built-in C data type; it is defined in an ONC RPC header (as a C **int**).

Names of user-written XDR routines are subject to the same restrictions as CICS programs.

You must take care when writing your own XDR routines. These run in the CICS address space and can overwrite CICS code and other user application storage, because they are defined with EXECKEY(CICS).

Code page conversions

Conversion between ASCII and EBCDIC (or vice versa) is done by XDR library functions supplied as part of z/OS Communications Server.

The relevant XDR routines are **xdr_text_char**, **xdr_string**, and **xdr_wrapstring**. These routines use EBCDIC-to-ASCII and ASCII-to-EBCDIC translate tables, which are loaded at z/OS Communications Server initialization from a data set containing one of the possible translate tables provided with z/OS Communications Server.

Thus all ONC RPC requests from all clients use the same translate table. There is no provision for ONC RPC data from different client workstations or from different client end users to have different character sets.

Various single-byte character set (SBCS) translate tables are provided with z/OS Communications Server, one of which is generated during z/OS Communications Server customization. If none of these is suitable, you could provide your own, as described in the *z/OS Communications Server: Customization and Administration Guide*.

z/OS Communications Server provides several code pages for double-byte character sets (DBCS). If you want to include DBCS in ONC RPC data you have to write your own XDR routines to convert the double-byte characters.

Step 4—Write the converter

Write the converter as described in "Write the CICS ONC RPC converter" on page 307, using reference information supplied in "Reference information for the converter functions" on page 315.

Step 5—Write a resource checker

This step is optional.

See "Writing the resource checker" on page 329 for details.

Step 6—Compile and link

This step puts the programs you have written into CICS load libraries.

Converter

The header files needed to compile the converter are discussed in "Organizing the converter" on page 309.

The program is linked into a CICS load library, since it is a normal CICS program.

XDR routines

If your XDR routines are not just XDR library functions, you must compile each XDR routine separately and link it into a CICS load library. If you used RPCL to define the data, the XDR source and header files for the compilation have been generated by RPCGEN.

Resource checker

If you need a resource checker, you must link it into a CICS load library. It must be called DFHRPRSC.

Step 7—Make CICS definitions

You must define the CICS program, converter program, resource checker, and any XDR routines that are not just library routines to CICS.

See "Defining CICS ONC RPC resources to CICS" on page 275.

Step 8—Make a connection manager entry

Use the connection manager to define each 4-tuple. Completing an entry for a 4-tuple in the connection manager ensures that you provide CICS ONC RPC with all the information that it needs to service the client request.

The fields used to define each 4-tuple are described in "Defining the attributes of a 4-tuple" on page 287.

Write the CICS ONC RPC converter

This section describes how you can write a converter to perform various tasks. Some of these tasks are required for all 4-tuples, others only for some.

The section describes in turn each of the tasks, indicating the converter function (**Getlengths**, **Decode**, or **Encode**) used.

The parameter details and responses of each of the converter functions are given at the end of the section in "Getlengths" on page 316, "Decode" on page 318, and "Encode" on page 323.

Tasks that can be performed by a converter

The tasks to be performed are:

- Telling the connection manager or the server controller the lengths of the input and output data for the CICS program
- Telling the connection manager the CICS program data format
- Mapping data between client and CICS program formats, as illustrated in Figure 37 on page 269 and Figure 39 on page 270
- Telling the server controller which alias and CICS program are to be used to service a request, if those specified when the 4-tuple was defined are to be changed

Lengths of the CICS program input and output data

CICS ONC RPC needs to know the length of the CICS program input and output data for each 4-tuple.

For each 4-tuple, the lengths may be defined in one of three places:

- In the connection manager if the lengths do not vary from call to call. You specify the lengths in the connection manager and specify NO for the Getlengths attribute of the 4-tuple. In this case **Getlengths** is not called.
- In **Getlengths** if the lengths do not vary from call to call, returning the values in the **glength_server_input_data_len** and **glength_server_output_data_len** output fields. In the connection manager you specify YES for the Getlengths attribute of the 4-tuple, and leave the length fields blank.

In either of these first two cases, if **Decode** is specified for the 4-tuple, the **Decode** function can change the lengths.

- In **Decode**, if the lengths of the data structures vary from call to call. You return the lengths on each call by using the **decode_server_input_data_len** and **decode_server_output_data_len** output fields. The lengths specified with the connection manager or **Getlengths** are supplied as inputs to **Decode** in these fields.

Setting the CICS program data format

CICS ONC RPC needs to know the CICS program data format for each 4-tuple.

You can set this either in **Getlengths** or in the connection manager. If you choose **Getlengths**, use the output field **glength_server_data_format**. The value specified with the connection manager is supplied as input to **Getlengths** in this field.

Mapping data between client and CICS program formats

You need to map the incoming data intended for the CICS program only if it is not in the format required by the CICS program.

This is typically for:
- Client data structures that contain pointers to other data. These are rebuilt by the inbound XDR routine in the same form as they existed in the client. The data for the CICS program must be copied into a single area of storage to be passed to the CICS program as its communication area.
- CICS programs that are written in a language other than C. The incoming client request always has a C data structure. If your CICS program is written in COBOL, for example, you need to perform a C-to-COBOL mapping in **Decode**.

The mapping is always done by **Decode** for the input data for the CICS program. In most cases, the output data needs to be mapped in the opposite direction by **Encode**.

On input, the client data is pointed to by the **Decode** input field **decode_client_data_ptr**. **Decode** maps this data into the form which the CICS program requires.

To achieve the mapping, **Decode** must allocate an area of CICS storage, using EXEC CICS GETMAIN SHARED. **Decode** must set the output field **decode_returned_data_ptr** to the address returned by the GETMAIN command, and put the input data passed from the client into the storage, making changes where applicable.

Changing the alias and CICS program

You can use **Decode** to redirect a client request to another CICS program.

CICS ONC RPC then ignores the original program name that was defined in the connection manager for the requested 4-tuple. To reroute a client request, specify a new CICS program name in the **decode_server_program** field in **Decode**. This facility allows a client to pass a CICS program name in the data it sends in the remote procedure call. The new CICS program must work with the same communication area format, converter, and XDR output routine as the original program.

You can use **Decode** to change the name of the alias transaction to run the CICS program by setting the **decode_alias_transid** output field. CICS ONC RPC then

ignores the transaction ID that was defined in the connection manager for the requested 4-tuple. This facility allows a client to pass the alias transaction ID in the data it sends with the remote procedure call.

Changing security information

You may want your CICS ONC RPC system to implement security checking on incoming client requests. Such checking usually involves checks on the client user ID and password. One of the ways the client can provide these is by including them in the data structure it sends.

Decode can retrieve this information from the incoming data, and return it in the output fields. The user ID should be returned in the output field **decode_userid**; the password should be returned as part of the data pointed to by the **decode_returned_data_ptr** field. These outputs can either be passed by the client or generated by **Decode** in whatever way you want. For instance, **Decode** can derive the CICS user ID and password for the client request by using the **decode_client_address** field, or the authentication fields **decode_aup_...** that identify the client.

Organizing the converter

You can write converters for any CICS-supported compiler. If you choose a language other than C or COBOL, you must write your own header files to define the CICS ONC RPC data structures and constants.

A converter is passed a communication area that contains a parameter that specifies which of the three functions **Getlengths**, **Decode**, or **Encode** is required, and parameters for the particular function, as described in the reference material: "Getlengths" on page 316, "Decode" on page 318, and "Encode" on page 323.

The following C header files (in the SDFHC370 target library) and COBOL copybooks (in the SDFHCOB target library) are provided to help with writing the converter:

- DFHRPUCH for C (DFHRPUCO for COBOL)—contains definitions of the constants that are used in the interface between CICS ONC RPC and the converter.
- DFHRPCDH for C (DFHRPCDO for COBOL)—defines the format of the communication area that is presented to the converter. The communication area is in two parts. The format of the first part is independent of the function that the converter is being asked to perform, and it contains:
  - The eyecatcher for the requested function
  - The function code for the requested function
  - A response to be supplied by the converter
  - A reason code to be supplied by the converter

  The format of the rest of the communication area depends on the converter function requested.

You need a header file produced by RPCGEN only if you used RPCL to define the data structures, and you are writing **Decode** or **Encode**. If you are writing your converter in a language other than C, you need to rewrite the header file in your chosen language, since RPCGEN produces its output only in C.

You need definitions of the CICS structures that you use, and the definition of the CICS program communication area.

Writing a converter in C

The following discussion is based on a converter that consists of four main parts:

- A routing part that consults the function code in the communication area, and then calls the appropriate function
- A function for **Getlengths** processing
- A function for **Decode** processing
- A function for **Encode** processing

Figure 59 shows how you can route control to the appropriate function.

```
EXEC CICS ADDRESS EIB(dfheiptr);        /*Get addressability of EIB*/

EXEC CICS ADDRESS COMMAREA(converter_parms_ptr);

switch(converter_parms_ptr->converter_function) {

 case URP_GETLENGTHS:
 {
   converter_getlengths();
   break;
 }
 case URP_DECODE:
 {
   converter_decode();
   break;
 }
 case URP_ENCODE:
 {
   converter_encode();
   break;
 }

 default:
 {
   converter_parms_ptr->converter_response = URP_INVALID;
 }

 }   /* end switch */

 EXEC CICS RETURN;

} /* end main */
```

Figure 59. Routing control to the functions in C

In this program fragment, `converter_parms_ptr` is a locally declared pointer to the converter_parms structure declared in DFHRPCDH. All the other names beginning converter_ are names from this structure.

