CICS Transaction Server for z/OS



External Interfaces Guide

Version 5 Release 4

CICS Transaction Server for z/OS



External Interfaces Guide

Version 5 Release 4

Note
Before using this information and the product it supports, read the information in "Notices" on page 183.

This edition applies to the IBM CICS Transaction Server for z/OS Version 5 Release 4 (product number 5655-Y04) and to all subsequent releases and modifications until otherwise indicated in new editions.

© Copyright IBM Corporation 1994, 2017.
US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

About this PDF. v

Chapter 1. CICS external interfaces . . . 1
Interfaces to CICS transactions and programs . . . 1
The client/server model 2
Distributed computing 2
TCP/IP protocols 4
ONC and DCE concepts 7
EXCI concepts 8
3270 bridge concepts. 9

Chapter 2. Bridging to 3270 transactions 11
Introduction to the 3270 bridge 11
The Link3270 bridge mechanism 12
The bridge facility 13
The application data structure (ADS) 13
Link3270 programming considerations 14
Transaction Routing considerations 18
Using the Link3270 bridge 20
Establish Link3270 suitability 21
Writing the Link3270 client 22
Using Link3270 messages. 26
Using Link3270 single transaction mode 29
Using Link3270 session mode 30
Calling the Link3270 bridge 35
Using data conversion with Link3270. 37
Managing the Link3270 bridge environment . . . 38
Defining Link3270 system initialization parameters 39
Defining the bridge facility 39
Administering the Link3270 bridge 46
INQUIRE/SET AUTOINSTALL with the Link3270 bridge 46
INQUIRE/SET BRFACILITY with the Link3270 bridge 46
INQUIRE TASK with the Link3270 bridge . . . 47
INQUIRE/SET TRACETYPE with the Link3270 bridge 47
INQUIRE TRANSACTION with the Link3270 bridge 47
XPI commands for the Link3270 bridge 47
Using Link3270 bridge load routing 48
Link3270 message formats 49
Link3270 message header (BRIH) 50
Inbound Link3270 vectors 58
Outbound Link3270 vectors 64
Link3270 ADS descriptor 81
Link3270 diagnostics 83
BRIH-RETURNCODE values 84
Link3270 sample programs 89
About the NACT transaction 89
Running the sample client programs 90
Setting up the NACT transaction 93

Chapter 3. CICS ONC RPC support . . 95
Introduction to ONC RPC 95
ONC RPC concepts. 96
ONC RPC facilities 97
ONC RPC naming and routing 99
CICS ONC RPC concepts 101
Setting up CICS ONC RPC 109
CICS ONC RPC setup tasks 110
Defining CICS ONC RPC resources to CICS . . 111
Configuring CICS ONC RPC using the connection manager 114
Starting the connection manager 114
Updating CICS ONC RPC status 117
Enabling CICS ONC RPC 118
Defining, saving, modifying, and deleting 4-tuples 120
Registering the 4-tuples 125
Unregistering 4-tuples 126
Disabling CICS ONC RPC 128
Updating the CICS ONC RPC data set 130
Processing the alias list 134
Developing CICS ONC RPC applications 136
Developing an ONC RPC application for CICS ONC RPC 136
Write the CICS ONC RPC converter. 140
Reference information for the converter functions 149
Security for ONC RPC 160
Security in ONC RPC 160
Security in CICS and its effect on CICS ONC RPC operations. 160
Writing the resource checker 162
Troubleshooting ONC 164
CICS ONC RPC recovery procedures 165
CICS ONC RPC operational considerations . . 165
Troubleshooting CICS ONC/RPC. 166
Using messages and codes for ONC RPC . . . 167
CICS ONC RPC trace information 168
ONC RPC dump and trace formatting 168
Debugging the ONC RPC user-replaceable programs. 169
Improving ONC RPC performance 170

Chapter 4. Introduction to the Distributed Computing Environment . 173
What is DCE? 173
Remote procedure call (RPC) 173
Directory Service 174
Security Service 174
Time Service. 174
File Service 175
Threads 175
DCE remote procedure calls 175
Overview of DCE with CICS 176
What CICS server programs can do 177

© Copyright IBM Corp. 1994, 2017

What you need for DCE RPC to a CICS server 177
Application programming for DCE remote procedure calls 178
Defining CICS programs as DCE servers . . . 179

Appendix. Routing program-link requests 181

Notices 183

Index 189

About this PDF

This PDF describes how you can use the ONC RPC, DCE, and 3270 bridge interfaces to make the services of CICS Transaction Server for z/OS available to external programs. Note that the information about using EXCI that used to be in this PDF is now in a separate PDF called *Using EXCI with CICS*.

For details of the terms and notation used in this book, see Conventions and terminology used in the CICS documentation in IBM Knowledge Center.

Date of this PDF

This PDF was created on June 6th 2017.

© Copyright IBM Corp. 1994, 2017

Chapter 1. CICS external interfaces

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

Interfaces to CICS transactions and programs

This information describes sources of external requests, and the routes that they can use into CICS.

WebSphere® MQ users
IBM MQ users can use the CICS 3270 bridge to access CICS transactions. See Introduction to the 3270 bridge and .

MVS™ applications
Applications running in MVS address spaces can use the External CICS Interface (EXCI) to access CICS programs. For more information see Introduction to the external CICS interface.

ONC RPC clients
ONC RPC clients can use CICS ONC RPC support to access CICS programs.

DCE RPC clients
DCE RPC clients use the Application Support (AS) server to access CICS programs..

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. For more information, see z/OS Communications Server: IP Configuration Guide.

Web browsers
Web browsers can use a number of access methods:

CICS Web support
The CICS support for web browsers. For more information see the Configuring CICS web support components

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
CICS Transaction Gateway is a Gateway component that can accept requests from client applications and route them into CICS. It uses the EXCI, IPIC, or APPC interconnectivity protocols to access CICS.

Java-enabled Web browsers
Java-enabled web browsers can use applets that communicate with CICS. Writers of applets can use Java™ classes provided with CICS 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 can run on a wide variety of client operating systems

© Copyright IBM Corp. 1994, 2017

and interface with CICS applications using the ECI, EPI, or ESI interfaces provided by CICS Transaction Gateway. For more information see 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. For more information see Introduction to FEPI.

Telnet clients
Telnet clients can use TN3270 to start transactions.

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

For further information on connectivity options see IBM Redbooks: CICS and SOA: Architecture and Integration Choices.

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 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:

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.

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 previous 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 1 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, Figure 1 shows the SNA layers that correspond very roughly to the OSI layers.

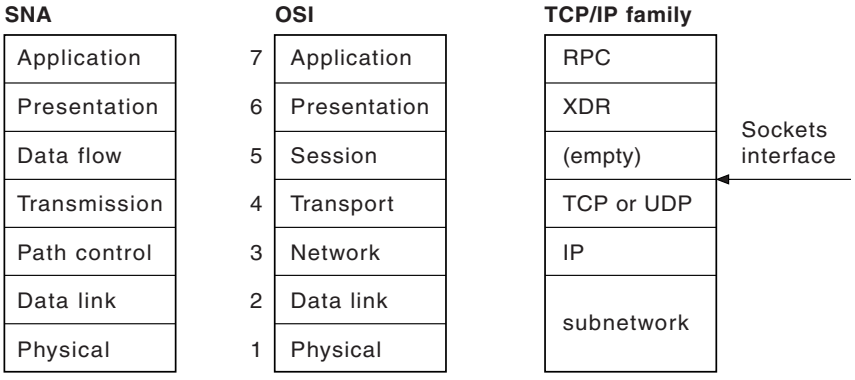


Figure 1. TCP/IP protocols compared to the OSI and SNA models

The protocols used by TCP/IP are shown in Figure 1.

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 2) 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.

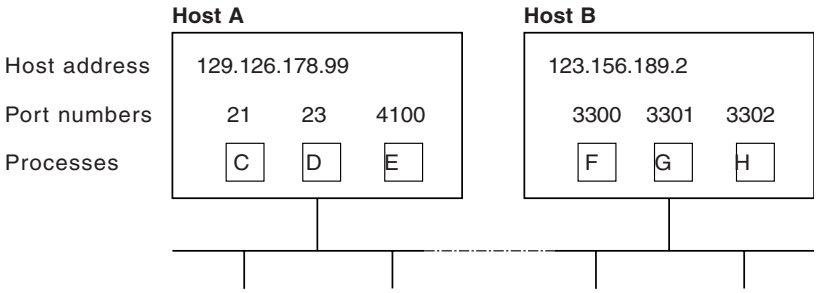


Figure 2. How applications are addressed

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 :   '00000000000000000000ffff01020304'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 2 on page 5, 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 2 on page 5, 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).

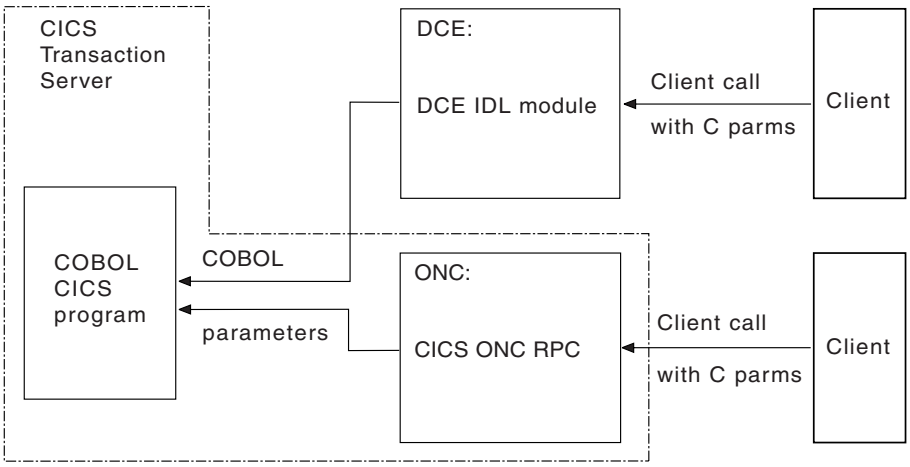


Figure 3. Remote procedures provided for DCE RPC and ONC RPC

Figure 3 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 IBM 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.

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 IBM 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. START BREXIT is not documented in this release of CICS. To use it or modify existing implementations, refer to http://publibfp.dhe.ibm.com/cgi-bin/bookmgr/BOOKS/dfhjatm6/CCONTENTS.

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
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 designed)
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

Chapter 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.

CICS provides sample client programs that use the ECI, EXCI and LINK interfaces to call the Link3270 bridge to run the sample transaction NACT. These sample programs provide coded examples that help you write your own client programs. For more information, see Link3270 sample programs.

- Introduction to the 3270 bridge
- Using the Link3270 bridge
- "Managing the Link3270 bridge environment" on page 38
- Link3270 message formats
- Link3270 diagnostics

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 IBM 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.

© Copyright IBM Corp. 1994, 2017

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 4 :

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 code page 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 .

