CICS® Transaction Server for VSE/ESA™

CICS External Interfaces Guide

Release 1

SC33-1669-01

CICS® Transaction Server for VSE/ESA™

CICS External Interfaces Guide

Release 1

SC33-1669-01

Note!

Before using this information and the product it supports, be sure to read the general information under “Notices” on page 205.

First Edition (February 2000)

This edition applies to Release 1 of the IBM licensed program CICS Transaction Server for VSE/ESA, program number 5648-054, and to all subsequent versions, releases, and modifications until otherwise indicated in new editions. Consult the latest edition of the applicable IBM system bibliography for current information on this product.

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

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

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

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

Preface

What this book is about

This book describes how you can make the CICS® transaction processing services of CICS TS for VSE/ESA® available to a variety of external users.

How to use this book

Read “Part 1. Overview” on page 1 for planning information, and for guidance about which other parts of the book to consult.

What you need to know to understand this book

Within this book, referfence is made to Release 1. The function described in this book has been added since the release of CICS Transaction Server for VSE/ESA Release 1 and can be identified within the code as 1.1.1.

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

Notes on terminology

When the term “CICS” is used without any qualification in this book, it refers to the CICS element of IBM® CICS Transaction Server for VSE/ESA.

Contents

Preface . . . iii
What this book is about . . . iii
How to use this book . . . iii
What you need to know to understand this book . . . iii
Notes on terminology . . . iii

Summary of changes . . . xi

Part 1. Overview . . . 1

Chapter 1. Introduction . . . 3
General concepts . . . 6
Distributed computing . . . 6
    Security support . . . 7
TCP/IP protocols. . . . 7
    TCP/IP internet addresses and ports . . . 8
Programming models . . . 9
Comparing mechanisms . . . 10
    EXCI . . . 10
    The 3270 bridge . . . 10
    The 3270 Bridge and FEPI . . . 11
Application design . . . 11
    Separating business and presentation logic . . . 12

Chapter 2. How to use this book . . . 15

Part 2. Bridging to 3270 transactions . . . 17

Chapter 3. Introduction . . . 19
Overview . . . 19
    Components of the 3270 bridge. . . . 20
    Bridge implementations provided . . . 25
Running 3270 transactions in a bridge environment . . . 27
    Simple terminal transactions . . . 27
    Pseudoconversational transactions . . . 28
    Conversational transactions . . . 29
Implementing a 3270 bridge environment . . . 30
    Determining the bridge environment model . . . 30
    Using the long running monitor model to implement the 3270 bridge . . . 31
    Using the two task model to implement the 3270 bridge . . . 33
    Using a single message model to implement the 3270 bridge . . . 35
    Using the direct model to implement the 3270 bridge . . . 36

Chapter 4. The Bridge environment . . . 39
User transaction programming considerations . . . 39
Defining the user transaction . . . 44
    TRANSACTION resource definition . . . 44
    PROFILE resource definition . . . 45
Inquiring about the bridge environment . . . 45
    ASSIGN command . . . 45
    INQUIRE TASK command . . . 46
    INQUIRE TRANSACTION command . . . 47

CEMT INQUIRE TASK 47
CEMT INQUIRE TRANSACTION 48
XPI commands 48
The bridge facility 49
QUERY 50
SET TERMINAL/NETNAME 51
Bridge facility global user exit 51

Chapter 5. Supplied 3270 bridge exits 55
The TS/TD supplied bridge exit 55
Using the DFH0CBRE exit 55
The Web bridge exit 56
Using the Web bridge exit 57
Data formats for the supplied bridge exits 57
BRDATA format. 58
Message data format 59

Chapter 6. Writing your own bridge programs 75
Designing your own bridge solution 75
Is a new bridge exit needed?. 75
Is a new formatter needed? 75
BMS macro generation utility program (DFHBMSUP) 76
Overview 76
Writing your own bridge exit 78
Transaction calls to the bridge exit. 78
Message calls to the bridge exit 80
API calls to the bridge exit. 81
Writing your own formatter 81
Calls to the formatter. 81
Return codes from the formatter 82
Bridge exit area (BRXA) 84
BRXA header area 84
BRXA transaction area 86
BRXA command area 90
BRXA command area - common 92
BRXA command area - terminal control 96
BRXA command area - BMS 96
BRXA command area - interval control 98
BRXA command area - syncpoint 99
BRXA command area - MSG. 99
ADS descriptor area 100
Supplied copybooks 105
Copybook example (DFHBRACD) 105