The processing is as follows:

1. The `converter_parms_ptr` pointer is set by using EXEC CICS ADDRESS COMMAREA.
2. The **switch** statement is used to select the function to be called. If you are not providing all the functions, you need fewer **case** statements.
3. If the function is not valid, the response URP_INVALID is returned from the converter. This test is always advised, especially if the converter does not provide all three functions.

Figure 60 is an example of a **Decode** function.

```
void converter_decode(void)
{
  decode_parms *decode_parms_ptr;

  decode_parms_ptr = (decode_parms *)converter_parms_ptr;

  if (strncmp
  (decode_parms_ptr->decode_eyecatcher,DECODE_EYECATCHER_INIT,8)
  == 0)
  {
    EXEC CICS GETMAIN
      SET(decode_parms_ptr->decode_returned_data_ptr)
      FLENGTH(sizeof(rem_proc_parms_103) + PW_LEN)
      SHARED
      NOSUSPEND
      CICSDATAKEY
      RESP(response)
      RESP2(response2);

    if (response != DFHRESP(NORMAL))
    {
      memcpy(outline,errmsg1,strlen(errmsg1));
      EXEC CICS WRITEQ TD QUEUE(tdq) FROM(outline) LENGTH(30);
      decode_parms_ptr->decode_response = URP_EXCEPTION;
      decode_parms_ptr->decode_reason = NO_STORAGE;
    }
    else
    {
      /* move password and data to decode_password and
      decode_server_input_data */

      decode_parms_ptr->decode_response = URP_OK;
    };
  }
  else
    decode_parms_ptr->decode_response = URP_INVALID;
}
```

Figure 60. Example of a Decode function in C

In this program fragment, names beginning decode_, except decode_parms_ptr, are names from the decode_parms structure defined in DFHRPCDH.

The processing is as follows:

1. The pointer decode_parms_ptr is set from converter_parms_ptr.
2. The eyecatcher is checked to see if it agrees with the function code. If it does:
   a. EXEC CICS GETMAIN is used to get storage for the password and for the communication area to be passed to the CICS program. The value of PW_LEN is set elsewhere in the program to 8 by #define. The output parameter decode_returned_data_ptr is used directly in the GETMAIN. In this case there is no conversion of data to be done, and the communication area size is the same as the size of the client data structure. (rem_proc_parms_103 is a structure that defines the input data after XDR conversion.)
   b. If the response to the EXEC CICS GETMAIN is not NORMAL, an error message is directed to a transient data queue, the converter response is set to URP_EXCEPTION, and the reason code is set to NO_STORAGE, which is locally declared.

c. If the response to the EXEC CICS GETMAIN is NORMAL, the data and password are transferred to the storage acquired by GETMAIN (not shown), and the converter response is set to URP_OK.

3. If the eyecatcher is not the one for the function being called, the converter response is set to URP_INVALID.

Writing a converter in COBOL

In the working storage section of the data division, you should use the COPY statement to copy the copybook DFHRPUCO, and any other copybooks you need. You should also define any other data items you need in working storage.

You use the COPY statement to include the definition of the communication area in the linkage section of the data division.

Figure 61 on page 313 shows the layout of the data division. Comments, which would be part of a well-documented converter, are omitted.

The following discussion is based on a converter that consists of four main parts:

- A routing part that consults the function code in the communication area, and then calls the appropriate function
- A function for **Getlengths** processing
- A function for **Decode** processing
- A function for **Encode** processing

Figure 62 on page 314 shows how you can route control to the appropriate function.

```
DATA DIVISION.

WORKING-STORAGE SECTION.

    COPY DFHRPUCO.

01  RESP                      PIC S9(8) COMP.
01  RESP2                     PIC S9(8) COMP.
01  REM-PROC-COMMSIZE         PIC S9(8) COMP VALUE +12.
01  CLIENT-OUT-SIZE           PIC S9(8) COMP VALUE +8.

LINKAGE SECTION.

 01 DFHCOMMAREA.
   02 COMM-PARMLIST PIC X(1).

 01 CONVERTER-PARMS REDEFINES DFHCOMMAREA.
   02 CONVERTER-EYECATCHER PIC X(8).
   02 CONVERTER-FUNCTION PIC 9(8) COMP.
   02 CONVERTER-RESPONSE PIC 9(8) COMP.
   02 CONVERTER-REASON PIC 9(8) COMP.
   02 CONVERTER-PARMLIST PIC X(1).

 01 GLENGTH-PARMS REDEFINES DFHCOMMAREA.
   02 GLENGTH-EYECATCHER PIC X(8).
   02 GLENGTH-FUNCTION PIC 9(8) COMP.
   02 GLENGTH-RESPONSE PIC 9(8) COMP.
   02 GLENGTH-REASON PIC 9(8) COMP.
   02 GLENGTH-SERVER-INPUT-DATA-LEN PIC S9(8) COMP.
   02 ...

 01 DECODE-PARMS REDEFINES DFHCOMMAREA.
   02 ...

 01 DECODE-RETURNED-DATA.
   02 DECODE-PASSWORD PIC X(8).
   02 DECODE-SERVER-INPUT-DATA PIC X(1).

 01 ENCODE-PARMS REDEFINES DFHCOMMAREA.
   02 ...
```

Figure 61. Layout of data division in COBOL

```
PROCEDURE DIVISION.

A-CONTROL SECTION.

A-0000-MAIN-TASK.

    MOVE URP-INVALID TO DECODE-RESPONSE.

    IF CONVERTER-FUNCTION = URP-GETLENGTHS
    PERFORM B-0000-GETLENGTHS END-IF.

    IF CONVERTER-FUNCTION = URP-DECODE THEN
    PERFORM C-0000-DECODE END-IF.

    IF CONVERTER-FUNCTION = URP-ENCODE THEN
    PERFORM D-0000-ENCODE END-IF.

A-9999-EXIT.

    EXEC CICS RETURN END-EXEC.
    GOBACK.
```

Figure 62. Routing control to the functions in COBOL

In this program fragment:

1. The response URP-INVALID is set.
2. The IF statements examine the function code in the communication area, and pass control to the appropriate function.
3. The converter returns to the program that called it. (If the IF statements selected a function, the DECODE-RESPONSE value returned is the response from that function.)

Figure 63 is an example of a **Decode** function.

```
C-0000-DECODE.

    IF DECODE-EYECATCHER IS NOT = DECODE-EYECATCHER-INIT
      MOVE URP-INVALID TO DECODE-RESPONSE
    ELSE
      SET ADDRESS OF CLIENT-IN-DATA TO DECODE-CLIENT-DATA-PTR
      ADD 8 TO REM-PROC-COMMSIZE
      EXEC CICS GETMAIN
                    SET(DECODE-RETURNED-DATA-PTR)
                    FLENGTH(REM-PROC-COMMSIZE)
                    SHARED
                    NOSUSPEND
                    CICSDATAKEY
                    RESP(RESP)
                    RESP2(RESP2)
                    END-EXEC
    SET ADDRESS OF DECODE-RETURNED-DATA
              TO DECODE-RETURNED-DATA-PTR
    MOVE "PASSWD" TO DECODE-PASSWORD
    SET ADDRESS OF REM-PROC-DATA
              TO ADDRESS OF DECODE-SERVER-INPUT-DATA
    MOVE CLIENT-IN-U-CHAR TO REM-PROC-U-CHAR
    MOVE CLIENT-IN-CHAR TO REM-PROC-CHAR
    MOVE URP-OK TO DECODE-RESPONSE.
```

Figure 63. Example of a Decode function in COBOL

In this program fragment, the names beginning DECODE- (except DECODE-PASSWORD) are fields in the communication area for the **Decode** function. DECODE-PASSWORD is the field at the beginning of the returned data. The processing is as follows:

1. The eyecatcher is checked to see if it agrees with the function code. If it does not, the URP-INVALID response is returned.

2. If it does:

   a. The structure CLIENT-IN-DATA is overlaid on the data coming from the inbound XDR routine addressed by DECODE-CLIENT-DATA-PTR.

   b. The communication area size is increased by 8 to allow for the password field.

   c. EXEC CICS GETMAIN is used to get storage for the password and for the communication area. REM-PROC-COMMSIZE is the size of the structure REM-PROC-DATA, which defines the format of the communication area. The address of the storage is put directly into DECODE-RETURNED-DATA-PTR.

   d. The structure DECODE-RETURNED-DATA is overlaid on the newly-acquired storage addressed by DECODE-RETURNED-DATA-PTR.

   e. The password is moved into DECODE-PASSWORD.

   f. The data is moved from CLIENT-IN-DATA to REM-PROC-DATA, and the response is set to URP-OK.

Using converters

Converters run as CICS programs under the connection manager, server controller, and aliases. Converters must reside in the same CICS system as CICS ONC RPC.

Preparation

Before using a converter, you must:

1. Translate the converter using the appropriate CICS translator. If it is a COBOL program, you must use the QUOTE translator directive.