The Link3270 mechanism supports non-conversational, pseudoconversational and

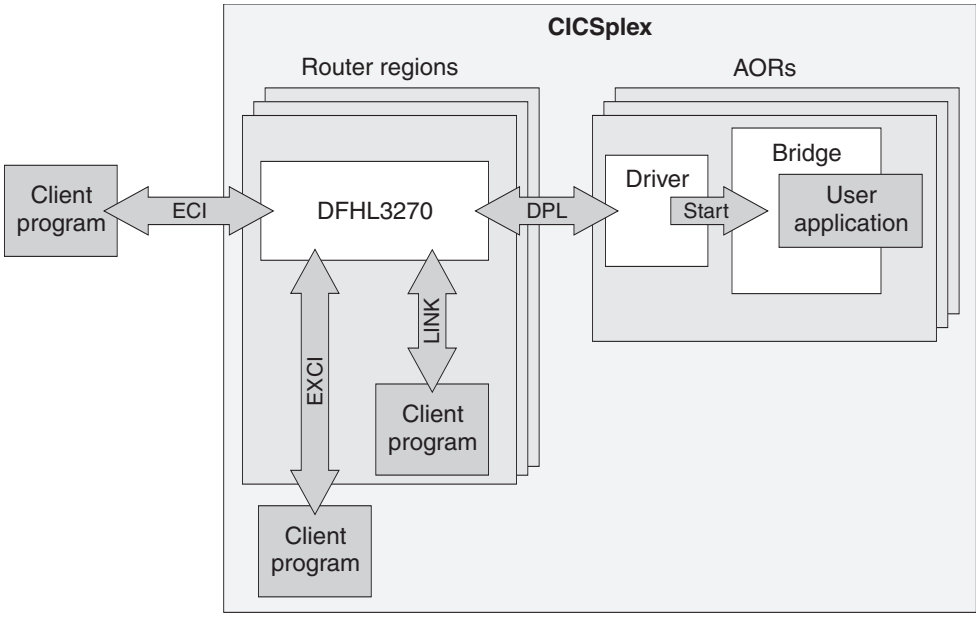


Figure 4. Link3270 request flow

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 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 30.

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 information about system initialization parameters, see Specifying CICS system initialization parameters.

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.

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 14 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 cannot reassemble the map set because you do not have the source, you can use the DFHBMSUP utility to re-create source statements from your map set load module.

For more information about DFHBMSUP, see BMS macro generation utility (DFHBMSUP).

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 Administering CICS operations), 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.

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 13.

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 "P

seudoconversational transactions" on page 33 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. 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 22.

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 "}". 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. 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.

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 Link3270 sample programs for more information about the Link3270 samples and NACT.

This section describes:

- "Establish Link3270 suitability"
- "Writing the Link3270 client" on page 22
- "Using Link3270 messages" on page 26
- "Using Link3270 single transaction mode" on page 29
- "Using Link3270 session mode" on page 30
- "Calling the Link3270 bridge" on page 35
- "Using data conversion with Link3270" on page 37

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"), 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") 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 , subject to the restrictions described in "Link3270 programming considerations" on page 14. 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 information about running the load module scanner utility, see Load module scanner (DFHEISUP).

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 Web, at the following URL:
`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 5 , 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.

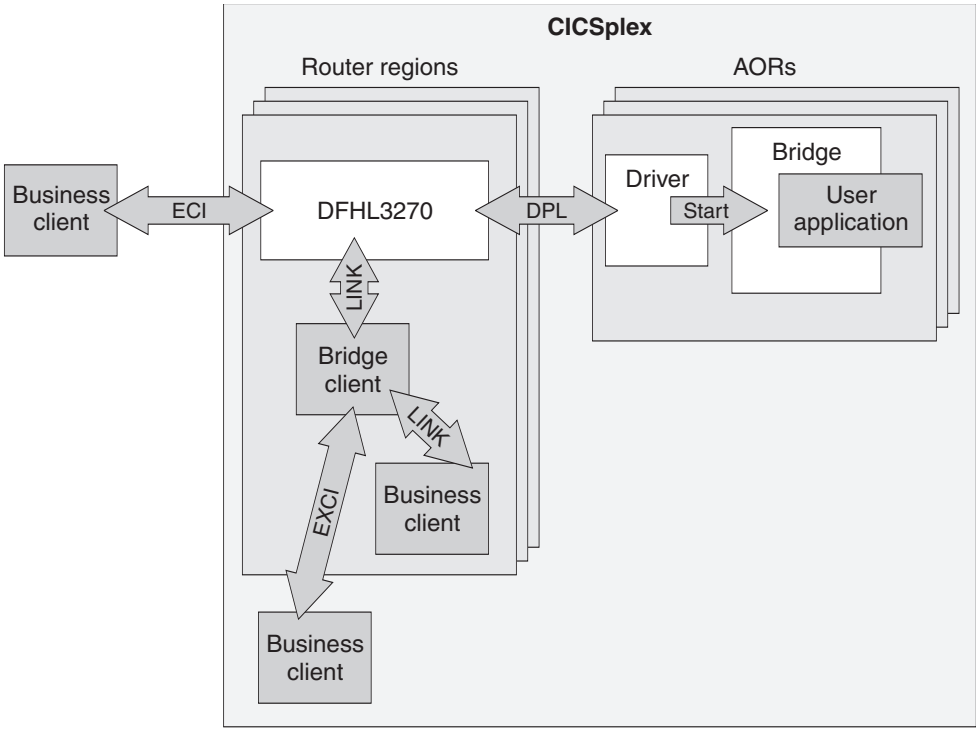


Figure 5. Link3270 host CICS client scenario

2. **CICS Workstation Client**

   In this scenario, shown in Figure 6 on page 24 , 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.

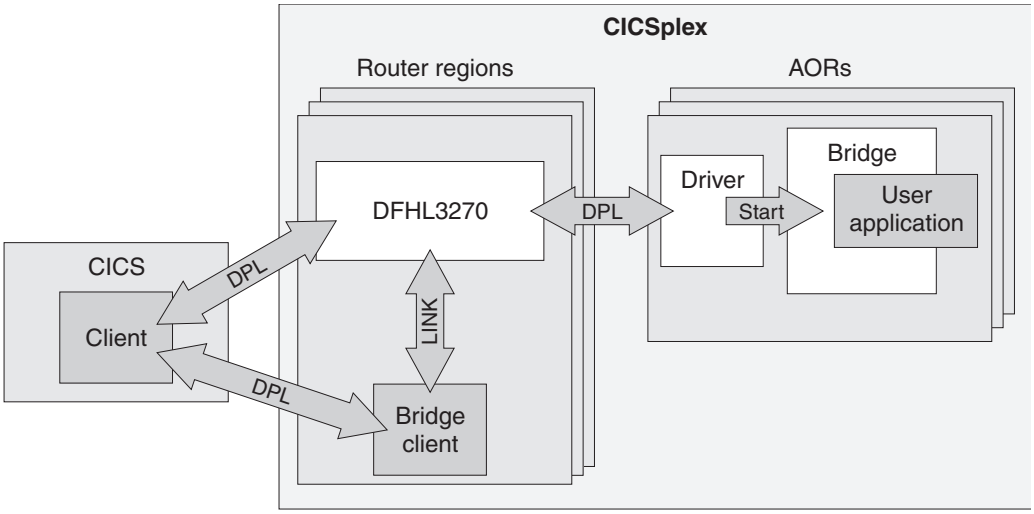


Figure 6. Link3270 CICS workstation client scenario

3. **Non - CICS Workstation Client**

   In this scenario, shown in Figure 7 , 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.

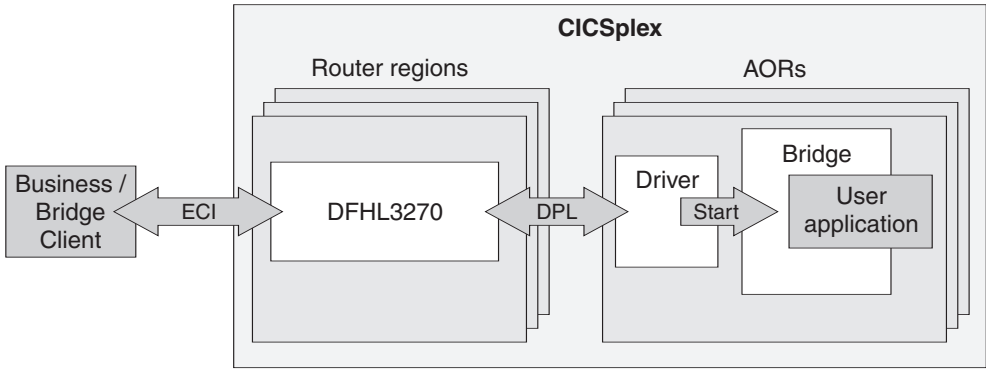


Figure 7. Link3270 non-CICS workstation client scenario

4. **3–tier Workstation client**

   In this scenario, shown in Figure 8 on page 25 , 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.

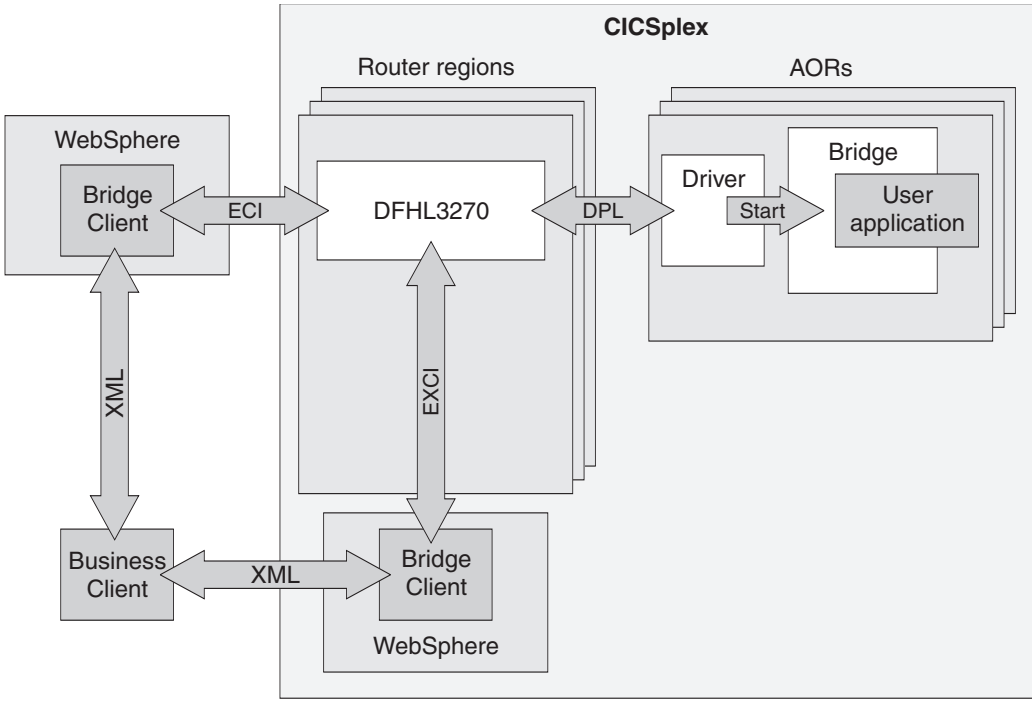


Figure 8. Link3270 3–tier client scenario

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 21) 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 DFHL3270 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.

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.

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.