Chapter 7. Problem determination 107
Troubleshooting 107
Defining the problem 107
Documentation about the problem 108
Using messages and codes. 108
Using Trace 108
Dump and trace formatting 109
Debugging the bridge exit 109
IDENTIFIER 109
EDF 109
Trace 110
Debugging the supplied bridge exit 110

Abend codes and Trace 110
Message validation 110

Part 3. External CICS Interface. 111

Chapter 8. Introduction to the external CICS interface 113
Overview. 113
The programming interfaces 113
Illustrations of the external CICS CALL interface 114
Illustration of the EXCI EXEC CICS interface 117
Benefits of the external CICS interface. 117
Requirements for the external CICS interface 117

Chapter 9. The EXCI CALL interface 119
The CALL interface commands 119
Initialize_User 120
Allocate_Pipe 123
Open_Pipe 125
DPL_Request 127
Close_PIPE 134
Deallocate_Pipe 136
Response code values 138
Return area for the EXCI CALL interface 138
Return area and function call EQUATE copybooks 139
Return codes 139
Dpl_retarea return codes. 139
Example of EXCI CALLs with null parameters 140

Chapter 10. The EXEC CICS interface 143
EXEC CICS LINK command 144
Retries on an EXEC CICS LINK command 148
Translation required for EXEC CICS LINK command 150

Chapter 11. Defining connections to CICS 151
CONNECTION resource definition 151
SESSIONS resource definitions for EXCI connections 153
Inquiring on the state of EXCI connections 155

Chapter 12. The EXCI user-replaceable module 157

Chapter 13. External CICS interface options table, DFHXCOPT 159

Chapter 14. Compiling and link-editing external CICS interface client programs 161
The external CICS interface stub, DFHXCSTB 161
The required linkage editor modes 161
Language considerations. 161
PL/I considerations 161
C considerations 161
Sample application programs 162
Description of the sample applications 162
Installing the EXCI sample definitions 163
Running the EXCI sample applications. 164
Results of running the EXCI sample applications 164
Job control language to run an EXCI client program 166

Chapter 15. Security 169
MRO logon and bind-time security 169
Link security 170
User security 170
Surrogate user checking 171

Chapter 16. Problem determination 173
Trace 173
System dumps 173
Formatting system dumps 173
Abends from related CICS programs 174
The EXCI service trap, DFHXCTRA. 177
EXCI trace entry points 177

Chapter 17. Response and reason codes returned on EXCI calls 189
Reason code for response: OK 189
Reason codes for response: WARNING 189
Reason codes for response: RETRYABLE 190
Reason codes for response: USER_ERROR 190
Reason codes for response: SYSTEM_ERROR 193

Chapter 18. Messages and Codes. 199

Part 4. Appendixes 201

Appendix. Routing program-link requests 203
Static routing 203
Dynamic routing 203

Notices 205
Programming interface information 206
Trademarks and service marks 206

Bibliography 209
CICS Transaction Server for VSE/ESA Release 1 Library 209

Books from VSE/ESA 2.5 base program libraries 211
VSE/ESA Version 2 Release 5. 211
High-Level Assembler Language (HLASM) 211
Language Environment for VSE/ESA (LE/VSE) 212
VSE/ICCF 212
VSE/POWER 212
VSE/VSAM 212
VTAM for VSE/ESA. 213

Books from VSE/ESA 2.5 optional program libraries 215
C for VSE/ESA (C/VSE) 215
COBOL for VSE/ESA (COBOL/VSE) 215
DB2 Server for VSE 216
DL/I VSE 216
PL/I for VSE/ESA (PL/I VSE) 216
Screen Definition Facility II (SDF II) 217
TCP/IP for VSE/ESA 217

Determining if a publication is current 219

Index 221

Sending your comments to IBM 227

Summary of changes

This is a new book to the CICS Transaction Server for VSE/ESA Release 1 Library.

Part 1. Overview

This part of the book outlines some the ways in which you can make CICS transaction processing services available to a variety of external users.

This part contains:

- “Chapter 1. Introduction” on page 3
- “Chapter 2. How to use this book” on page 15

Chapter 1. Introduction

This book describes the following sources of external requests, and the routes that they can use into CICS:

CICS 3270 Bridge
Provides a method for accessing applications that would normally require a terminal through alternatives, such as TS and TD queues or the CICS web support. See “Part 2. Bridging to 3270 transactions” on page 17.