2. Compile the output from the translator.

3. Link the converter as a standard CICS application program into a CICS load library used by the CICS system on which CICS ONC RPC is installed.

4. Define the converter to CICS as a program.

5. Use the connection manager to specify the converter in one of the 4-tuple definitions, and define which of the converter functions are required for that 4-tuple.

Reference information for the converter functions

This section contains reference material for each of the three functions of a converter.

Each function is documented in the same way:

- A summary table of parameters, showing which are for input only, which for input and output, and which for output only.

  – **Input** is for parameters that your function may consult, but not change.

  – **Inout** is for parameters that your function may consult, and change.

  – **Output** is for parameters that your function must not consult, but may change.

- A description of the processing that the function 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 converter 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.

Getlengths

Getlengths is called when the definition of the 4-tuple is being registered.

Getlengths is called when the definition of the 4-tuple is being registered, provided that the definition of the 4-tuple specified that **Getlengths** was to be called. It is not called to process client requests.**Getlengths** is responsible for providing CICS ONC RPC with:

- The size of the data that is passed to and from the CICS program
- The data format (contiguous or overlaid) of the CICS program data

Summary of parameters

The names of the parameters are given in abbreviated form: each name in the table must be prefixed with **glength_** to give the name of the parameter.

To find the C type of each parameter, consult the header file DFHRPCDH provided with CICS ONC RPC. For COBOL, consult the copybook DFHRPCDO.

Input glength_	Inout glength_	Output glength_
eyecatcher function	server_data_format	server_input_data_len server_output_data_len response reason

Parameters

glength_eyecatcher
>   (Input only)

>   A string of length 8. (The values of the eyecatchers are defined in the DFHRPUCH header file and the DFHRPUCO copybook.)

glength_function
>   (Input only)

>   A code indicating that **Getlengths** is being called. The value is URP_GETLENGTHS.

glength_reason
>   (Output only)

>   A reason code—see "Response and reason codes" on page 317.

glength_response
>   (Output only)

>   A response code—see "Response and reason codes" on page 317.

glength_server_data_format

> (Input and output)

> On input, that value specified for Server Data Format for the 4-tuple in the connection manager.

> On output, the value is to control:
> - How the input data pointer for **Encode** will be set up
> - How the communication area length to be checked by the connection manager is calculated

> The values you can supply are as follows:

> **URP_CONTIGUOUS**

>> The value of the data pointer that will be passed to **Encode**, or to the outbound XDR routine if **Encode** is not used for this 4-tuple, is the address of the CICS program communication area plus the output value of **glength_server_input_data_len**, though **Decode** can modify this offset.

>> The connection manager calculates a communication area length by adding the output values of **glength_server_input_len** and **glength_server_output_len**. If this length is different from the actual length of the communication area passed to the CICS program, errors might occur in the processing of client requests.

> **URP_OVERLAID**

>> The value of the data pointer that will be passed to **Encode**, or to the outbound XDR routine if **Encode** is not used for this 4-tuple, is the address of the CICS program communication area.

>> The connection manager calculates a communication area length by taking the larger of the output values of **glength_server_input_len** and **glength_server_output_len**. If this length is different from the actual length of the communication area passed to the CICS program, errors might occur in the processing of client requests.

glength_server_input_data_len

> (Output only)

> For the use of this field, see the description of **glength_server_data_format**. If you do not set a value in this field, a default value of zero is used.

glength_server_output_data_len

> (Output only)

> For the use of this field, see the description of **glength_server_data_format**. If you do not set a value in this field, a default value of zero is used.

Response and reason codes

You must return one of the following values in the **glength_response** field:

URP_OK
> The connection manager checks that the communication area length does not exceed 32 767. If it does not, the information is saved and used to process incoming client requests, and the 4-tuple is registered. If it does, the connection manager writes an exception trace entry (trace point 9EE6), sends a message (DFHRP1991) describing the error to the terminal from which the connection manager was started, and does not register the 4-tuple.

URP_EXCEPTION

> The connection manager writes an exception trace entry (trace point 9EE5), sends a message (DFHRP1988) to the terminal from which the connection manager was started, and does not register the 4-tuple.

URP_INVALID

> The connection manager writes an exception trace entry (trace point 9EE5), sends a message (DFHRP1989) to the terminal from which the connection manager was started, and does not register the 4-tuple.

URP_DISASTER

> The connection manager writes an exception trace entry (trace point 9EE5), sends a message (DFHRP1990) to the terminal from which the connection manager was started, and does not register the 4-tuple.

If you return any other value in **glength_response**, it is treated as URP_DISASTER.

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 **Getlengths**. The reason code is output in any trace that results from the invocation of **Getlengths**, and you may use it as a debugging aid.

See "Numeric values of response and reason codes" on page 337 for the numeric values of the response codes in trace output.

Decode

Decode is invoked by the server controller after the inbound XDR routine. **Decode** processing must avoid making the server controller wait for resources, because waiting prevents the server controller from dealing efficiently with other requests.

Decode has four main responsibilities:
- To set data lengths for the CICS program when the lengths are not the same for all requests.
- To map the input data passed from the inbound XDR routine to the input data format required by the CICS program.
- To set the user ID and password that are used to control subsequent processing.
- To set the name of the alias and CICS program for the request if those specified for the 4-tuple need to be changed.

Summary of parameters

The names of the parameters are given in abbreviated form; each name in the table must be prefixed with **decode_** to give the name of the parameter.

To find the C type of each parameter, consult the header file DFHRPCDH provided with CICS ONC RPC. For COBOL, consult the copybook DFHRPCDO.

Input decode_	Inout decode_	Output decode_
eyecatcher	server_program	returned_data_ptr
function	alias_transid	userid
client_address	server_input_data_len	user_token
client_data_ptr	server_output_data_len	response
server_data_format		reason
program_number		
version_number		
procedure_number		
aup_time		
aup_machname_ptr		
aup_machlen		
aup_uid		
aup_gid		
aup_len		
aup_gids_ptr		

Decode must issue an **EXEC CICS GETMAIN** command to allocate storage for the communication area to be passed to the CICS program. Note the following points about GETMAIN options:

- You must use the SHARED option, because the storage is acquired under the server controller, but is used under the alias.
- You must use the FLENGTH option.
- You must use the NOSUSPEND option to prevent the server controller from being made to wait for storage, because waiting prevents the server controller from attending to incoming requests.
- To prevent overwriting by user-key programs, consider using the CICSDATAKEY option in the following circumstances:
  - The CICS program to be called by the alias is in another CICS system.
  - The CICS program to be called by the alias is defined as EXECKEY(CICS).
  - The CICS program to be called by the alias is defined as EXECKEY(USER), but the amount of data to be copied is small.

If an overlaid data format is specified, the requested length must be the greater of the output values of **decode_server_input_data_len** and **decode_server_output_data_len** plus 8 for DECODE-PASSWORD. If the data format is not overlaid, this length must be the sum of the output values of **decode_server_input_data_len** and **decode_server_output_data_len** plus 8 for DECODE-PASSWORD.

Because **Decode** specifies the SHARED option, the data remains available to CICS ONC RPC modules and to CICS programs. CICS ONC RPC frees the storage when it is no longer required.

Parameters

decode_alias_transid

(Input and output)

On input, the name of the alias associated with the 4-tuple for the client request.

On output, the name of the transaction to be started by the server controller to process this client request.

See "Changing the alias and CICS program" on page 308.

decode_aup_gid
(Input only)

The UNIX group id of the client.

decode_aup_gids_ptr
(Input only)

A pointer to an array of 32-bit integers that are the UNIX group IDs of which the client is a member.

decode_aup_len
(Input only)

The number of elements in the array of UNIX group identifiers pointed to by **decode_aup_gids_ptr**.

decode_aup_machlen
(Input only)

The number of characters in the machine name.

decode_aup_machname_ptr
(Input only)

A pointer to a variable-length character string representing the name of the machine on which the client is running.

decode_aup_time
(Input only)

The time at which the client created the credentials. The time is measured in seconds since 00h00m GMT on 1 January 1970.

decode_aup_uid
(Input only)

The UNIX user ID of the client.

decode_client_address
(Input only)

The 32-bit internet address of the client from which the request was received.

decode_client_data_ptr
(Input only)

A pointer to the data passed from the client. If no data exists, this pointer points to a null string.

Note: The data area pointed to by this pointer must not be changed by **Decode**, because CICS storage management errors are likely to occur.

decode_eyecatcher
(Input only)

A string of length 8. The values of the eyecatchers are defined in the DFHRPUCH header file and the DFHRPUCO copybook.

decode_function
(Input only)

A code indicating that **Decode** is being called. The value is URP_DECODE.

decode_procedure_number
>   (Input only)

>   The procedure number of the 4-tuple to which the client request was made.

decode_program_number
>   (Input only)

>   The program number of the 4-tuple to which the client request was made.

decode_reason
>   (Output only)

>   A reason code; see "Response and reason codes" on page 322.

decode_response
>   (Output only)

>   A response code; see "Response and reason codes" on page 322.

decode_returned_data_ptr
>   (Output only)

>   A pointer to an area of storage allocated by the converter that contains these fields:

>   • **decode_password**: the password to be used for user authentication
>   • **decode_server_input_data**: the data that is to be passed to the CICS program as input.