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 32) 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 30 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 32), 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 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 Link3270 programs using the extended copybooks (listed in Table 6 on page 29), 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 following tables. Table 5 on page 29 shows the basic copybooks. Table 6 on page 29 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. For more information, see Link3270 sample programs.

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) 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" on page 30.

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) 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.

P seudoconversational 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.

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 will contain 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.

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 Link3270 message formats.

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 Link3270 diagnostics for a full description of the return codes from the Link3270 call.

See LINK for a full description of the LINK command.

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 Using the Link3270 bridge for a description of single transaction and session modes. See The EXCI programming interfaces 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 abendcode. See Link3270 diagnostics 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 code page of your client program is different from the code page 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 conversion process 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 Defining the conversion table.

Converting BRIH and BRIV header data

If you are using code pages other than the defaults, that is, other than 437 for the client and 037 for the server, then you must ensure that the correct code page 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 code pages as those specified on the `TYPE=INITIAL` statement, then the `TYPE=ENTRY` statements do not need to specify the code pages.

If an application program, or programs, need to use different code pages, 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 Assembling and link-editing the conversion programs . This will convert the BRIH and BRIV vector headers using the code pages described in your conversion template.

DFHCNV example for Link3270:

This shows an example of a DFHCNV for a CICS region using code page 939 for the server programs and code page 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 code pages, but PROG2 uses client code page 932 instead of 939. In order to achieve this, the TYPE=ENTRY statement for PROG2 contains overrides for the client and server code pages.

```
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 9. 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 Link3270 call if you are not using the default. The client code page must be specified in value 2 of CGCSGID in the AOR as well as in the DFHCNV conversion table in the router region.

The code page conversion for user data is based on the code page 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.

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 40.

Optionally, you can also:

- Allocate specific names to bridge facilities using the autoinstall user replaceable program. See “Defining the bridge facility.”
- 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 balancing.

This section 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”
- Administering the Link3270 bridge
- Using Link3270 bridge load balancing

Defining Link3270 system initialization parameters

You can optionally define system initialization parameters to help manage the Link3270 bridge.

About this task

Typically, CICS defines all bridge facilities automatically. However, you can write a user replaceable module to control the autoinstallation of bridge facilities if required.

Procedure

1. In the bridge router region, define the AIBRIDGE system initialization parameter to specify whether the autoinstall user replaceable module (URM) is called when bridge facilities are created and deleted.
2. Define the SPCTR and STNTR system initialization parameters if you want standard and special tracing for the bridge (BR) and partner (PT) domains.

Results

You have configured CICS to use a URM to install bridge facilities and added tracing.

Defining the bridge facility

The bridge facility is an emulated 3270 terminal. It is a virtual terminal, created by DFHL3270 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 facility

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.

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 30).

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 “Using the Link3270 bridge” on page 20 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 need to use IDCAMS to create file DFHBRNSF.

Figure 10 shows a some sample IDCAMS statements that you can modify to create the DFHBRNSF file. Change the cluster name and volume values 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 10. Sample IDCAMS job to create the DFHBRNSF file

See Coupling facility data tables 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 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 How bridge facility virtual terminals are autoinstalled for information about the autoinstall URM.

Initializing the TCTUA

The bridge facility can have a Terminal Control Table User Area (TCTUA), 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.

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

Table 9. INQUIRE TERMINAL values (continued)

Keyword	Returned value
CORRELID	blanks
EXITTRACING	NOTAPPLIC
LINKSYSTEM	blanks
MODENAME	blanks
REMOTENAME	blanks
REMOTESYSTEM	blanks
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.

KEYWORD	EFFECT
ZCPTRACING	Ignored.

Administering the Link3270 bridge

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 the System commands for details of INQUIRE and SET commands, and the 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**.

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**.

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.

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.

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.

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 INQUIRE_CONTEXT function returns the following information:

- The name of the bridge exit program used by a task running the START BREXIT bridge mechanism.
- The bridge facility token 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 Transaction management XPI functions 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 11 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.

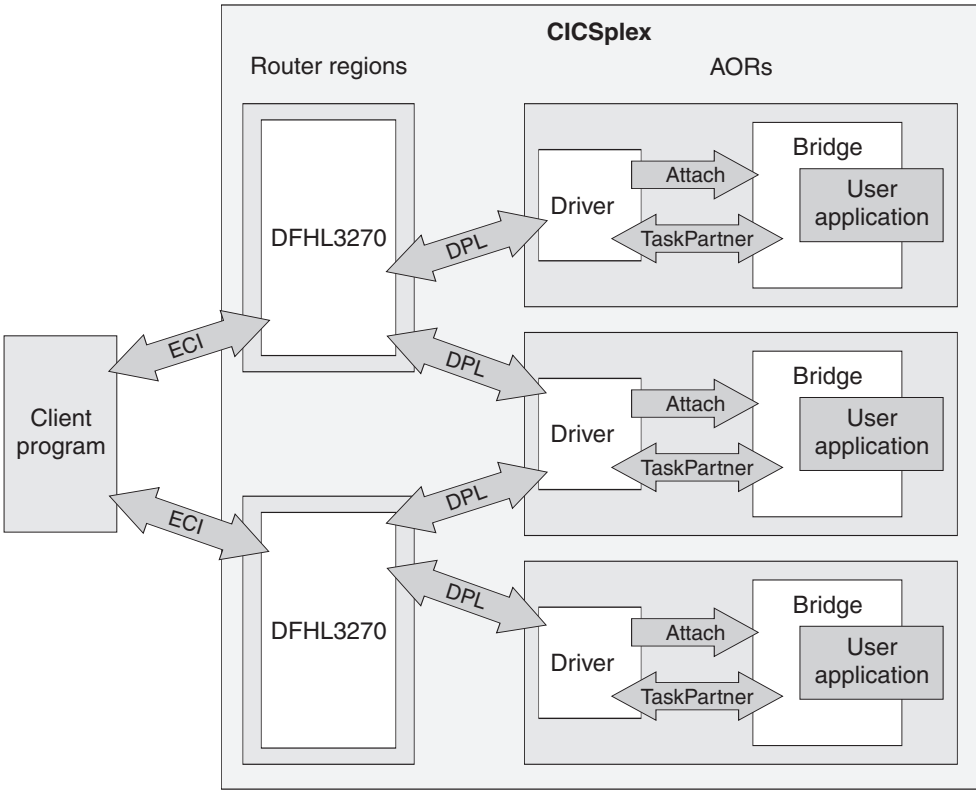


Figure 11. Link3270 load balancing

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 Routing bridge requests dynamically for information about using a dynamic routing program.

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, the current version is set to indicate basic support. The extended copybooks 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

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.

"Inbound BRIH message header" on page 51 describes the values of fields used for input. "Outbound BRIH message header" on page 55 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	

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

Offset Hex	Type	Len	Name	Default
(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
(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 and Using Link3270 session mode.

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 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 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 attributes.

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 attributes.

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
(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

Table 12. The output BRIH message header (continued)

Offset Hex	Type	Len	Name