VSE™ applications
Applications running in VSE partitions can use the External CICS Interface (EXCI) to access CICS programs. See “Part 3. External CICS Interface” on page 111.

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

User socket applications
User socket applications can use the CICS Sockets feature of TCP/IP. See *TCP/IP for VSE/ESA User’s Guide*.

Web browsers
Web browsers can use a variety of methods:

CICS Web support
The CICS Web support is a CICS-provided facility for supporting Web browsers. See the *CICS Transaction Server for VSE/ESA CICS Internet Guide.*

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

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

CICS Transaction Gateway
The CICS Transaction Gateway, a workstation application that uses a CICS client to route ECI and EPI requests to a CICS server.

CICS client applications
CICS client applications use a CICS client and the ECI or the EPI. See *CICS Family: Client/Server Programming*.

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 TS. Programs running in CICS TS can use the CICS front end programming interface (FEPI) to start transactions in the same or another instance of CICS TS. See *CICS Front End Programming Interface User’s Guide*.

Telnet clients
Telnet clients can use TN3270 to start transactions.

3270 users

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

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

Key to figure 2

TC	=	Terminal Control
TR	=	Transaction Routing
DPL	=	Distributed Program Link
EXCI	=	EXternal CICS Interface
ECI	=	External Call Interfaces
EPI	=	External Presentation Interface
CWP	=	CICS WebServer Plugin
⬭	=	Sources of external requests
▨	=	Targets of external requests
⬚	=	CICS provided interfaces
■	=	CICS components
□	=	Other product components

Figure 1.