>   The pointer might be null if no password exists and if no data is to be passed to the CICS program.

decode_server_data_format
>   (Input only)

>   A value that controls these operations:

>   • How the input data pointer for **Encode** will be set up
>   • How the communication area length to be checked by the connection manager is calculated

>   **URP_CONTIGUOUS**
>   >   The value of the data pointer that will be passed to **Encode**, or to the outbound XDR routine if **Encode** is not used for this 4-tuple, is the address of the CICS program communication area plus the output value of **decode_server_input_data_len**.

>   >   The server controller calculates a communication area length by adding the output values of **decode_server_input_data_len** and **decode_server_output_data_len**. If this length is different from the actual length of the communication area passed to the CICS program, errors might occur in the processing of client requests.

>   **URP_OVERLAID**
>   >   The value of the data pointer that will be passed to **Encode**, or to the outbound XDR routine if **Encode** is not used for this 4-tuple, is the address of the CICS program communication area.

>   >   The server controller calculates a communication area length by taking the larger of the output values specified of **decode_server_input_data_len** and **decode_server_output_data_len**. If this length is different from the

length of the communication area passed to the CICS program, errors might occur in the processing of client requests.

decode_server_input_data_len
(Input and output)

On input, the output value of **glength_server_input_data_len**, or the value specified for Server Input Length for this 4-tuple in the connection manager.

On output, see the description of **decode_server_data_format**.

decode_server_output_data_len
(Input and output)

On input, the output value of **glength_server_output_data_len**, or the value specified for Server Output Length for this 4-tuple in the connection manager.

On output, see the description of **decode_server_data_format**.

decode_server_program
(Input and output)

On input, the name of the CICS program associated with the 4-tuple for the client request.

On output, the name of the CICS program to be linked to by the alias.

Use this field if you want to direct the client call to a different CICS program.

decode_userid
(Output only)

An 8-character field, the user ID known to CICS that correlates to the requesting client ID. If you store no value in this field, the user ID used in subsequent processing is the default CICS user ID.

decode_user_token
(Output only)

A fullword that can be used to pass information to the **Encode** function that is subsequently invoked for the client request.

decode_version_number
(Input only)

The version number of the 4-tuple to which the client request was made.

Response and reason codes

You must return one of the following values in the **decode_response** field:

URP_OK
The server controller checks that the communication area length does not exceed 32 767. If it does not, the alias is started using the information supplied as output. If it does, the server controller writes an exception trace entry (trace point 9FC2) and issues a message (DFHRP0516) describing the error. The alias is not started, and an **svcerr_systemerr** call is used to send a reply to the client.

URP_EXCEPTION
The server controller writes an exception trace entry (trace point 9FAA), and issues a message that depends on the reason code:

- URP_CORRUPT_CLIENT_DATA: message DFHRP0626

  An **svcerr_decode** call is used to send a reply to the client.
- URP_AUTH_BADCRED: message DFHRP0628

  An **svcerr_auth** call with a why-value of AUTH_BADCRED is used to send a reply to the client.
- URP_AUTH_TOOWEAK: message DFHRP0629

  An **svcerr_auth** call with a why-value of AUTH_TOOWEAK is used to send a reply to the client.
- Any other value: message DFHRP0631

  An **svcerr_systemerr** call is used to send a reply to the client.

URP_INVALID

The server controller writes an exception trace entry (trace point 9FAA) and issues a message (DFHRP0632).

An **svcerr_systemerr** call is used to send a reply to the client.

URP_DISASTER

The server controller writes an exception trace entry (trace point 9FAA) and issues a message (DFHRP0635).

An **svcerr_systemerr** call is used to send a reply to the client.

If you return any other value in **decode_response**, the server controller writes an exception trace entry (trace point 9FAA) and issues a message (DFHRP0625). An **svcerr_systemerr** call is used to send a reply to the client.

You can supply a 32-bit reason code 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 **Decode**, except as indicated above under URP_EXCEPTION. The reason code is included in any trace that results from the invocation of **Decode**, and you can use it as a debugging aid.

See "Numeric values of response and reason codes" on page 337 for the numeric values of the response and CICS-defined reason codes in trace output.

Encode

Encode is called by the alias after the CICS program ends. **Encode** is responsible for taking the data returned from the CICS program and changing its format so that it is suitable to be passed to the outbound XDR routine for return to the client.If no restructuring of outbound data is required, you can specify to the connection manager that **Encode** is not to be called.The reference to the CICS program data to be returned to the client is passed to **Encode** in the **encode_input_data_ptr** input field. This data is in CICS program format, which is a communication area structure in any CICS supported language. The CICS program data may be mapped from this format into the format required by the client, which is likely to be C, and might include pointer references, by allocating an area of storage and mapping the server data into it.**Encode** must set **encode_output_data_ptr** to point to the start of the allocated storage.

Summary of parameters

The names of the parameters are given in abbreviated form: each name in the table must be prefixed with **encode_** to give the name of the parameter.

To find the C type of each parameter, consult the header file DFHRPCDH provided with CICS ONC RPC. For COBOL, consult the copybook DFHRPCDO.

Input encode_	Inout encode_	Output encode_
eyecatcher function input_data_ptr input_data_len user_token	none	output_data_ptr output_data_len response reason

Encode must issue EXEC CICS GETMAIN to allocate storage for the data that it returns. Note the following points about GETMAIN options:

- You do not need to use the SHARED option.
- You must use the FLENGTH option.
- If your CICS system is using storage protection, you can use the CICSDATAKEY option to prevent overwriting by user-key programs.

Parameters

encode_eyecatcher
> (Input only)
>
> A string of length 8. (The values of the eyecatchers are defined in the DFHRPUCH header file and the DFHRPUCO copybook.)

encode_function
> (Input only)
>
> A code indicating that **Encode** is being called. The value is URP_ENCODE.

encode_input_data_len
> (Input only)
>
> The length in bytes of the data returned from the CICS program. The value is determined as follows:
> 1. It is the output value of **decode_server_output_data_len**, if **Decode** set it.
> 2. If **Decode** did not set the value, it is the output value of **glength_server_output_data_len**, if **Getlengths** was called when the 4-tuple was registered.
> 3. If neither of the above is the case, it is the value specified for Server Output Length in the connection manager when the 4-tuple was defined.

encode_input_data_ptr
> (Input only)
>
> A pointer to the data returned from the CICS program. The setting of this pointer depends on the definition of the 4-tuple in the connection manager, **Getlengths** processing when the 4-tuple was registered, and **Decode** processing for the client request.

encode_output_data_len
> (Output only)
>
> The length in bytes of the data to be passed to the outbound XDR routine.

encode_output_data_ptr
> (Output only)

A pointer to an area of allocated storage that contains the data that is to be passed to the outbound XDR routine.

encode_reason
(Output only)

A reason code—see "Response and reason codes."

encode_response
(Output only)

A response code—see "Response and reason codes."

encode_user_token
(Input only)

A fullword containing information which was output from **Decode** for this client request.

Response and reason codes

You must return one of the following values in the **encode_response** field:

URP_OK
The alias passes the output data to the outbound XDR routine.

URP_EXCEPTION
The alias writes an exception trace entry (trace point 9F17), and issues a message (DFHRP0161). An **svcerr_systemerr** call is used to send a reply to the client.

URP_INVALID
The alias writes an exception trace entry (trace point 9F17), and issues a message (DFHRP0162). An **svcerr_systemerr** call is used to send a reply to the client.

URP_DISASTER
The alias writes an exception trace entry (trace point 9F17), and issues a message (DFHRP0169). An **svcerr_systemerr** call is used to send a reply to the client.

If you return any other value in **encode_response**, the alias writes an exception trace entry (trace point 9F17), and issues a message (DFHRP0163). 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 **Encode**. The reason code is output in any trace that results from the invocation of **Encode**, and you may use it as a debugging aid.

See "Numeric values of response and reason codes" on page 337 for the numeric values of the response in trace output.

Chapter 26. CICS ONC RPC security

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.

It covers the following topics:
- "Security in CICS and its effect on CICS ONC RPC operations"
- "Writing the resource checker" on page 329.

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 64 shows how CICS security interacts with the operation of CICS ONC RPC.
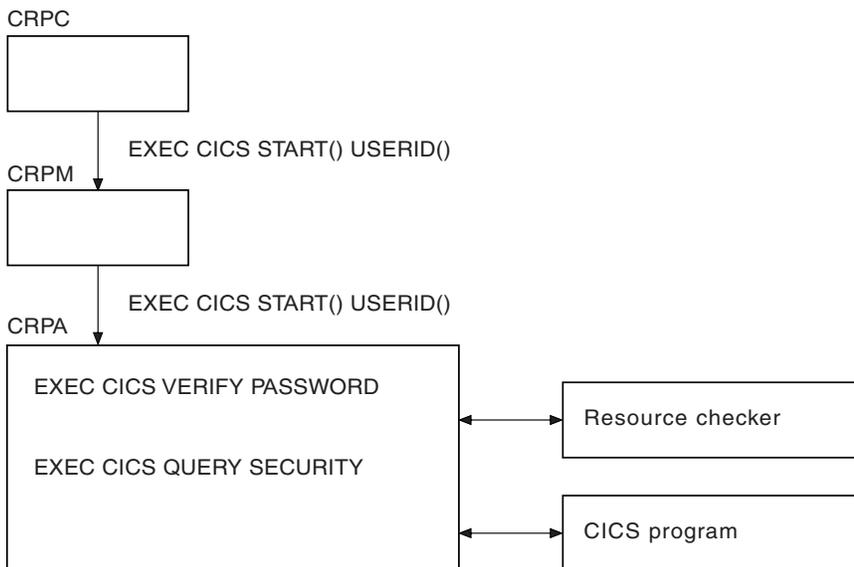
Figure 64. How CICS security interacts with CICS ONC RPC operations

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.