(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 84 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 84 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 84 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 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 84 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 84 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 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 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" on page 59
- "Link3270 RECEIVE vector" on page 60
- "Link3270 RECEIVE MAP vector" on page 61
- "Link3270 RETRIEVE vector" on page 63

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) 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 RECEIVE (3270 logical) 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 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

Offset Hex	Type	Len	Name
(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
(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 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.

"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 65
- "Link3270 SEND vector" on page 66
- "Link3270 SEND CONTROL vector" on page 67
- "Link3270 SEND MAP vector" on page 70
- "Link3270 SEND TEXT vector" on page 73
- "Link3270 SYNCPOINT vector" on page 77
- "Link3270 CONVERSE request vector" on page 78
- "Link3270 RECEIVE request vector" on page 80
- "Link3270 RECEIVE MAP request vector" on page 80
- "Link3270 SEND PAGE vector" on page 76
- "Link3270 PURGE MESSAGE vector" on page 77

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.

This vector is the data supplied by an EXEC CICS ISSUE ERASEAUP command. See ISSUE ERASEAUP 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) and CONVERSE: VTAM options.

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 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
(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 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 MAPPINGDEV 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]**

Offset Hex	Type	Len	Name
	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. 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 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. 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 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

Offset Hex	Type	Len	Name
(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]**
(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. 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 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 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 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 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 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 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. 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 MAPPEDin the IBM Knowledge Center 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 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 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.

This vector is the data supplied by an EXEC CICS PURGE MESSAGE command. See PURGE MESSAGE 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 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 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) for details of the command options.

Offset Hex	Type	Len	Name
(0)	STRUCTURE	48	BRIV-CONVERSE-REQUEST

Offset Hex	Type	Len	Name
(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) 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 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

Offset Hex	Type	Len	Name
(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.

Link3270 diagnostics

Link3270 messages are subject to a number of validation stages.

Validation error types

Invalid Message
If a COMMAREA is passed to DFHL3270 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 DFHL3270.

Return codes and abend codes for Link3270 message validation errors

Return codes and abend codes are 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" 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.

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

Link3270 sample programs

CICS provides sample client programs that use the ECI, EXCI and LINK interfaces to call the Link3270 bridge to run the sample transaction NACT. These sample programs provide coded examples that help you write your own client programs.

The sample transaction NACT 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:

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 Select Link3270 client scenarios for an illustration of the host client.

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 Select Link3270 client scenarios for an illustration of the workstation client.

About 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

After you set up the NACT transaction, the Link3270 environment, and the clients, you can run the sample client programs.

Before you begin

Ensure that you complete the setup procedures:

- "Setting up the Link3270 environment" on page 91
- "Setting up the NACT transaction" on page 93
- "Setting up CICS-based clients" on page 91
- "Setting up the z/OS-based client" on page 92
- "Setting up the workstation client" on page 92

Procedure

- **Running the samples from a CICS-based client**
  1. At a CICS terminal, enter the transaction name BRCO (for the COBOL sample) or BRCH (for the C sample).
- **Running the samples from a z/OS-based client**
  1. To run the samples, you can use the following sample JCL supplied in file DFH$BRXJ in SDFHINST. You must edit this JCL as follows:

     hlq
     Specify your own prefix, which is assigned during CICS installation.

     application library
     Specify the name of the library that contains the load modules.

     Note: 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 12. EXCI sample JCL

Result: The output from the NACT sample is returned in the output from this job.

- **Running the samples from a workstation client**

  1.

Setting up the Link3270 environment

Use the following procedure to set up the Link3270 environment.

Procedure

Set up the Link3270 environment. Ensure that you have defined:

- Link3270 system initialization parameters
- The DFHBRNSF file

Setting up CICS-based clients

Add the load library containing the load modules to the RPL concatenation of your CICS startup job.

Procedure

1. Install and set up the NACT transaction.
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 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.

What to do next

Run the sample programs from a CICS-based client.

Setting up the z/OS-based client

Use the following procedure to set up the z/OS-based client.

Procedure

1. Install and set up the NACT transaction.
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 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 Defining connections to CICS and Introduction to the external CICS interface for an introduction and guidance on how to use of the EXCI interface.

What to do next

Run the sample programs from a z/OS-based client.

Setting up the workstation client

Use the following procedure to set up the workstation client.

Procedure

1. Install and setup the NACT transaction.

2. Install the CICS Transaction Gateway on your workstation, as described in the relevant CICS Transaction Gateway product information for your workstation platform. See CICS Transaction Gateway for Multiplatforms.
3. Download the following programs and header files:

Table 14. Required files

File	Source library
dfh$brec.c	CICSTS54.CICS.SDFHSAMP
dfh$brxc.h	CICSTS54.CICS.SDFHSAMP
dfh$brmh.h	CICSTS54.CICS.SDFHSAMP
dfhbrich.h	CICSTS54.CICS.SDFHC370
dfhbrihh.h	CICSTS54.CICS.SDFHC370

4. Compile DFH$BREC with these header files and DFHAID.h in your path.
5. Set up a TCP/IP or IPIC server connection to CICS as described in CICS Transaction Gateway for Multiplatforms.

What to do next

Run the sample programs from a workstation client.

Setting up 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, which are 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
DFH0CNAF	Copybook	SDFHSAMP
DFH0CNAG	Copybook	SDFHSAMP
DFH0CNAL	Copybook	SDFHSAMP
DFH0CNAM	Copybook	SDFHSAMP
DFH0CNAR	Copybook	SDFHSAMP
DFH0CNAU	Copybook	SDFHSAMP

Table 15. NACT sample components (continued)

File	Type	Library
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.