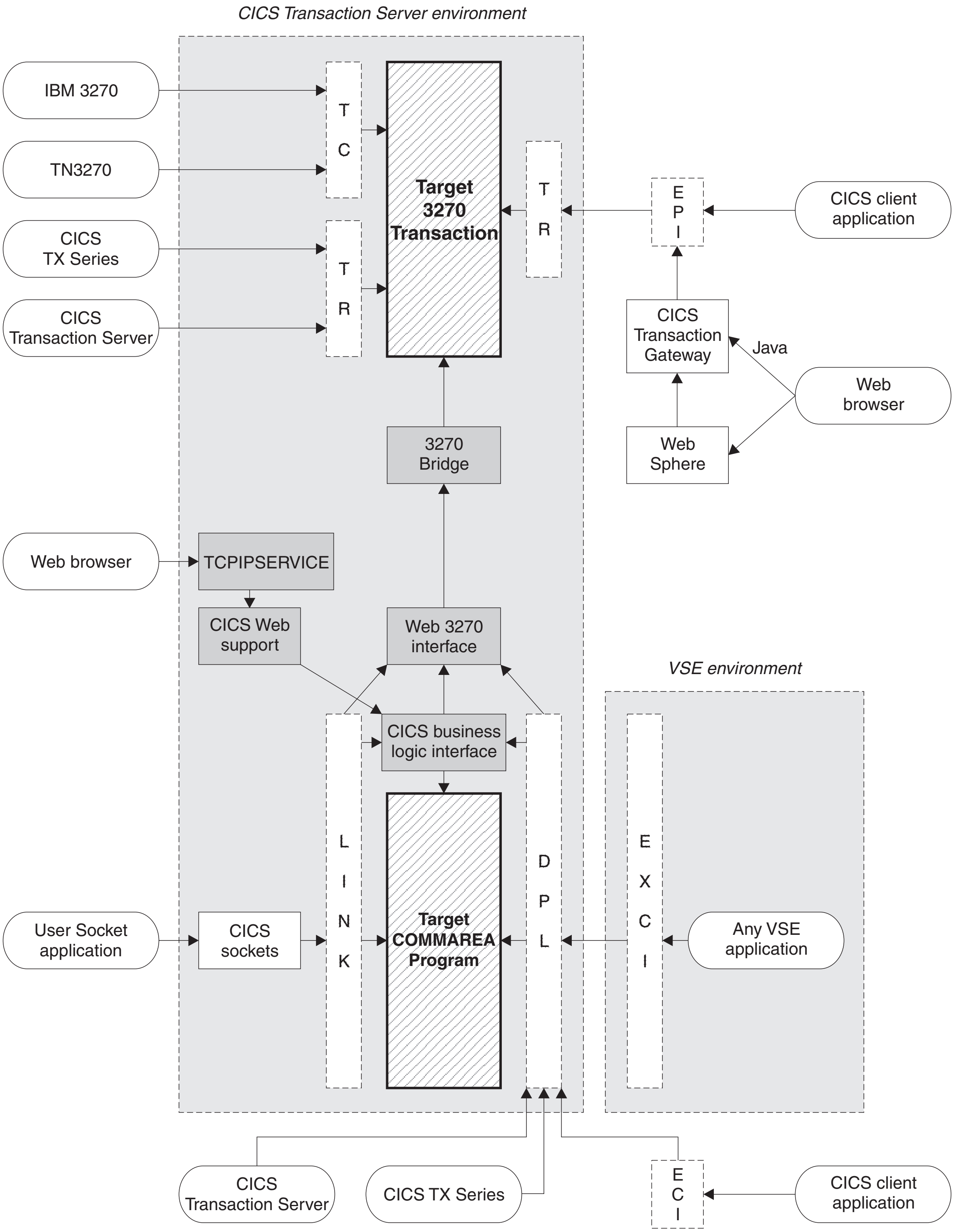


Figure 2. Client access to existing business logic

General concepts

All the mechanisms described in this book follow a similar pattern. A client is the source of the external request which comes into CICS over a network using a variety of transport protocols, or from another CICS region, using Inter Region Communication (IRC). CICS (or another product) provides a transport-specific listener (a long-running task) that starts another task (a facilitator such as an **alias** or a **mirror**), to process the incoming request. The facilitator uses CICS services to access the application.

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

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

Distributed computing

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

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

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

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

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

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

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

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 VSE/ESA server programs to be called from client programs running on a variety of operating systems. For information about CICS Clients, see the *CICS Family: Client/Server Programming* manual.

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

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

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

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

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

Security support

CICS Transaction Server for VSE/ESA supports:

- A single network signon (through the ATTACHSEC option of the DEFINE CONNECTION command)
- Authentication of the client system through bind-time security.

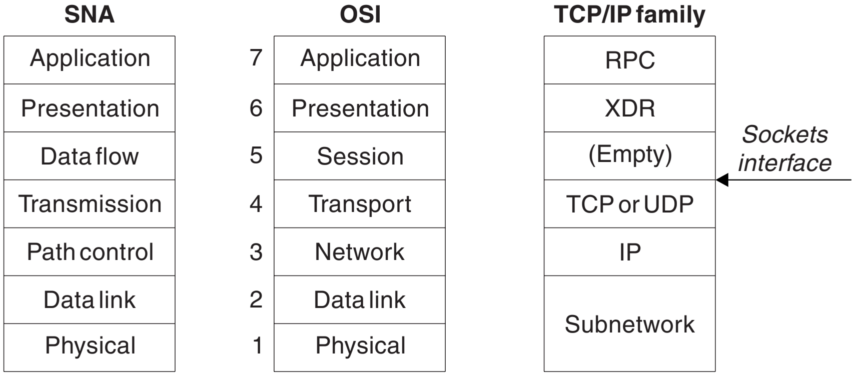
An external security manager can be used to provide resource access control and login facilities.

In all the above scenarios the client environment must know which server CICS system to communicate with. This is normally done by specifying the name of the required remote CICS system in the definition of the relevant remote CICS resource, or in the client application program.

TCP/IP protocols

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

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



SNA		OSI	TCP/IP family
Application	7	Application	RPC
Presentation	6	Presentation	XDR
Data flow	5	Session	(Empty)
Transmission	4	Transport	TCP or UDP
Path control	3	Network	IP
Data link	2	Data link	Subnetwork
Physical	1	Physical	

Figure 3. TCP/IP protocols compared to the OSI and SNA models

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

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

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

User Datagram Protocol (UDP)
UDP is also a transport-layer protocol and is an alternative to TCP. It provides a connectionless data transmission service between applications. UDP has less error checking than TCP. If UDP users want to be able to respond to errors, the communicating programs must establish their own protocol for error handling. With high-quality transmission networks, UDP errors are of little concern.

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

TCP/IP internet addresses and ports

TCP/IP provides for process-to-process communication, which means that calls need an addressing scheme that specifies both the physical host connection (Host A and Host B in Figure 4 on page 9) and the software process or application (C, D, E, F, G, and H). The way this is done in TCP/IP is for calls to specify the host by an *internet address* and the process by a *port number*. You may find internet addresses also referred to elsewhere as internet protocol (IP) addresses or host IDs.