Using RACF Secured Sign-on

RACF Secured Sign-on support allows clients to gain security access to CICS facilities by sending a PassTicket (that is, a one-time-only password). This avoids the security hazard of a password being transmitted across the network in clear text.

For further information, see *Resource Access Control Facility: System Programmer's Guide, Version 2 Release 2*. This includes details of the algorithm that the RPC client must use to generate the PassTicket. This algorithm includes the DES algorithm.

PassTicket generation

The algorithm that generates the PassTicket is a function of:
* 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 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
above 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" on page 337 for the numeric
values of the response and CICS-defined reason codes in trace output.

Chapter 27. CICS ONC RPC problem determination

This section helps you debug problems in CICS ONC RPC user-replaceable programs, the IBM-supplied parts of CICS ONC RPC, and in the system setup of CICS ONC RPC.

If you suspect that you have a problem in another part of CICS, see .

The formats of messages and trace outputs in CICS ONC RPC are also described.

Diagnostic information is designed to provide first failure data capture, so that if an error occurs, enough information about the error is available directly without having to reproduce the error situation. The information is presented in the following forms:

Messages
> CICS ONC RPC provides CICS messages. The CICS ONC RPC messages have the prefix DFHRP and are listed in CICS messages and codes overview in Messages and Codes Vol 2.

Trace CICS ONC RPC outputs system trace entries containing all the important information required for problem diagnosis.

Dump Dump formatting is provided for data areas relating to CICS ONC RPC.

Abend codes
> Transaction abend codes are standard 4-character names. The abend codes output by CICS ONC RPC are listed in CICS messages and codes overview in Messages and Codes Vol 1.

The rest of this chapter describes:
- "CICS ONC RPC recovery procedures"
- "CICS ONC RPC operational considerations" on page 334.
- "Troubleshooting CICS ONC/RPC" on page 334
- "Using messages and codes for ONC RPC" on page 336
- "CICS ONC RPC trace information" on page 336
- "ONC RPC dump and trace formatting" on page 337
- "Debugging the ONC RPC user-replaceable programs" on page 337

CICS ONC RPC recovery procedures

Software errors within the server controller may cause it to perform an immediate disable (if this is not prevented by the nature of the error). After an immediate disable of CICS ONC RPC, CICS continues to run.

CICS ONC RPC is not included in CICS recovery. Registration details are not saved on the CICS catalog. If CICS abends and is then restarted, RPC interface registrations from the previous run are not preserved. After a CICS abend, you must enable CICS ONC RPC as described in "Enabling CICS ONC RPC" on page 284. However, 4-tuple definitions can be stored in the CICS ONC RPC data set. Each time you enable CICS ONC RPC, the definitions can be retrieved from the CICS ONC RPC data set.

If z/OS Communications Server abends, CICS ONC RPC enters immediate disable processing, but CICS continues to run.

The abending of an alias transaction might cause changes to recoverable resources to be backed out. See "Updating recoverable resources" on page 268.

CICS immediate shutdown might leave 3-tuples registered with z/OS Communications Server. These 3-tuples can be registered again when CICS ONC RPC is enabled without loss of z/OS Communications Server resources, since CICS ONC RPC always unregisters a 3-tuple before it registers it.

CICS ONC RPC operational considerations

The server controller uses EXEC CICS START to start the aliases that run the CICS programs.

CICS limits on the numbers of tasks that can be started may prevent aliases from running as soon as they are started by the server controller. This leads to delays in servicing the client requests, and this may lead to time-outs in the client.

In the XDR routines, storage allocation is done using MVS facilities, not CICS facilities. The storage is owned by the RP TCB. If an XDR routine abends, the storage is not freed by the server controller or the alias, nor is it freed by MVS, since the RP task does not end. Repeated abends in XDR routines may lead to shortage of storage that can only be corrected by stopping CICS.

MVS task control blocks (TCBs) used by ONC RPC

The TCB that interacts with z/OS Communications Server goes into a wait as a result of that interaction.

This is avoided by using an extra TCB, the RP TCB, for issuing calls to z/OS Communications Server.

The RP TCB is used for some processing for every client request, but most of the call processing done by CICS ONC RPC takes place under the QR TCB. The split between the two TCBs is transparent to you for most of your work, but you need to be aware of it for problem determination.

ONC RPC task-related user exit (TRUE)

CICS ONC RPC includes a task-related user exit; this is used to anchor shared storage and to improve CICS ONC RPC's response to CICS shutdown. CICS ONC RPC does not use a TRUE to pass commands and data to and from z/OS Communications Server.

Troubleshooting CICS ONC/RPC

This section provides some hints on troubleshooting.

About this task

It follows the general outline:
1. Define the problem.
2. Obtain information (documentation) on the problem.

Defining the problem

When you have a problem, first try to define the circumstances that gave rise to it.

About this task

If you need to report the problem to the IBM software support center, this information is useful to the support personnel.

1. What is the system configuration?
   - CICS Transaction Server release
   - z/OS Communications Server release
   - Language Environment release
2. What is the connection manager configuration?
   - Operating options
   - Registered 4-tuples
3. When did the problem first occur?
4. What were you trying to accomplish at the time the problem occurred?
5. What changes were made to the system before the occurrence of the problem?
   - To CICS ONC RPC
   - To the CICS program being called by the client
   - To the converter being used in the call
   - To the XDR routines being used in the call
   - To the client
   - To CICS Transaction Server
   - To z/OS Communications Server
6. What is the problem?
   - Incorrect output
   - Hang/Wait: If you suspect that CICS ONC RPC aliases may be in a hung state, you can use the connection manager to display a list of alias transactions and can display associated details. See "Processing the alias list" on page 300.
   - Loop: Use CEMT INQUIRE to display the details of the transaction.
   - Abend in user-replaceable program
   - Abend in a CICS program
   - Abend in the IBM-supplied part of CICS ONC RPC
   - Performance problem
   - Storage violation
   - Logic Error
7. At what point in the processing did the problem occur? (Use Figure 36 on page 267.)
8. What was the state of z/OS Communications Server? (Try the **rpcinfo** command.)

Documentation about the problem

To investigate most problems, you must look at the dumps, traces, and logs provided with MVS and CICS.

- System Dump: This contains the CICS internal trace
- CICS auxiliary trace, if enabled
- z/OS Communications Server trace
- GTF trace, if enabled
- Console log

- CSMT log
- CRPO log
- CICS job log

To identify which are likely to be useful for your problem, try to work out the area of CICS ONC RPC giving rise to the problem, and read the relevant section in the rest of this section.

Using messages and codes for ONC RPC

CICS ONC RPC messages have identifiers of the form DFHRP*nnnn*, where *nnnn* are four numeric characters.

They are sent to the CICS ONC RPC message transient data queue CRPO, or the terminal user, or both, depending on the event that is being reported. If you define CRPO as an indirect destination for CSMT, the CICS ONC RPC messages appear in CSMT. Some messages are sent to the console.

When CICS ONC RPC issues a message as a result of an error, it also makes an exception trace entry. CICS ONC RPC also generates information messages, for instance during enable processing and disable processing.

CICS ONC RPC messages are supplied in English, Kanji, and Chinese. The CICS message editing utility can be used to translate them into other languages supported by CICS.

CMAC (online help facility for messages and codes)

You can use utilities supplied as part of CICS to update your base CMAC file with the CICS ONC RPC CMAC file.

The CICS ONC RPC abend codes are listed in *CICS Messages and Codes*.

CICS ONC RPC trace information

CICS ONC RPC outputs CICS system trace, which is formatted using software supplied as part of CICS ONC RPC.

Exception trace entries produced by CICS ONC RPC are written to CICS internal trace even when the Trace operating option is set to NO. See "Setting and modifying options" on page 285 for information about the Trace option.

If selected, level 2 trace gives a full trace of the data being transmitted between the client and the CICS program. CICS trace output is described in Trace entries overview in Trace Entries.

Feature trace points

Trace points with domain identifier FT are feature trace points.

The format of these entries is slightly different from standard trace points in that the Module identifier contains the short name of the feature and a full module name. Feature trace point IDs are not globally defined. This means that a feature can reuse the trace point IDs of another feature. You should obtain information about the trace points of any other product from that product's documentation.

Numeric values of response and reason codes

The response codes from the converter and resource checker appear in the trace output as numeric values.