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 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 in Table 15 on page 93 are accessible. See Device dependent support 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.

Results

You can now test that installation is complete by entering the transaction NACT on a CICS terminal.

Chapter 3. 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:

- Introduction to ONC RPC
- CICS ONC RPC concepts
- Setting up CICS ONC RPC
- "Configuring CICS ONC RPC using the connection manager" on page 114
- Programming with CICS ONC RPC
- CICS ONC RPC security in Reference
- CICS ONC RPC problem determination
- Improving ONC RPC performance

Introduction to ONC RPC

CICSONC RPC allows client applications to access CICS programs by calling them as remote procedures using the ONC RPC format.

CICSONC 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. For information about the function of ONC RPC Version 3.9 supported by TCP/IP for MVS, see z/OS Communications Server: IP Programmer's Guide and Reference.

Figure 13shows how CICS ONC RPC allows a variety of client applications to communicate with CICS programs using ONC RPC.

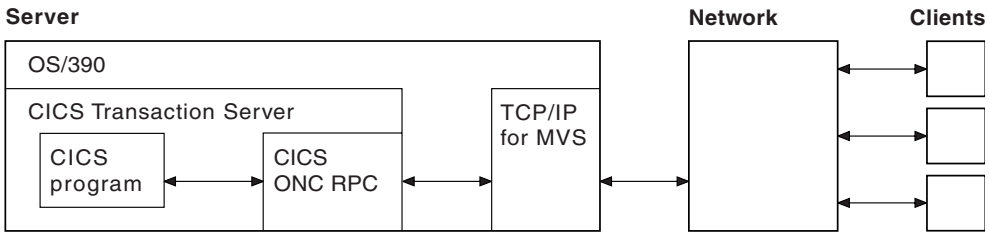


Figure 13. How CICS ONC RPC might be used

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

© Copyright IBM Corp. 1994, 2017

- BMS commands
- Sign-on and sign-off commands.

The rest of this chapter describes:
- "ONC RPC concepts"
- "ONC RPC facilities" on page 97
- "ONC RPC naming and routing" on page 99

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 14, 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.

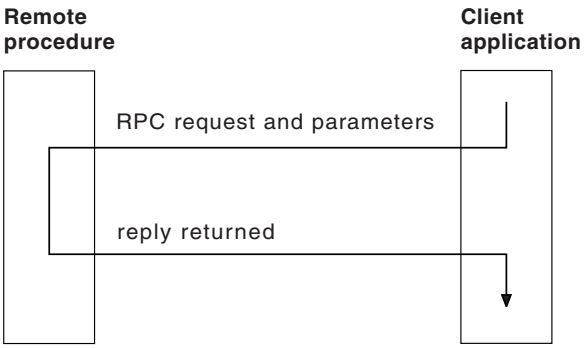


Figure 14. Basic RPC operation

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 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 15, **inproc** and **outproc** are the XDR routines.

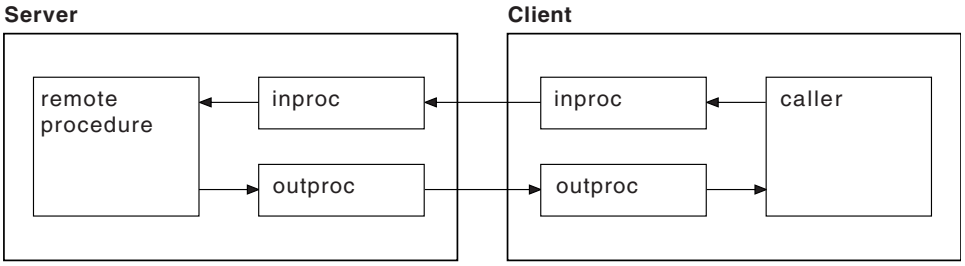


Figure 15. XDR routines used in a remote procedure call

Notice that in Figure 15, 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," 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 16 illustrates the role of RPCGEN in application development.

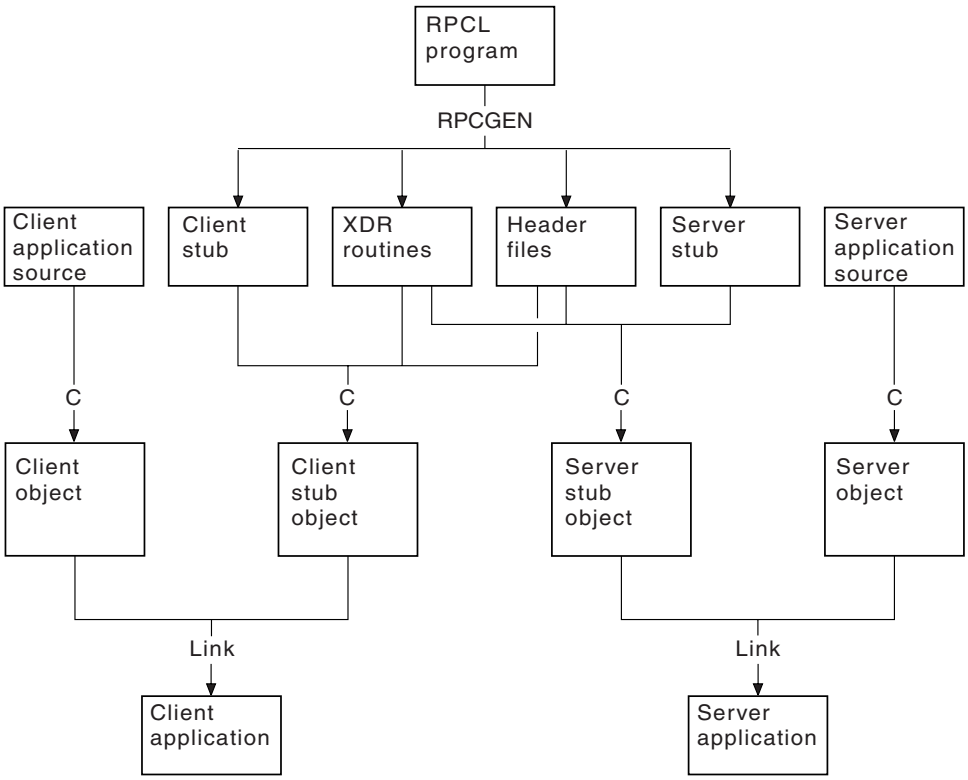


Figure 16. Using the RPCGEN compiler

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 timeouts 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 17 shows how the IP address, port number, and procedure number identify the server procedure.

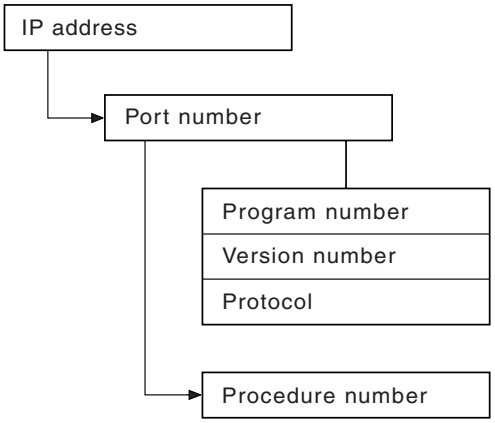


Figure 17. TCP/IP and RPC routing

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.

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 102
- "CICS ONC RPC user-replaceable programs" on page 103
- "CICS ONC RPC control flow" on page 104
- "CICS ONC RPC data flow" on page 106

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 18 shows a state of CICS ONC RPC in which five 4-tuples are associated with three CICS programs.

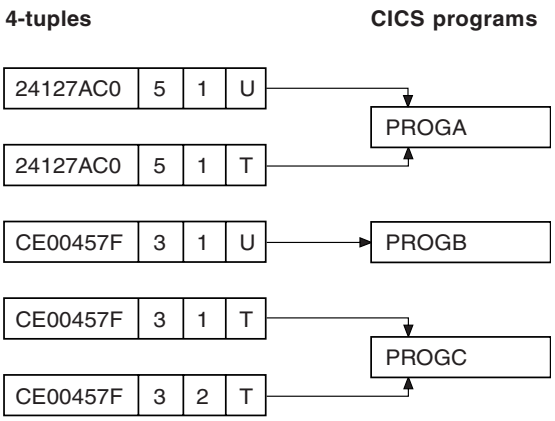


Figure 18. Remote procedures and CICS programs

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.

How you set up and control the relationship between 4-tuples and CICS programs is described in Lengths of the CICS program input and output data.

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 Lengths of the CICS program input and output data.

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 19. 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.

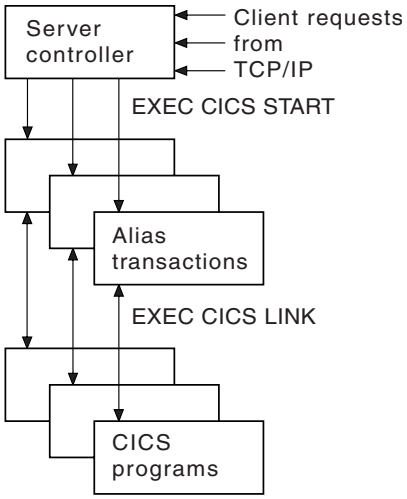


Figure 19. The server controller and alias transactions

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 97.

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 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 106 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 106 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.

CICS ONC RPC control flow

This shows the components involved in processing a typical client ONC RPC request.

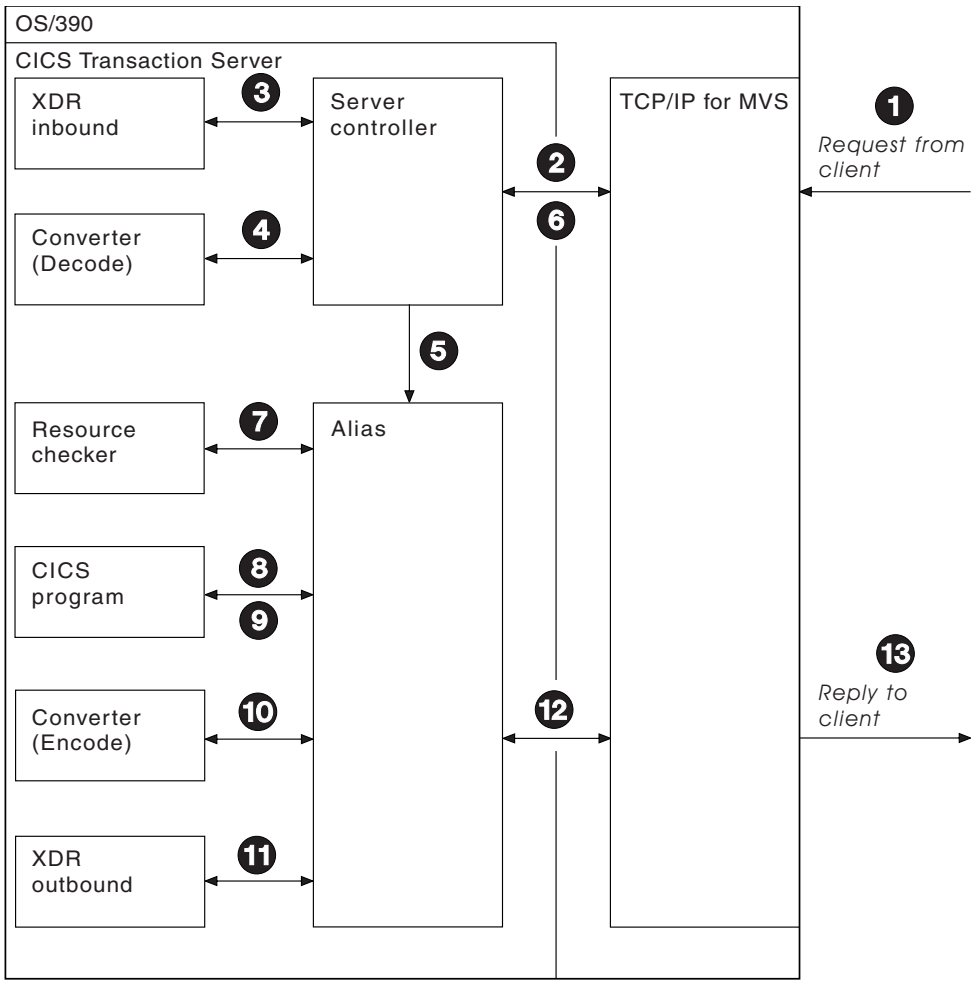


Figure 20. Call processing

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.

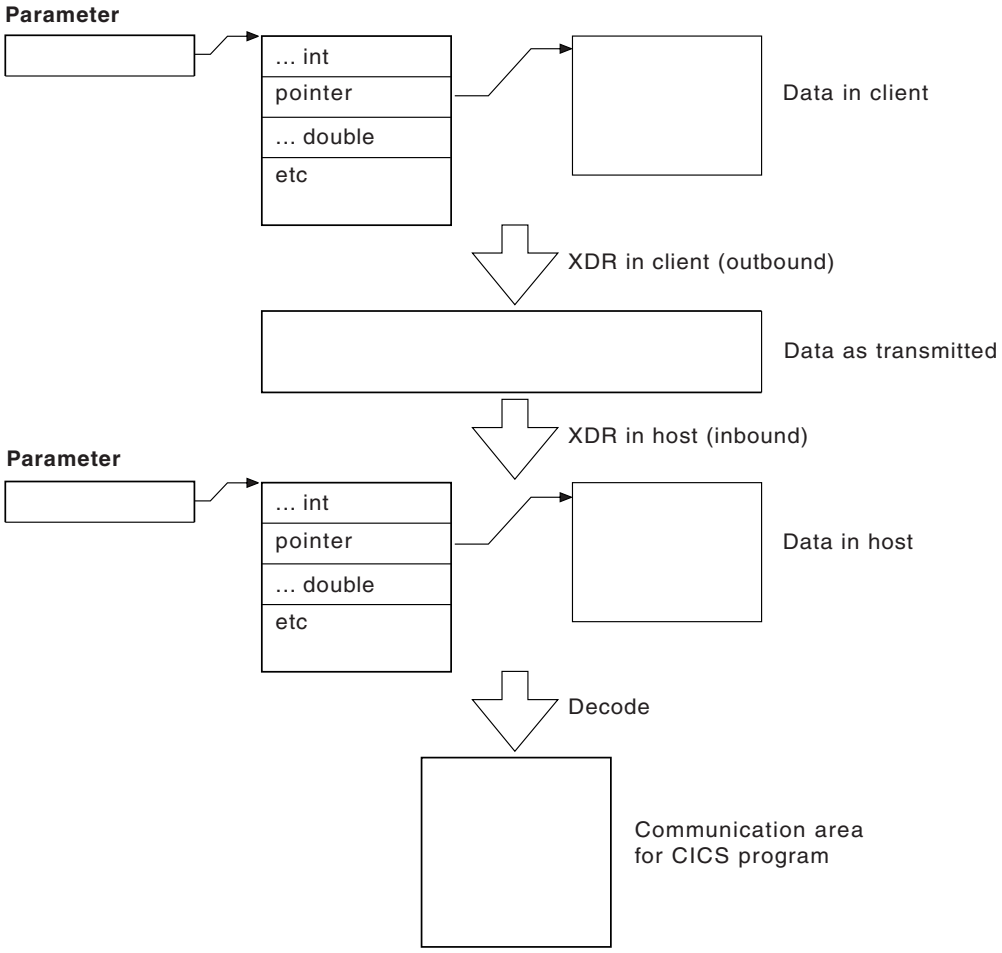


Figure 21. Data flow from client to CICS program

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 22 on page 108 illustrates these two possibilities.

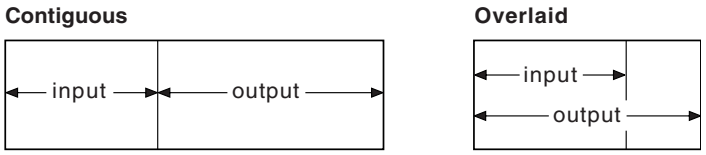


Figure 22. Use of communication area according to data format

From CICS program to client:

This shows the progress of data from the CICS program back to the client.

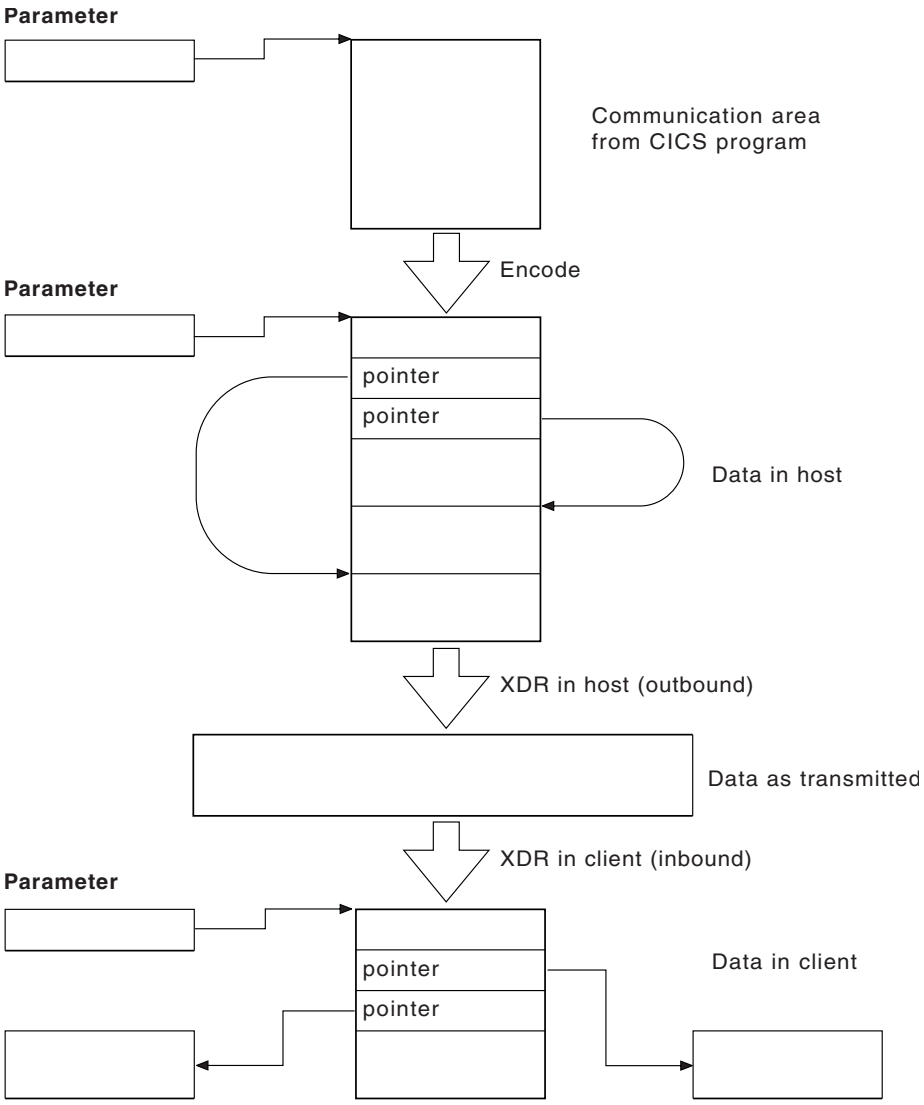


Figure 23. Data flow from CICS program to client

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.

Setting up CICS ONC RPC

CICSONC RPC allows client applications to access CICS programs by calling them as remote procedures using the ONC RPC format. Follow this information to set 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.

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 DFHPD710 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 Communications 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 121. The name of the alias can also be changed by the **Decode** function, as described in "Changing the alias and CICS program" on page 142.

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 121.

EXECKEY option:

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.

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).

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.

CICS operates with storage protection when the system initialization parameter STGPROT is set to YES, or allowed to default to YES.

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 111 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.

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

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 24. 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 25. 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.

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.

(See Figure 24 on page 115.)

Select an option, then press Enter.

Option For more information see:

1 “Enabling CICS ONC RPC” on page 118

2 “Updating the CICS ONC RPC data set” on page 130

Starting the connection manager when CICS ONC RPC is enabled

If CICS ONC RPC is enabled, panel DFHRP04 is shown.

(See Figure 25 on page 115.)

Select an option, then press Enter.

Option For more information see:

1 "Disabling CICS ONC RPC" on page 128

2 "Updating the CICS ONC RPC data set" on page 130

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 26. 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 120

3 "Unregistering 4-tuples" on page 126

4 "Processing the alias list" on page 134

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 27. 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 28. 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 28, 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 following fields. 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 128.

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 29 on page 121.) 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 29. Panel DFHRP03

Option For more information see:

1 See information later in this section.

2 "Defining the attributes of a 4-tuple"

3 "Unregistering 4-tuples" on page 126

4 See information later in this section.

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 30. 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 "Step 3—Write the XDR routines" on page 138 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 "Step 3—Write the XDR routines" on page 138 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 111.

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.

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 121. 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 121. 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 133. 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 133. 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 133. 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 133.

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 121. To register the definition, press Enter.
- You can register 4-tuples from a list. See "Working with a list of 4-tuples" on page 132.
- 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 31 on page 127.)