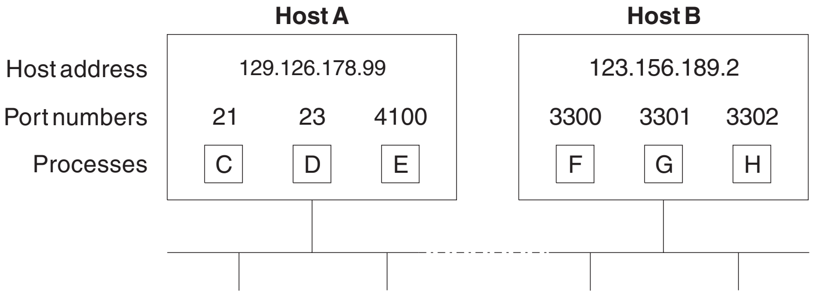


Figure 4. How applications are addressed

Internet addresses

Each host on a TCP/IP internet is identified by its internet address. An internet address is 32 bits, but it is usually displayed in dotted decimal notation. Each byte is converted to a decimal number in the range 0 to 255, and the four numbers are separated by dots thus: 129.126.178.99.

Remember that an internet is a collection of networks — hence the internet address must specify both the network and the individual host. How this is done varies with the size of the network. In the example just given, 129.126 specifies the network, 178.99 specifies the host on that network.

Port numbers (for servers)

An incoming connection request specifies the server that it wants by specifying the server's port number. For instance, in Figure 4, a call requesting port number 21 on host A is directed to process C.

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

Port numbers (for clients)

Client applications must also identify themselves with port numbers so that server applications can distinguish different connection requests. The method of allocating client port numbers must ensure that the numbers are unique; such port numbers are termed *ephemeral port numbers*. For example, in Figure 4, process F is shown with port number 3300 on host B allocated.

Programming models

The programming models implemented in CICS are inherited from those designed for 3270s, and exhibit many of the characteristics of conversational, terminal-oriented applications. There are basically three styles of programming model:

- Terminal-initiated, that is, the conversational model
- Distributed program link, that is, the RPC model
- START, that is, the queuing model.

Once initiated, the applications typically use these and other methods of continuing and distributing themselves, for example, with pseudoconversations, RETURN IMMEDIATE or DTP. The main difference between these models is in the way that they maintain state (for example, security), and hence state becomes an integral part of the application design. This presents the biggest problem when you attempt to convert to another application model.

A pseudoconversational model is mostly associated with terminal-initiated transactions and was developed as an efficient implementation of the conversational model. With increased use of 1-in and 1-out protocols such as HTTP, it is becoming necessary to add the pseudoconversational characteristic to the RPC model.

State management and its associated token management, which were previously controlled by the terminal, now need additional techniques to support this move. Similarly, when START requests are disassociated from the terminal, difficulties arise in returning the requests to their starting point.

Comparing mechanisms

This section contrasts the different mechanisms by listing some of the characteristics and benefits of each interface.

EXCI

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

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

The provision of this programming interface means that, for example, VSE 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 VSE 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-VSE platforms, and pass requests to CICS over an open system interface (OSI) using the IBM OpenEdition® Distributed Computing Environment Application Support MVS/ESA CICS feature (OE DCE AS/CICS). In this way the external CICS interface provides an open interface to a wide variety of other application platforms.

The 3270 bridge

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 VTAM.

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. Unlike other front-end methods, there is only a single unit of work.

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 recreate 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 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

Bridge	FEPI
Enabling technology	An application programming interface
Based on application data structure	Based on the 3270 data stream
Enables optimization due to integral knowledge of the target	Easier to create generic driver (data structure is architected)
Efficient; no terminal control involved	VTAM managed connection between source and target
Single COMMAREA API and user replaceable program	Requires system programming and VTAM skills
CICS specific: source and target must be in the same region	Ideal for driving remote applications, not just CICS
Driven exit decides method of communication with the client	Can be freed from the workings of the target; terminal emulation
Knowledge of UOW	No coordination
Ideal when the routing is done elsewhere	Sysplex support requires three regions
Available only for CICS TS for VSE/ESA V1R1.1 and later	Available for CICS TS for VSE/ESA V1R1.0

Application design

You can access existing applications originally designed for other environments by using the bridging facilities described, or write new ones specifically for a new environment. In general, it is good practice to split applications into a part containing the business code that is reusable, and a part responsible for presentation to the client. This technique enables you to improve performance by optimizing the parts separately, and allows you to reuse your business logic with different forms of presentation.