- URP_OK (0)
- URP_EXCEPTION (4)
- URP_INVALID (8)
- URP_DISASTER (12)

The CICS-defined reason codes from the converter and resource checker appear in the trace output as numeric codes as follows:

- URP_AUTH_BAD_CRED (1)
- URP_AUTH_TOO_WEAK (2)
- URP_CORRUPT_CLIENT_DATA (3)

ONC RPC dump and trace formatting

To switch dump formatting on and off for CICS ONC RPC, you change the CICS VERBEXIT in the JCL for dump formatting.

```
IPCS VERBEXIT DFHPD670 FT=0|1|2|3,TR=1|2
```

The parameters have these meanings:

FT=0 Suppress system dump for all features

FT=1 Produce system dump summary listing for all registered features

FT=2 Produce system dump for all registered features

FT=3 Produce system dump summary listing and a system dump for all registered features

TR=1 Produce abbreviated trace (includes trace for all registered features)

TR=2 Produce full trace (includes trace for all registered features)

Full details of these and other parameters are described in the Starting up CICS regions in the Operations and Utilities Guide.

CICS ONC RPC output in the formatted dump consists of the major control blocks of CICS ONC RPC, with interpretation of some of the fields. The CICS ONC RPC output can be found in the IPCS output by searching for ===RP. It is under the heading CICS ONC RPC Feature for z/OS.

Each trace entry for CICS ONC RPC has a comment ONC RPC to distinguish it from other trace points with the FT prefix.

Debugging the ONC RPC user-replaceable programs

The user-replaceable programs are:

- The user-written XDR routines
- The converters
- The resource checker
- The CICS programs that service the client requests

The debugging of the CICS programs is not dealt with in this manual.

XDR routines

The XDR routines, inbound and outbound, run under the RP TCB.

The CICS application programming interface is not available under the RP TCB, so you cannot use EDF, CICS abend handling, or CICS trace to diagnose problems. The **printf** function must not be used. If an XDR routine has a program check, a C run-time message is written to the CICS job log.

Converter and resource checker

The converter and resource checker run under the QR TCB, and the CICS application programming interface is available.

Using EDF

EDF is available for debugging the resource checker and the **Encode** function.

If you want to use EDF, you must:

- Ensure that the alias is terminal-attached. To do this, you must set the EDF Terminal ID field in the connection manager, as described in EDF Terminal ID. The chosen terminal must be a local terminal that is either logged on, or predefined.
- Define CEDF(YES) in the program definition of converter, or resource checker. The resource checker must run in the same CICS region as CICS ONC RPC if you want to use EDF to debug it.

Using trace entries

Diagnostic information can be output to the CICS trace by the use of the EXEC CICS ENTER TRACENUM command.

The amount of trace information and the information contained within trace entries is at your discretion. See ENTER TRACENUM, in the *CICS Application Programming Reference* for more information about this command.

Writing messages

Diagnostic messages can be output by using EXEC CICS WRITEQ TD.

Message information content, message format, frequency, and destination are at your discretion.

Abends

You are recommended to use EXEC CICS HANDLE ABEND to trap abends. You should collect the diagnostic information you need by tracing, and other forms of diagnostic output, and then return a URP_DISASTER response.

Chapter 28. CICS ONC RPC performance and tuning

Important: This information contains Diagnosis, Modification, or Tuning Information.

The performance of a single client request is affected by various aspects of the client, the network, CICS ONC RPC, the user-replaceable programs, and CICS.

The client

> The client time-out interval must take account of the possible delays in dealing with a client request in CICS ONC RPC and in CICS.
>
> If a client request cannot be processed, an error reply is sent to the client.

The network

> This manual does not deal with performance problems of TCP/IP networks.

z/Os Communications Server resources

> If, while registering 4-tuples, you cause the connection manager to register too many 3-tuples with z/OS Communications Server, you might reduce the service that CICS ONC RPC can give to incoming client requests. See "Limits on registration" on page 292.

CICS ONC RPC
> The allocation of different alias transaction names to different 4-tuples must be coordinated with the priorities given to the transactions in CICS.

The converter URM

> **Getlengths** is called only by the connection manager, and has no effect on the performance of a single client request, or on throughput.
>
> **Decode** is called by the server controller. Delays here can reduce the throughput of CICS ONC RPC as well as reducing the performance of a single client request. The following recommendations are made:
> * Do not use CICS trace here except to solve specific problems.
> * Use NOSUSPEND on EXEC CICS GETMAIN. If GETMAIN errors occur because there is not enough storage, look for solutions that do not involve using the SUSPEND option.
>
> **Encode** is called by the alias. Delays here reduce the performance of single client requests, but not the throughput of CICS ONC RPC.

The resource checker URM

> The resource checker is called by the alias, so delays here affect the performance of a single client request, but have no effect on throughput.

Part 5. Using CICS as a DCE server

CICS support for DCE remote procedure calls (RPCs) enables a non-CICS client program running in an open systems Distributed Computing Environment (DCE) to call a server program running in a CICS system and to pass and receive data using a communications area.

The CICS program is invoked as if linked-to by another CICS program.

This part contains:

Chapter 29. Introduction to the Distributed Computing Environment

CICS Transaction Server for z/OS supports DCE remote procedure calls.

CICS Transaction Server for z/OS enables a CICS program to act as a server for a DCE RPC. (Note that DCE RPC uses the DCE Security and Directory Services.) This is described in Chapter 30, "DCE remote procedure calls," on page 347.

The main advantage of a DCE remote procedure call over a CICS DPL call is that you can call CICS programs from non-CICS environments.

This section tells you what the Distributed Computing Environment (DCE) is and why you might want to use it. For more detailed information, see Chapter 30, "DCE remote procedure calls," on page 347.

What is DCE?

DCE (Distributed Computing Environment) is an architecture defined by the Open Software Foundation (OSF) to provide an Open Systems platform to address the challenges of distributed computing.

It is being ported to all major IBM and many non-IBM environments. Note that all current DCE implementations use TCP/IP rather than SNA as their communication protocol.

DCE is based on three distributed computing models:

Client/server
A way of organizing a distributed application

Remote procedure call
A way of communicating between parts of a distributed application

Shared files
A way of handling data in a distributed system, based on a personal computer file access model.

Note: CICS alone (without DCE) also supports distributed computing. See "Distributed computing" on page 6.

The rest of this section gives a high level view of the services provided by DCE.

Remote procedure call (RPC)

One way of implementing communications between a client and a server of a distributed application is to use the procedure call model.

In this model, the client makes what looks like a procedure call, and waits for a reply from the server. The procedure call is translated into network communications by the underlying RPC mechanism. The server receives a request and executes the procedure, returning the results to the client.

In DCE RPC, you define one or more DCE RPC interfaces, using the DCE interface definition language (IDL). Each interface comprises a set of associated RPC calls (called *operations*), each with their input and output parameters. You compile the IDL, which generates data structure definitions and executable stubs for both the client and the server. The matching parameter data structures ensure a common view of the parameters by both client and server. The matching client and server executable stubs handle the necessary data transformations to and from the network transmission format, and between different machine formats (EBCDIC and ASCII).

You use the DCE Directory Service to advertise that your server now supports the new interface you defined using the IDL. Your client code can likewise use the Directory Service to discover which servers provide the required interface.

You can also use the DCE Security Service to ensure that only authorized client end users can access your newly defined server function.

Directory Service

The DCE Directory Service is a central repository for information about resources in the distributed system.

Typical resources are users, machines, and RPC-based services. The information consists of the name of the resource and its associated attributes. Typical attributes could include a user's home directory, or the location of an RPC-based server.

The DCE Directory Service consists of several parts: the Cell Directory Service (CDS), the Global Directory Service (GDS), the Global Directory Agent (GDA), and a Directory Service programming interface. The CDS manages a database of information about the resources in a group of machines called a DCE *cell*. The Global Directory Service implements an international, standard directory service and provides a global namespace that connects the local DCE cells into one worldwide hierarchy. The GDA acts as a go-between for cell and global directory services. Both CDS and GDS are accessed using a single Directory Service application programming interface (API).

Security Service

There are three aspects to DCE security: authentication, secure communications, and authorization. They are implemented by several services and facilities that together comprise the DCE Security Service. These include the Registry Service, the Authentication Service, the Privilege Service, the Access Control List (ACL) Facility, and the Login Facility.

The identity of a DCE user or service is authenticated by the Authentication Service. Communications are protected by the integration of DCE RPC with the Security Service. Communication over the network can be checked for tampering or encrypted for privacy. Finally, access to resources is controlled by comparing the credentials conferred to a user by the Privilege Service with the rights to the resource, which are specified in the resource's Access Control List. The Login Facility initializes a user's security environment, and the Registry Service manages the information (such as user passwords) in the DCE Security database.

Time Service

The DCE Time Service (DTS) provides synchronized time on the computers participating in a Distributed Computing Environment.

DTS synchronizes a DCE host's time with Coordinated Universal Time (UTC), an international time standard. DTS cannot keep the time in each machine precisely the same, but can maintain it to a known accuracy. DTS also provides services which return a time range to an application (rather than a single time value), and which compare time ranges from different machines. They can be used to schedule and synchronize events across the network.

File Service

The DCE File Service (DFS) allows users to access and share files stored on a File Server anywhere on the network, without having to know the physical location of the file.

Files are part of a single, global namespace. A user anywhere on a network can access any file, just by knowing its name. The File Service achieves high performance, particularly through caching of file system data. Many users can access files that are located on a given File Server without a large amount of network traffic or delays.

Note: The File Service is based on a personal computer view of files, and is not relevant to the CICS Transaction Server for z/OS environment.