```
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 31. 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 33 on page 128.)

```
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 32. 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 33. 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 34. 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 35. 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 131
2 "Working with a list of 4-tuples" on page 132
3 "Changing the attributes of a 4-tuple" on page 133
4 "Changing the attributes of a 4-tuple" on page 133

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 36. 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 37. 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 133 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 38. 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 122.
- CICS attributes—see EDF Terminal ID.
- CICS ONC RPC attributes—see "CICS ONC RPC attributes" on page 124.

```
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 39. 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 41 on page 136.)

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 40. 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 41. Panel DFHRP18

Developing CICS ONC RPC applications

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 42 on page 137 , which showed the process for ONC RPC without CICS ONC RPC.

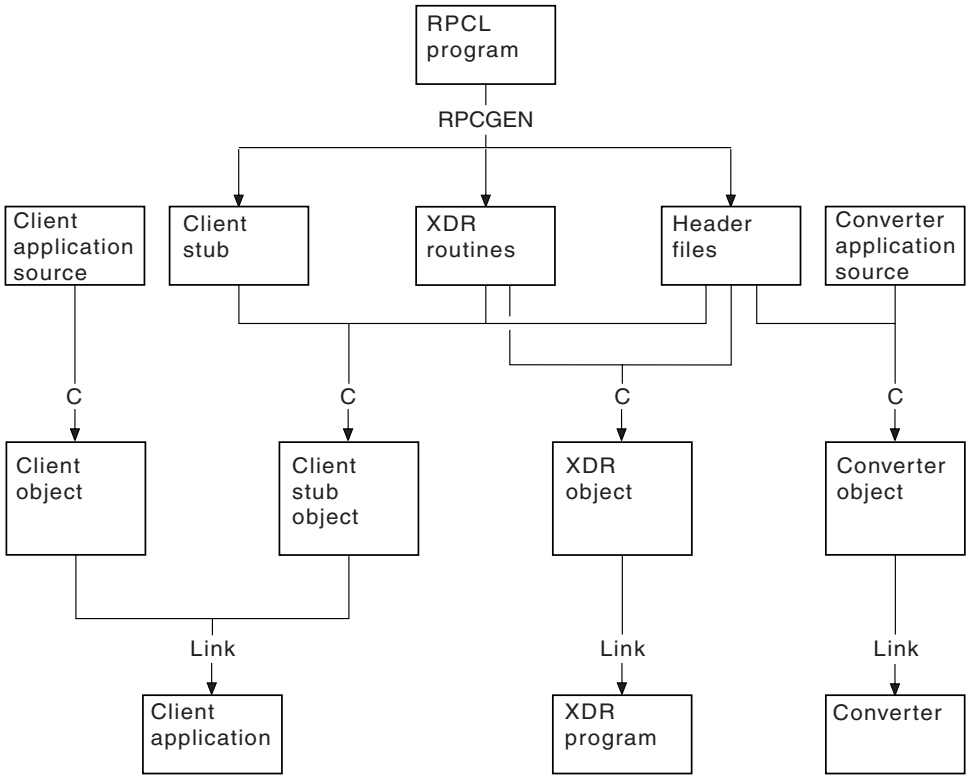


Figure 42. Program development with CICS ONC RPC

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 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 in the following steps. 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 16 on page 139), 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. CICS ONC RPC security in Reference tells you about this and other security facilities available for use with CICS ONC RPC. Writing the resource checker 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(r) ONC RPC resources to CICS(r) .
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.

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 16 on page 139), you do not need to write an XDR routine. Instead you specify one of the XDR library functions described later 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 z/OS Communications Server: IP Programmer's Guide and Reference .

CICS ONC RPC supports only the functions listed in the following table. 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 16. 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 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 z/OS Communications Server: IP Configuration Reference.

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" , using reference information supplied in "Reference information for the converter functions" on page 149.

Step 5: Write a resource checker

This step is optional.

See Writing the resource checker 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 142.

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(r) ONC RPC resources to CICS(r) .

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.

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 150 , “Decode” on page 152 , and “Encode” on page 157.

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
- 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 150 , “Decode” on page 152 , and “Encode” on page 157.

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 43 on page 144 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 43. 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 44 on page 145 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 44. 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 45 on page 147 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 46 on page 148 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 45. 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 46. 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 47 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 47. 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 151.

glength_response
(Output only)

A response code—see “Response and reason codes” on page 151.

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 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 function client_address client_data_ptr server_data_format program_number version_number procedure_number aup_time aup_machname_ptr aup_machlen aup_uid aup_gid aup_len aup_gids_ptr	server_program alias_transid server_input_data_len server_output_data_len	returned_data_ptr userid user_token response reason

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 142.

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 156.

decode_response
(Output only)

A response code; see "Response and reason codes" on page 156.

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 for 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 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. Otherwise, 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 for the numeric values of the response in trace output.

Security for ONC RPC

Important: This information contains Product-sensitive Programming Interface and Associated Guidance Information.

Security is an important concern in the provision of ONC RPC support in the CICS environment, because CICS ONC RPC provides an Open Systems communications interface into CICS.

ONC RPC has its own security methods (called authentication in RPC) with dedicated fields in the ONC RPC call and reply message headers. There are three types of RPC authentication:

- **UNIX authentication**, which is used to transmit the client's UNIX user ID, group ID, and other identification information.
- **Data Encryption Standard (DES) authentication**, which is not available at ONC RPC Version 3.9, and so cannot be used with CICS ONC RPC.
- **Null authentication**, which offers no security checking.

This section describes how CICS ONC RPC interacts with the security facilities of ONC RPC and CICS.

Security in ONC RPC

ONC RPC has its own security methods (called authentication in RPC) with dedicated fields in the ONC RPC call and reply message headers.

There are three types of RPC authentication:

- **UNIX authentication**, which is used to transmit the client's UNIX user ID, group ID, and other identification information.
- **Data Encryption Standard (DES) authentication**, which is not available at ONC RPC Version 3.9, and so cannot be used with CICS ONC RPC.
- **Null authentication**, which offers no security checking.

Security in CICS and its effect on CICS ONC RPC operations

During the operation of CICS ONC RPC, various CICS commands are used to make security checks with an external security manager (ESM).

The checks will always give positive results if SEC=NO is specified as a system initialization parameter. The checks will always give negative results if SEC=YES was specified, but the ESM abended while CICS was operating. The following discussion of the use made of CICS security commands assumes that SEC=YES is specified, and that the ESM is active.

- When a transaction whose user ID is *userid1* issues EXEC CICS START USERID(*userid2*), a surrogate-user check is made with the ESM to see that *userid1* is authorized to use *userid2*. The check is made only if XUSER=YES is specified as a system initialization parameter.

  This command is issued when the connection manager starts the server controller, and each time the server controller starts an alias transaction. In the first case, the user ID used is the one supplied to the connection manager as CRPM Userid on panel DFHRP02. In the second case, the user ID used is the one output from **Decode**.

- EXEC CICS VERIFY PASSWORD is issued by the alias before it links to the CICS program that services the client request. A check is made with the ESM that the user ID and password are an acceptable combination.
- EXEC CICS QUERY SECURITY is used by the alias to check that the user ID under which it is executing is authorized to use the CICS program. The check is made only if XPPT=YES is specified as a system initialization parameter.
- During the operation of the CICS program, security checks are made each time the program tries to access a protected resource. The check is made only if RESSEC(YES) is specified in the definition of the alias transaction, and the system initialization parameter controlling security checking for the resource type is set to YES.
- During the operation of the CICS program, security checks are made each time the program tries to use a command from the CICS SPI (system programming interface). The check is made only if CMDSEC(YES) is specified in the definition of the alias transaction, and if XCMD=YES is specified as a system initialization parameter.

Figure 48 shows how CICS security interacts with the operation of CICS ONC RPC.

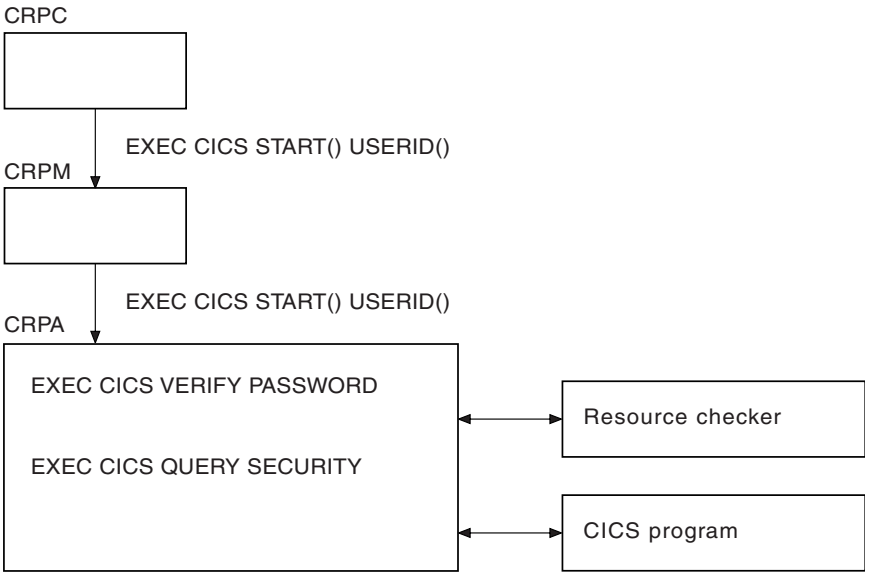


Figure 48. How CICS security interacts with CICS ONC RPC operations

The figure shows that the alias will link to the user-supplied resource checker program if one is configured, but the use of the resource checker program is not recommended. You should use the CICS security facilities, and make the appropriate definitions in the ESM.

RACF Secured Sign-on for ONC RPC clients

RACF Secured Sign-on support allows RPC clients to gain security access to CICS facilities by sending a PassTicket. This avoids the security hazard of a password being transmitted across the network in clear text.

For further information, see *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 for ONC RPC clients: The algorithm that generates the PassTicket for an ONC RPC client is a function of the following items:

- The CICS user ID of the client.
- The CICS application ID of the CICS region running CICS ONC RPC.
- A secured sign-on application key, known to both sides.
- A time and date stamp.

To generate the PassTicket, the RPC client must:

- Know its CICS user ID, the server CICS application ID, and the application key.
- Synchronize its clock to within ten minutes of the server.
- Have access to the encryption algorithm on its machine. Only the DES algorithm may be used.

Writing the resource checker

Your resource checker program must be called DFHRPRSC. There can be only one resource checker in a CICS region.

The resource checker allows you to check the credentials of inbound client requests.

The resource checker can check the client address, passed as an input parameter, against a list of known clients for the host on which the request has been received. The password passed to the resource checker is blank.

Reference information for the resource checker

The resource checker is optionally invoked by the alias before it attempts to link to the CICS program that is to service the client request. It must say whether the client request is allowed to proceed.