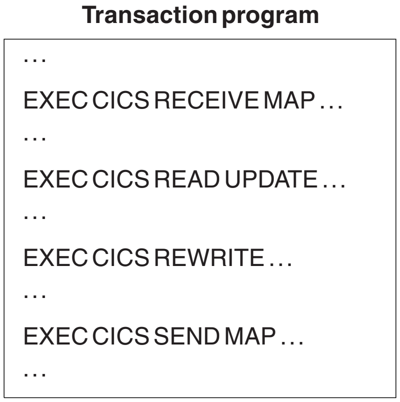
When separating the business and presentation logic, you need to consider the following:

- Avoid affinities between the two parts of the application.
- Be aware of the DPL-restricted API; see *CICS Application Programming Reference* for details.
- Be aware of hidden presentation dependencies, such as EIBTRMID usage.

Separating business and presentation logic

Figure 5 illustrates a simple CICS application that accepts data from an end user, updates a record in a file, and sends a response back to the end user. The transaction that runs this program is the second in a pseudoconversation. The first transaction has sent a BMS map to the end user’s terminal, and the second transaction reads the data with the EXEC CICS RECEIVE MAP command, updates the record in the file, and sends the response with the EXEC CICS SEND MAP command.

The EXEC CICS RECEIVE and EXEC CICS SEND MAP commands are part of the transaction’s presentation logic, while the EXEC CICS READ UPDATE and EXEC CICS REWRITE commands are part of the business logic.



Transaction program

```
...
EXEC CICS RECEIVE MAP ...
...
EXEC CICS READ UPDATE ...
...
EXEC CICS REWRITE ...
...
EXEC CICS SEND MAP ...
...
```

Figure 5. CICS functions in a single application program

A sound principle of modular programming in CICS application design is to separate the presentation logic from the business logic, and to use a communication area and the EXEC CICS LINK command to make them into a single transaction. Figure 6 on page 13 illustrates this approach to application design.

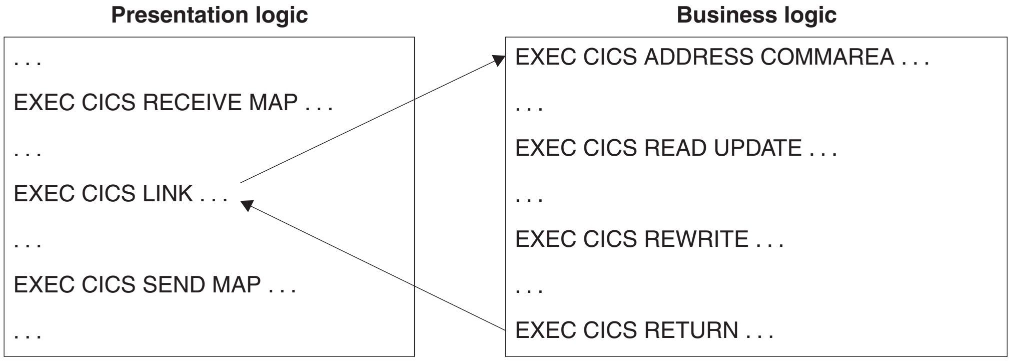


Figure 6. Separation of business and presentation logic

Once the business logic of a transaction has been isolated from the presentation logic and given a communication area interface, it is available for reuse with different presentation methods. For example, you could use CICS Web support with the CICS business logic interface, to implement a two-tier model where the presentation logic is HTTP-based.

Chapter 2. How to use this book

The rest of the manual is organized as follows:

- “Part 2. Bridging to 3270 transactions” on page 17 describes how to use the transaction bridging facilities.
- “Part 3. External CICS Interface” on page 111 describes how to use the CICS EXCI.