Threads

DCE Threads supports the creation, management, and synchronization of multiple threads of control within a single process.

This component is conceptually a part of the operating system layer, the layer below DCE. If the host operating system already supports threads, DCE can use that software and DCE Threads is not necessary. Because all operating systems do not provide a threads facility and DCE components require threads be present, this user-level threads package is included in DCE.

Chapter 30. DCE remote procedure calls

CICS works with the DCE Base Services and Application Support features in z/OS so that a CICS program can act as a DCE server.

Refer to the following z/OS product documentation for information about DCE Base Services:

- *DCE: Understanding the Concepts*
- *Introducing the z/OS Distributed Computing Environment*
- *z/OS Distributed Computing Environment: Application Development Guide* for guidance information about developing the client code and using the DCE base services.
- *z/OS Distributed Computing Environment: Application Development Reference* for reference information about application programming interfaces (APIs).

Refer to the following z/OS product documentation for information about DCE Application Support:

- *z/OS DCE Application Support Programming Guide*, for information about how to install CICS remote procedure call server programs.
- *z/OS Distributed Computing Environment: Application Support Configuration and Administration Guide*, for information about the administration tasks that complement the programming tasks.

Overview of DCE with CICS

The optional DCE Application Support server in z/OS enables a DCE client application anywhere in the DCE environment to access the resources of a CICS system. The client program uses the simple DCE Remote Procedure Call (RPC), or the DCE Transactional Remote Procedure Call (TRPC), mechanism to call a CICS application program.

The client program does not need to know where the required CICS application is located, because MVS provides the location information. When the client and server are on different systems, the differences are transparent to the application programmer.

The Application Support server supports client programs written in C, and CICS application programs written in COBOL. The Application Support server automatically handles the conversions of the COBOL and C data types. Components of z/OS Unix DCE Base MVS handle conversions of EBCDIC and ASCII data types, if needed.

Thus the Application Support server provides the powerful CICS application environments on the host, and the familiar (to the client workstation programmer) C language and RPC mechanism on the client.

The Application Support server:
- Coexists with all other ways of accessing CICS.
- Allows access to existing CICS applications and data.

- Allows new CICS applications to be developed as servers in the z/OS DCE Executive MVS environment.
- Allows access to all files and databases available to CICS, including DB2 databases.
- Gives the host programmer continued access to all the facilities and tools in the CICS environments. This includes requests to run other programs on the same subsystem or different subsystems using the existing CICS mechanisms.
- Allows a client program to access CICS and does not require the client machine to have CICS transaction processing function installed.

DCE terminology

The CICS server programs are called *operations*.

Each RPC requests the execution of one operation. The declarations for each operation, including the specifications for the input and output parameters, are contained in an *interface definition*. You define one or more related operations in an interface, using the Interface Definition Language (IDL).

IDL defines the server functions that a client can call. IDL is a declarative language with syntax similar to the C language. The Application Support server contains IDL extensions that enable a programmer to use COBOL syntax to define the parameters for the CICS application programs. The programmer coding the IDL declarations may be a COBOL or a C programmer.

Note: There are restrictions on the COBOL and C data structures that can be defined using the IDL..

What CICS server programs can do

The Application Support server and CICS are connected by the external CICS interface (EXCI), which uses CICS interregion connection (IRC) facilities.

The Application Support server maps the DCE RPC parameters into a CICS communications area, and then uses EXCI to invoke the required CICS program, as if it had been called by an EXEC CICS LINK command.

TRPCs from a client program within the scope of a single client transaction are handled by a single CICS task. A syncpoint issued by the client application commits or backsout all resources owned by the CICS server task as well as any owned by the client application.

Each RPC from a client program is handled as a CICS task, with an implied syncpoint at the end of the task. Note that this syncpoint only commits resources owned by the CICS server task. It does not commit any resources owned by the client program.

Your server program can access any file or database available in the CICS environment. It can use CICS distributed facilities to access data and programs that are managed by other CICS, IMS™, or other APPC-connected systems.

You can use DCE RPCor TRPC to access CICS programs for one or more of the following reasons:
- To access CICS data from a platform which does not support CICS, but which does support DCE.

- To access CICS data from workstation programs which do not run in a CICS environment. You may want to do this even if the workstation platform supports CICS.
- To use the DCE Security Service, with its high level of protection against interception of network traffic.
- To use the DCE Directory Service, to provide client independence of the location of the required server program.

For details of how to write CICS server programs, see Chapter 32, "Application programming for DCE remote procedure calls," on page 353.

What you need for DCE RPC to a CICS server

This is the list of products you need for DCE RPC to a CICS server.
- Connectivity through TCP/IP protocols to the client workstation, and to the DCE directory and security servers. This normally means a TCP/IP network, though for some partner platforms it may be possible to use an SNA network with ANYNET support at both ends to transport TCP/IP protocols using SNA transmission protocols.
- IBM z/OS Communications Server to present a TCP/IP interface to the DCE software, even if you are using an SNA network and ANYNET software.
- OS/390 Unix Systems Services Distributed Computing Environment Base Services MVS, Version 5 Release 1 or later.
- OS/390 Unix Systems Services Distributed Computing Environment Application Support MVS CICS Feature, Version 1 Release 1 or later.

Chapter 31. Defining CICS programs as DCE servers

You can define CICS programs as servers to DCE remote procedure calls (RPCs) by using the Interface Definition Language (IDL) to define one or more related operations.

CICS DCE server programs are called *operations* in DCE terminology. Each RPC requests the execution of one operation. The declarations for each operation, including the specifications for the input and output parameters, are contained in an *interface definition*. You define one or more related operations in an interface, using the Interface Definition Language (IDL).

For more detailed information, see the *z/OS DCE Application Support Programming Guide* and the *z/OS DCE Application Support Configuration and Administration Guide*.

Interface definition

When you write your CICS server program and your DCE client program you must:

1. Use the GENUUID command of the z/OS DCE Application Support server to obtain a skeleton interface definition. (An interface defines one or more related *operations*. Each operation relates to a server program.) The skeleton includes a Universal Unique Identifier (UUID) that uniquely identifies the interface.

2. Use the DCE Interface Definition Language (IDL) to identify each operation in the interface and define its input and output parameters.

3. Use the IDL compiler to generate data structure definitions for the RPC parameters and execution stubs for both client and server programs.

   The client stub packages (*marshalls*) the RPC parameters for transmission over the network to the server, and unpackages (*unmarshalls*) the parameters received from the server.

   The server stubs contain function that converts host COBOL data types to C data types and vice versa. They also package and unpackage RPC parameters, and convert data between EBCDIC and ASCII representations.

4. Link edit and load the server stubs into the server stub library.

5. Link edit the client stub with the client program.

You must also define your server programs to CICS using RDO, as described in PROGRAM resources in the Resource Definition Guide. The definitions can be statically defined and installed, or autoinstalled when the programs are first called.

Interface Installation

When you have completed your CICS server program you need to advertise its availability to potential clients. You do this by using the Application Support server administration facilities to install the interface. This exports details of the interface to the DCE distributed directory. Client programs can then use DCE facilities to locate servers which support required interfaces.

Chapter 32. Application programming for DCE remote procedure calls

Support for DCE remote procedure calls (RPCs) enables a non-CICS client program running in an Open Systems Distributed Computing Environment (DCE) to link to a server program running in a CICS system.

For an introduction to DCE RPCs, see Chapter 30, "DCE remote procedure calls," on page 347.

Writing a server program

Note: This is an overview only of how to write CICS programs to act as servers to DCE remote procedure calls. For further related information, see Chapter 31, "Defining CICS programs as DCE servers," on page 351.

CICS server programs must:

- Use a communications area to pass input and output parameters.
- Pass input and output parameters by value (not by pointer).
- Contain only data-handling logic. Existing applications that have their data-handling and terminal input/output logic in separate programs can be used without modification.
- Ideally, be written in COBOL. This is because the Application Support server compiler produces only COBOL data structure definitions for your CICS communications area, to match the RPC parameters. You can, however, write your server application in another programming language, by manually defining a communications area data structure that exactly overlays that produced in COBOL by DCE.

CICS server programs can:

- Use the same subset of EXEC CICS commands as CICS DPL server programs. The restricted commands are listed in the programming information in Exception conditions for LINK command, in *CICS Application Programming Reference*.
- Use CICS intercommunication facilities to access data and programs owned by other APPC-connected systems. For example, they can use the Front End Programming Interface (FEPI) to emulate a 3270 terminal, and thereby act as a front end for other unchanged CICS or IMS applications.
- Communicate with applications in remote CICS systems, using function shipping, DPL, or distributed transaction processing.

The Application Support server does not support CICS application programs that:

- Contain terminal input/output logic to the principal facility. (Note that you can use APPC terminal control commands to do distributed transaction processing to a remote back-end system.)
- Use basic mapping support (BMS).

These restrictions are the same as those for CICS distributed program link servers. Thus, you may be able to use server programs written for CICS-to-CICS DPL as servers to DCE clients.

As described in Interface definition, you must use the DCE MVS Application Support server compiler to generate a data structure definition for the RPC parameters passed to your server program, and an execution stub for the server. You must link edit and load the stub into the server stub library.

Writing a client program

For information about coding DCE client programs, see the *z/OS Distributed Computing Environment: Application Development Guide* and the *z/OS Distributed Computing Environment: Application Development Reference*.