The reference information for the resource checker is presented as follows:

- A summary table of parameters, showing which are for input only, and which for output only.
  - **Input** is for parameters that your resource checker may consult, but not change.
  - **Output** is for parameters that your resource checker must not consult, but may change.
- A description of the processing that the resource checker is expected to do.
- A list of parameters in alphabetical order, with a description of how CICS ONC RPC sets up the inputs, and what use it makes of the outputs.
- A list of the responses and reason codes that the resource checker can return, with a description of the action that CICS ONC RPC takes for each response and reason code.

The descriptions give the names of the program elements as they appear in C. In COBOL the names are all in uppercase, and the underscores are replaced by hyphens.

Summary of parameters

The format of the communication area containing the resource checker parameters is in the C header file DFHRPRDH, and the COBOL copybook DFHRPRDO. You will also need values defined in the C header file DFHRPUCH, or in the COBOL copybook DFHRPUCO.

Input	Output
res_check_alias_transid **res_check_cics_password_ptr** **res_check_cics_userid** **res_check_client_ip_address** **res_check_eyecatcher** **res_check_host_ip_address** **res_check_server_program_name**	**res_check_reason** **res_check_response**

Parameters

res_check_alias_transid
(Input only)

The 4-character name of the alias transaction that has linked to the resource checker.

res_check_cics_password_ptr
(Input only)

A pointer to the 8-character password passed from the requesting client or supplied by **Decode**. The value of this field is blank, and it is provided for compatibility with earlier versions of CICS ONC RPC.

res_check_cics_userid
(Input only)

The 8-character CICS user ID under which the alias is running.

res_check_client_ip_address
(Input only)

The fullword internet address of the client.

res_check_eyecatcher
(Input only)

A string of length 8. (Its value is defined in the header file DFHRPUCH and the copybook DFHRPUCO).

res_check_host_ip_address
(Input only)

The fullword internet address of the z/OS Communications Server host with which the server controller is in communication.

res_check_reason
(Output only)

The reason to be returned to the alias.

res_check response
(Output only)

The response to be returned to the alias.

res_check_server_program_name
(Input only)

The 8-character name of the CICS program that is to be invoked to perform the server function requested by the client.

Response and reason codes

You must return one of the following values in the **res_check_response** field.

URP_OK
The alias will continue to process the client request.

URP_EXCEPTION
The alias writes an exception trace entry (trace point 9F0E), and issues a message that depends on the reason code:

- URP_AUTH_BADCRED—message DFHRP0130

  An **svcerr_auth** call with a why-value of AUTH_BADCRED is used to send a reply to the client.
- URP_AUTH_TOOWEAK—message DFHRP0184

  An **svcerr_auth** call with a why-value of AUTH_TOOWEAK is used to send a reply to the client.
- Any other value—message DFHRP0185

  An **svcerr_systemerr** call is used to send a reply to the client.

URP_INVALID
The alias writes an exception trace entry (trace point 9F0E), and issues a message (DFHRP0186).

An **svcerr_systemerr** call is used to send a reply to the client.

URP_DISASTER
The alias writes an exception trace entry (trace point 9F0E), and issues a message (DFHRP0187).

An **svcerr_systemerr** call is used to send a reply to the client.

If you return any other value in **res_check_response**, the alias writes an exception trace entry (trace point 9F0E), and issues a message (DFHRP0188). An **svcerr_systemerr** call is used to send a reply to the client.

You can supply a 32-bit reason code in conjunction with the response value to provide further information in error cases. CICS ONC RPC does not take any action on the reason code returned by the resource checker, except as indicated previously under URP_EXCEPTION. The reason code is output in any trace or messages that result from the resource checker, and you may use it as a debugging aid.

See Numeric values of response and reason codes for the numeric values of the response and CICS-defined reason codes in trace output.

Troubleshooting ONC

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.

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 are listed in CICS messages.

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.

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. 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.

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 timeouts 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.
   - 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?
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.

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.

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 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.

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 DFHPD710 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 Administering.

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 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.

Improving ONC RPC performance

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 timeout 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.

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.

Chapter 4. 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 "DCE remote procedure calls" on page 175 .

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, you should refer to the books listed in "DCE remote procedure calls" on page 175 .

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.

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.

© Copyright IBM Corp. 1994, 2017

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 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.

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 OS/390® 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 UNIX System Services 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: Restrictions apply to the COBOL and C data structures that can be defined by 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 Application programming for DCE remote procedure calls.

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.
- Distributed Computing Environment Base Services that is provided with z/OS.
- z/OS Unix Systems Services Distributed Computing Environment Application Support MVS CICS Feature, Version 1 Release 1 or later.

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.

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 Defining CICS programs as DCE servers.

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 Reference -> Application development.
- 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. (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 z/OS DCE 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.

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. 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.

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. 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.

© Copyright IBM Corp. 1994, 2017

Notices

This information was developed for products and services offered in the U.S.A. This material might be available from IBM in other languages. However, you may be required to own a copy of the product or product version in that language in order to access it.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property rights may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive, MD-NC119
Armonk, NY 10504-1785
United States of America

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan Ltd.
19-21, Nihonbashi-Hakozakicho, Chuo-ku
Tokyo 103-8510, Japan

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION “AS IS” WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE. Some jurisdictions do not allow disclaimer of express or implied warranties in certain transactions, therefore this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM websites are provided for convenience only and do not in any manner serve as an endorsement of those websites. The materials at those websites are not part of the materials for this IBM product and use of those websites is at your own risk.

© Copyright IBM Corp. 1994, 2017

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who want to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact

IBM Director of Licensing
IBM Corporation
North Castle Drive, MD-NC119
Armonk, NY 10504-1785
US

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.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to actual people or business enterprises is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. The sample programs are provided "AS IS", without warranty of any kind. IBM shall not be liable for any damages arising out of your use of the sample programs.

Programming interface information

CICS supplies some documentation that can be considered to be Programming Interfaces, and some documentation that cannot be considered to be a Programming Interface.

Programming Interfaces that allow the customer to write programs to obtain the services of CICS Transaction Server for z/OS, Version 5 Release 4 are included in the following sections of the online product documentation:

- Developing applications
- Developing system programs
- RACF security overview
- Developing for external interfaces
- Reference: application developmenth
- Reference: system programming
- Reference: connectivity

Information that is NOT intended to be used as a Programming Interface of CICS Transaction Server for z/OS, Version 5 Release 4 , but that might be misconstrued as Programming Interfaces, is included in the following sections of the online product documentation:

- Troubleshooting and support
- Reference: diagnostics

If you access the CICS documentation in manuals in PDF format, Programming Interfaces that allow the customer to write programs to obtain the services of CICS Transaction Server for z/OS, Version 5 Release 4 are included in the following manuals:

- Application Programming Guide and Application Programming Reference
- Business Transaction Services
- Customization Guide
- C++ OO Class Libraries
- Debugging Tools Interfaces Reference
- Distributed Transaction Programming Guide
- External Interfaces Guide
- Front End Programming Interface Guide
- IMS Database Control Guide
- Installation Guide
- Security Guide
- Supplied Transactions
- CICSPlex SM Managing Workloads
- CICSPlex SM Managing Resource Usage
- CICSPlex SM Application Programming Guide and Application Programming Reference
- Java Applications in CICS

If you access the CICS documentation in manuals in PDF format, information that is NOT intended to be used as a Programming Interface of CICS Transaction Server for z/OS, Version 5 Release 4 , but that might be misconstrued as Programming Interfaces, is included in the following manuals:

- Data Areas
- Diagnosis Reference
- Problem Determination Guide
- CICSPlex SM Problem Determination Guide

Trademarks

IBM, the IBM logo, and ibm.com® are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at Copyright and trademark information at www.ibm.com/legal/copytrade.shtml.

Adobe, the Adobe logo, PostScript, and the PostScript logo are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, and/or other countries.

Intel, Intel logo, Intel Inside, Intel Inside logo, Intel Centrino, Intel Centrino logo, Celeron, Intel Xeon, Intel SpeedStep, Itanium, and Pentium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Terms and conditions for product documentation

Permissions for the use of these publications are granted subject to the following terms and conditions.

Applicability
These terms and conditions are in addition to any terms of use for the IBM website.

Personal use
You may reproduce these publications for your personal, noncommercial use provided that all proprietary notices are preserved. You may not distribute, display or make derivative work of these publications, or any portion thereof, without the express consent of IBM.

Commercial use
You may reproduce, distribute and display these publications solely within your enterprise provided that all proprietary notices are preserved. You may not make derivative works of these publications, or reproduce, distribute or display these publications or any portion thereof outside your enterprise, without the express consent of IBM.

Rights Except as expressly granted in this permission, no other permissions, licenses or rights are granted, either express or implied, to the publications or any information, data, software or other intellectual property contained therein.

IBM reserves the right to withdraw the permissions granted herein whenever, in its discretion, the use of the publications is detrimental to its interest or, as determined by IBM, the above instructions are not being properly followed.

You may not download, export or re-export this information except in full compliance with all applicable laws and regulations, including all United States export laws and regulations.

IBM MAKES NO GUARANTEE ABOUT THE CONTENT OF THESE PUBLICATIONS. THE PUBLICATIONS ARE PROVIDED "AS-IS" AND WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, AND FITNESS FOR A PARTICULAR PURPOSE.

IBM online privacy statement

IBM Software products, including software as a service solutions, (“Software Offerings”) may use cookies or other technologies to collect product usage information, to help improve the end user experience, to tailor interactions with the end user or for other purposes. In many cases no personally identifiable information is collected by the Software Offerings. Some of our Software Offerings can help enable you to collect personally identifiable information. If this Software Offering uses cookies to collect personally identifiable information, specific information about this offering’s use of cookies is set forth below:

For the CICSPlex SM Web User Interface (main interface):
Depending upon the configurations deployed, this Software Offering may use session and persistent cookies that collect each user’s user name and other personally identifiable information for purposes of session management, authentication, enhanced user usability, or other usage tracking or functional purposes. These cookies cannot be disabled.

For the CICSPlex SM Web User Interface (data interface):
Depending upon the configurations deployed, this Software Offering may use session cookies that collect each user's user name and other personally identifiable information for purposes of session management, authentication, or other usage tracking or functional purposes. These cookies cannot be disabled.

For the CICSPlex SM Web User Interface ("hello world" page):
Depending upon the configurations deployed, this Software Offering may use session cookies that collect no personally identifiable information. These cookies cannot be disabled.

For CICS Explorer®:
Depending upon the configurations deployed, this Software Offering may use session and persistent preferences that collect each user’s user name and password, for purposes of session management, authentication, and single sign-on configuration. These preferences cannot be disabled, although storing a user's password on disk in encrypted form can only be enabled by the user's explicit action to check a check box during sign-on.

If the configurations deployed for this Software Offering provide you, as customer, the ability to collect personally identifiable information from end users via cookies and other technologies, you should seek your own legal advice about any laws applicable to such data collection, including any requirements for notice and consent.

For more information about the use of various technologies, including cookies, for these purposes, see IBM Privacy Policy and IBM Online Privacy Statement, the section entitled “Cookies, Web Beacons and Other Technologies” and the IBM Software Products and Software-as-a-Service Privacy Statement.