Writing a server program

This information gives an overview only of how to write CICS programs to act as servers to DCE remote procedure calls. For further related information, see Chapter 31, "Defining CICS programs as DCE servers," on page 351.

CICS server programs must do the following:
- Use a communications area to pass input and output parameters.
- Pass input and output parameters by value (not by pointer).
- Contain only data-handling logic. Existing applications that have their data-handling and terminal input/output logic in separate programs can be used without modification.
- Ideally, be written in COBOL. This is because the Application Support server compiler produces only COBOL data structure definitions for your CICS communications area, to match the RPC parameters. You can, however, write your server application in another programming language, by manually defining a communications area data structure that exactly overlays that produced in COBOL by DCE.

CICS server programs can do the following:
- Use the same subset of EXEC CICS commands as CICS DPL server programs. For a list of the restricted commands, see Exception conditions for LINK command in the *CICS Application Programming Reference*.
- Use CICS intercommunication facilities to access data and programs owned by other APPC-connected systems. For example, they can use the Front End Programming Interface (FEPI) to emulate a 3270 terminal, and thereby act as a front end for other unchanged CICS or IMS applications.
- Communicate with applications in remote CICS systems, using function shipping, DPL, or distributed transaction processing.

The Application Support server does not support CICS application programs that do the following:
- Contain terminal input/output logic to the principal facility. (Note that you can use APPC terminal control commands to do distributed transaction processing to a remote back-end system.)
- Use basic mapping support (BMS).

These restrictions are the same as those for CICS distributed program link servers. Thus, you may be able to use server programs written for CICS-to-CICS DPL as servers to DCE clients.

As described in Interface definition, you must use the DCE MVS Application Support server compiler to generate a data structure definition for the RPC

parameters passed to your server program, and an execution stub for the server. You must link-edit and load the stub into the server stub library.

Part 6. Appendixes

Appendix. Routing program-link requests

"Traditional" CICS-to-CICS distributed program link (DPL) calls, instigated by
EXEC CICS LINK PROGRAM commands, can be "daisy-chained" from region to
region, by defining the program as remote in each region except the last (server)
region, where it is to execute.

Important: For detailed information about routing program-link requests, see CICS
distributed program link, in the *CICS Intercommunication Guide*. This appendix is an
overview of how program-link requests received from outside CICS can be routed
to other regions.

The same applies to program-link requests received from *outside CICS*. For
example, all of the following types of program-link request can be routed:
- Calls received from:
  - CICS Web support
  - The CICS Transaction Gateway
- Calls from external CICS interface (EXCI) client programs
- External Call Interface (ECI) calls from any of the CICS Client workstation
  products
- Distributed Computing Environment (DCE) remote procedure calls (RPCs)
- ONC/RPC calls.

Static routing

A program-link request received from outside CICS can be statically routed to a
remote CICS region by specifying the name of the remote region on the
REMOTESYSTEM option of the installed program definition.

Dynamic routing

A program-link request received from outside CICS can be dynamically routed by
defining the program to CICS as DYNAMIC(YES), and coding your dynamic
routing program to route the request.

Notices

This information was developed for products and services offered in the U.S.A. IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106, Japan

The following paragraph does not apply in the United Kingdom or any other country where such provisions are inconsistent with local law:

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore this statement may not apply to you.

This publication 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.

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 United Kingdom Laboratories, MP151, Hursley Park, Winchester, Hampshire, England, SO21 2JN.

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.

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.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

Bibliography

CICS books for CICS Transaction Server for z/OS

General
CICS Transaction Server for z/OS Program Directory, GI13-0565
CICS Transaction Server for z/OS What's New, GC34-7192
CICS Transaction Server for z/OS Upgrading from CICS TS Version 3.1, GC34-7188
CICS Transaction Server for z/OS Upgrading from CICS TS Version 3.2, GC34-7189
CICS Transaction Server for z/OS Upgrading from CICS TS Version 4.1, GC34-7190
CICS Transaction Server for z/OS Installation Guide, GC34-7171

Access to CICS
CICS Internet Guide, SC34-7173

CICS Web Services Guide, SC34-7191

Administration
CICS System Definition Guide, SC34-7185
CICS Customization Guide, SC34-7161
CICS Resource Definition Guide, SC34-7181
CICS Operations and Utilities Guide, SC34-7213
CICS RACF Security Guide, SC34-7179
CICS Supplied Transactions, SC34-7184

Programming
CICS Application Programming Guide, SC34-7158
CICS Application Programming Reference, SC34-7159
CICS System Programming Reference, SC34-7186
CICS Front End Programming Interface User's Guide, SC34-7169
CICS C++ OO Class Libraries, SC34-7162
CICS Distributed Transaction Programming Guide, SC34-7167
CICS Business Transaction Services, SC34-7160
Java Applications in CICS, SC34-7174

Diagnosis
CICS Problem Determination Guide, GC34-7178
CICS Performance Guide, SC34-7177
CICS Messages and Codes Vol 1, GC34-7175
CICS Messages and Codes Vol 2, GC34-7176
CICS Diagnosis Reference, GC34-7166
CICS Recovery and Restart Guide, SC34-7180
CICS Data Areas, GC34-7163
CICS Trace Entries, SC34-7187
CICS Debugging Tools Interfaces Reference, GC34-7165

Communication
CICS Intercommunication Guide, SC34-7172
CICS External Interfaces Guide, SC34-7168

Databases
CICS DB2 Guide, SC34-7164

CICS IMS Database Control Guide, SC34-7170

CICS Shared Data Tables Guide, SC34-7182

CICSPlex SM books for CICS Transaction Server for z/OS

General
CICSPlex SM Concepts and Planning, SC34-7196
CICSPlex SM Web User Interface Guide, SC34-7214

Administration and Management
CICSPlex SM Administration, SC34-7193
CICSPlex SM Operations Views Reference, SC34-7202
CICSPlex SM Monitor Views Reference, SC34-7200
CICSPlex SM Managing Workloads, SC34-7199
CICSPlex SM Managing Resource Usage, SC34-7198
CICSPlex SM Managing Business Applications, SC34-7197

Programming
CICSPlex SM Application Programming Guide, SC34-7194
CICSPlex SM Application Programming Reference, SC34-7195

Diagnosis
CICSPlex SM Resource Tables Reference Vol 1, SC34-7204
CICSPlex SM Resource Tables Reference Vol 2, SC34-7205
CICSPlex SM Messages and Codes, GC34-7201
CICSPlex SM Problem Determination, GC34-7203

Other CICS publications

The following publications contain further information about CICS, but are not provided as part of CICS Transaction Server for z/OS, Version 4 Release 2.

Designing and Programming CICS Applications, SR23-9692

CICS Application Migration Aid Guide, SC33-0768

CICS Family: API Structure, SC33-1007

CICS Family: Client/Server Programming, SC33-1435

CICS Family: Interproduct Communication, SC34-6853

CICS Family: Communicating from CICS on System/390, SC34-6854

CICS Transaction Gateway for z/OS Administration, SC34-5528

CICS Family: General Information, GC33-0155

CICS 4.1 Sample Applications Guide, SC33-1173

CICS/ESA 3.3 XRF Guide , SC33-0661

Accessibility

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

You can perform most tasks required to set up, run, and maintain your CICS system in one of these ways:

- using a 3270 emulator logged on to CICS
- using a 3270 emulator logged on to TSO
- using a 3270 emulator as an MVS system console

IBM Personal Communications provides 3270 emulation with accessibility features for people with disabilities. You can use this product to provide the accessibility features you need in your CICS system.

Index

Numerics

W

X

Z

Readers' Comments — We'd Like to Hear from You

CICS Transaction Server for z/OS
Version 4 Release 2
External Interfaces Guide

Publication No. SC34-7168-01

We appreciate your comments about this publication. Please comment on specific errors or omissions, accuracy, organization, subject matter, or completeness of this book. The comments you send should pertain to only the information in this manual or product and the way in which the information is presented.

For technical questions and information about products and prices, please contact your IBM branch office, your IBM business partner, or your authorized remarketer.

When you send comments to IBM, you grant IBM a nonexclusive right to use or distribute your comments in any way it believes appropriate without incurring any obligation to you. IBM or any other organizations will only use the personal information that you supply to contact you about the issues that you state on this form.

Comments:

Thank you for your support.

Submit your comments using one of these channels:
- Send your comments to the address on the reverse side of this form.
- Send a fax to the following number: +44 1962 816151
- Send your comments via email to: idrcf@uk.ibm.com

If you would like a response from IBM, please fill in the following information:

_____

Name

_____

Company or Organization

_____

Phone No.

_____

Address

_____

_____

Email address

Readers' Comments — We'd Like to Hear from You

SC34-7168-01

IBM®

Fold and Tape　　　　　**Please do not staple**　　　　　Fold and Tape

PLACE
POSTAGE
STAMP
HERE

IBM United Kingdom Limited
User Technologies Department (MP095)
Hursley Park
Winchester
Hampshire
United Kingdom
　SO21 2JN

Fold and Tape　　　　　**Please do not staple**　　　　　Fold and Tape

Readers' Comments — We'd Like to Hear from You

SC34-7168-01

IBM ®

SC34-7168-01