Index

Numerics

3270 bridge
  benefits 9
  bridge facility properties 43
  mechanism 12
  Transaction routing 18
4-tuple 101
4-tuple records 118

A

ADS 13
ADSD 14
AIBRIDGE 39
alias (CICS ONC RPC)
  definition 111
  role in call processing 104
  specifying 123
  specifying EDF terminal ID 123
  transaction definition 111
alias transaction 103
ANYNET software 109
Application data structure (ADS) 13
Application data structure descriptor (ADSD) 14
application programming
  DCE remote procedure calls 178
authentication, RPC 160
Automatic Enable option 120, 132

B

batched RPC requests 101
benefits of external CICS interface 8
BMS ACCUM option 14
BMS and the Link 3270 bridge 14
BMS support 14
bridge
  ADS 13
  ADSD 14
  bridge facility definition 39
  BRIH copybooks 49
  BRIV copybooks 49
  conversational transactions 32
  driver 12
  dynamic routing 48
  inbound vectors 58
  INQUIRE AUTOINSTALL 46
  INQUIRE BRFACILITY 46
  INQUIRE TASK 47
  INQUIRE TRACETYPE 47
  INQUIRE TRANSACTION 47
  load routing 48
  mechanism 12
  message formats 49
  message header (BRIH) 50
  outbound vectors 64
  pseudoconversational transactions 33
  router 12
  SET BRFACILITY 46
bridge *(continued)*
  SET TRACETYPE 47
  system initialization parameters 39
bridge (3270) 11
  benefits 9
  BMS ACCUM option 14
  BMS support 14
  bridge facility definition 39
  bridge facility properties 43
  CEDF 15
  CEDX 15
  CSFE 15
  CSGM 15
  DLI 16
  FACILITYLIKE 39
  global user exits 15, 16
  inbound vectors 58
  INQUIRE AUTOINSTALL 46
  INQUIRE BRFACILITY 46
  INQUIRE TASK 47
  INQUIRE TRACETYPE 47
  INQUIRE TRANSACTION 47
  ISSUE PASS 16
  ISSUE PRINT 16
  mechanism 12
  message header (BRIH) 50
  Monitoring 16
  outbound vectors 64
  programming restrictions 14
  security 16
  SET BRFACILITY 46
  SET TRACETYPE 47
  START 16
  STARTed transactions 17
  system initialization parameters 39
  TCTUA 17
  use of ASSIGN 14
BRIH 50
  inbound parameters 51
  outbound message header 56
BRIV outbound 64
BRIV- inbound 58
BRMAXKEEPTIME 39
broadcast RPC 101

C

callback RPC 101
CEDF 15
CEDX 15
CICS external interfaces 1
CICS ONC RPC data set 118
CICS ONC RPC definition record 118
CICS ONC RPC options 119
CICS program
  API restrictions 95
CICS system initialization parameters
  SEC 160
  XCMD 161
  XPPT 161
  XUSER 160
CICS TCP/IP Socket Interface 109
CICS-key storage 112
CICSDATAKEY option on GETMAIN 153, 158
client stub 98
clients supported by CICS ONC RPC 109
CMDSEC 112, 161
code page conversions 139
command security 112
compiling 140
connection manager 102
connection manager (CICS ONC RPC)
  definition 111
  panel format 116
connection manager panels
  DFHRP01 114
  DFHRP02 119
  DFHRP03 120
  DFHRP04 114
  DFHRP06 128
  DFHRP10 117
  DFHRP11 126
  DFHRP12 127
  DFHRP13 127
  DFHRP14 132
  DFHRP15 132
  DFHRP16 117
  DFHRP17 134
  DFHRP18 134
  DFHRP20 130
  DFHRP21 133
  DFHRP22 131
  DFHRP2B 134
  DFHRP5 121
  DFHRP5B 122
connection-oriented data transmission 5
connectionless data transmission 5
contiguous communication area 151, 155
converter 103
converter (CICS ONC RPC)
  defining functions provided 124
  role in call processing 104
  writing 140
CRPA transaction 103
CRPC transaction 102, 114
CRPM transaction 102
CRPO transient data queue 113, 116
CSFE 15
CSGM 15

D

data format 107, 124, 151, 155
datagram 5
DCE (distributed computing environment) 8
  and CICS 173
  benefits of 8
  overview 173

© Copyright IBM Corp. 1994, 2017

DCE (distributed computing environment) *(continued)*
remote procedure call (RPC)
application programming 178
benefits of 177
calling CICS programs 179
CICS server programs 177
overview 176
requirements for use with CICS 177
resource definition 179
DCE RPC servers 7
Decode function 104
decode_alias_transid field 153
decode_aup_gid field 154
decode_aup_gids_ptr field 154
decode_aup_len field 154
decode_aup_machlen field 154
decode_aup_machname_ptr field 154
decode_aup_time field 154
decode_aup_uid field 154
decode_client_address field 154
decode_client_data_ptr field 154
decode_eyecatcher field 154
decode_function field 154
decode_procedure_number field 155
decode_program_number field 155
decode_reason field 155
decode_response field 155
decode_returned_data_ptr field 155
decode_server_data_format field 155
decode_server_input_data_len field 141, 156
decode_server_output_data_len field 141, 156
decode_server_program field 156
decode_user_token field 156
decode_userid field 156
decode_version_number field 156
DES authentication 160
disable processing
types 129
disabling CICS ONC RPC 128
distributed computing environment (DCE) 8
and CICS 173
benefits of 8
overview 173
remote procedure call (RPC)
application programming 178
benefits of 177
calling CICS programs 179
CICS server programs 177
overview 176
requirements for use with CICS 177
resource definition 179
DLI 16
driver
bridge 12
dumps 168

E

EDF 169
enabling CICS ONC RPC 120
Encode (CICS ONC RPC)
whether required 157
Encode function 104
encode_eyecatcher field 158
encode_function field 158
encode_input_data_len field 158
encode_input_data_ptr field 158
encode_output_data_len field 159
encode_output_data_ptr field 159
encode_reason field 159
encode_response field 159
encode_user_token field 159
ENTER TRACENUM command 169
ephemeral port numbers 7
EXEC CICS GETMAIN
CICSDATAKEY option in Decode 153
CICSDATAKEY option in Encode 158
FLENGTH option in Decode 153
FLENGTH option in Encode 158
NOSUSPEND option in Decode 153
SHARED option in Decode 153
SHARED option in Encode 158
EXEC CICS QUERY SECURITY 161
EXEC CICS START USERID 160
EXEC CICS SYNCPOINT 106
EXEC CICS VERIFY PASSWORD 161
EXEC CICS WRITEQ TD 170
EXECKEY option 112
external CICS interface (EXCI)
benefits 8
external interfaces 1

F

FACILITYLIKE 39
fast-path commands 115
File Transfer Protocol 6
FLENGTH option on GETMAIN 153, 158

G

Getlengths function 103
whether required 141
glength_eyecatcher field 150
glength_function field 150
glength_reason field 150
glength_response field 150
glength_server_data_format field 141, 151
glength_server_input_data_len field 141, 151
glength_server_output_data_len field 141, 151
global user exits 15, 16

I

inbound XDR routine 103, 105, 107
internet address 5
Internet address 5
Internet Protocol (IP) 4
IP address 5
IPCS VERBEXIT 110
IPv4 addresses and IPv6 addresses 5
ISSUE PASS 16
ISSUE PRINT 16

J

JCL for dump formatting
CICS ONC RPC 110

L

Language Environment 109
Link3270
Transactuion routing 18
linking 140
local resources, defining
CICS programs as DCE servers 179

M

mapset definition 113
messages 167
migrating between CICS versions 111
Monitoring 16

N

nonblocking call type
specifying 123
nonblocking RPC call 101
NOSUSPEND option on GETMAIN 153
Null authentication 160

O

ONC 97
ONC RPC 96
open system interface (OSI) 8
OSI (open system interface) 8
outbound XDR routine 103, 106, 109
overlaid communication area 151, 155

P

PassTicket 162
performance 170
port number 5
Portmapper 100
prerequisites for CICS ONC RPC 109
procedure number 99
procedure zero 100
program number 99
programs for CICS ONC RPC
defining in CICS 112
protocol 101

Q

QR TCB 166

R

RACF Secured Sign-on 161
Register from Data Set option 125
registering 4-tuples 125
registration
  with CICS ONC RPC 126
  with TCP/IP for MVS 126
remote procedure call (RPC)
  benefits of 177
  calling CICS programs 179
  CICS server programs 177
    resource definition 179
  overview 176
  requirements for use with CICS 177
remote procedures
  procedure number 99
  procedure zero 100
  program number 99
  version number 99
REMOTENAME parameter 113
REMOTESYSTEM parameter 113
resource checker (CICS ONC RPC)
  specifying option 119, 131
  writing 162
resource definition
  DCE remote procedure call
    server programs 179
resource definition in CICS 111
resource security 112
RESSEC 112, 161
restrictions
  bridge 14
router
  bridge 12
RP TCB 166
RPC 96
RPC library calls 99
RPCGEN compiler 98
RPCL specification
  definition 98

S

SBCS translate tables 139
SEC system initialization parameter 160
Secured Sign-on 161
security 16, 160
server application set 138
server controller 102
  user ID 120, 132
server controller (CICS ONC RPC)
  definition 111
  role in call processing 104
server stub 98
SHARED option on GETMAIN 153, 158
sockets interface 5
START 16
STARTed transactions 17
starting the connection manager 114
STGPROT parameter 112
storage (CICS ONC RPC)
  user-key/CICS-key 112
  XDR routines overwriting 139
storage protection 112
storage requirements (CICS ONC RPC) 110
synchronous RPC call 100

T

task control blocks 166
task-related user exit (TRUE) 166
TASKDATAKEY option 112
TCP/IP 4, 97
TCP/IP for MVS
  CICS TCP/IP Socket Interface 109
TCTUA 17
Telnet 6
trace (CICS ONC RPC)
  information 168
  setting trace level 119, 131
  setting trace option 119, 131
Transaction routing
  Link3270 18
transient data definitions 113
Transmission Control Protocol (TCP) 5

U

UNIX authentication 160
URP_DISASTER response
  to resource checker 164
URP_DISASTER response (CICS ONC RPC)
  to Decode 157
  to Encode 159
  to Getlengths 152
URP_EXCEPTION response
  to resource checker 164
URP_EXCEPTION response (CICS ONC RPC)
  to Decode 157
  to Encode 159
  to Getlengths 152
URP_INVALID response
  to resource checker 164
URP_INVALID response (CICS ONC RPC)
  to Decode 157
  to Encode 159
  to Getlengths 152
URP_OK response
  to resource checker 164
URP_OK response (CICS ONC RPC)
  to Decode 156
  to Encode 159
  to Getlengths 151
use of ASSIGN 14
User Datagram Protocol (UDP) 5
user-key storage 112

V

vector
  default vectors 28
  inbound BRIV vectors 27
  outbound BRIV vectors 27
version number 99

W

well-known ports 6

X

XCMD system initialization parameter 161
XDR 97
XDR routines 103
  inbound 103, 105, 107
  library functions 138
  outbound 103, 106, 109
  specifying 123
  writing 138
XLT definitions 114
XPPT system initialization parameter 161
XUSER system initialization parameter 160

Z

z/OS Communications Server 166