Part 2. Bridging to 3270 transactions

This part of the book describes the 3270 bridge. It covers the following topics:

- “Chapter 3. Introduction” on page 19
- “Chapter 4. The Bridge environment” on page 39
- “Chapter 5. Supplied 3270 bridge exits” on page 55
- “Chapter 6. Writing your own bridge programs” on page 75
- “Chapter 7. Problem determination” on page 107

Chapter 3. Introduction

This part of the book describes the function that allows you to run CICS 3270-based transactions without a 3270 terminal. It covers the following topics:

- “Overview”
- “Running 3270 transactions in a bridge environment” on page 27
- “Implementing a 3270 bridge environment” on page 30
- “The 3270 bridge” on page 10

Overview

The 3270 bridge provides an interface so that you can run 3270-based CICS transactions without a 3270 terminal. Commands for the 3270 terminal are intercepted by CICS and replaced by a messaging mechanism that provides a bridge between the end-user and the CICS transaction.

With the bridge feature, a **client** application that may be executing outside the CICS environment can use transport mechanisms such as the Internet to access and run a CICS 3270-based **user transaction**.

The client application can also be a CICS transaction, using, for example, a temporary storage queue to pass 3270 requests and data to a user transaction executing in the same CICS region. This provides an similar function to the FEPI interface.

The user transaction can be an existing 3270 or BMS-based CICS transaction. It runs unchanged as if it were being driven by a real terminal.

The following diagram shows the flow of a client request in a bridge environment. The client application requests execution of 3270 transactions by sending a **message** to a **bridge monitor** transaction , over a **transport mechanism** (which can be any method supported by CICS, including the Web, and CICS transient data or temporary storage queues). “Components of the 3270 bridge” on page 20 describes all the components of the bridge environment shown in this diagram.

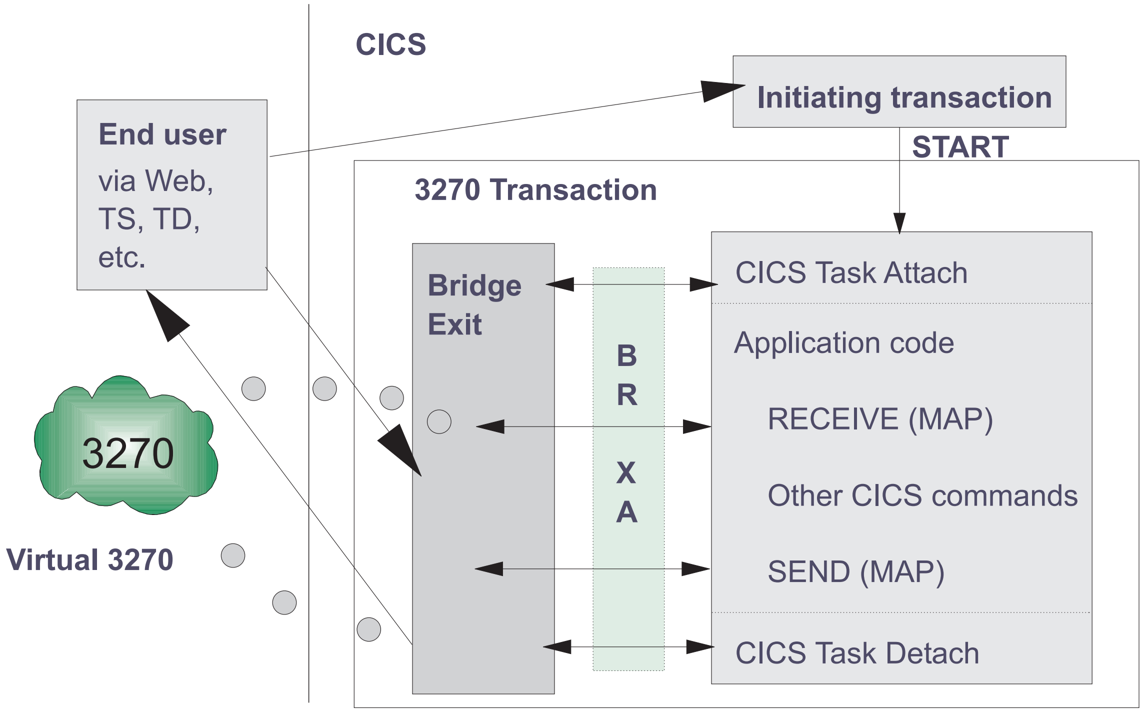


Figure 7. The 3270 bridge environment

Components of the 3270 bridge

Before you plan to use the 3270 bridge, you need to understand the following terms:

- The bridge mechanism
- The user transaction
- The client application
- The transport mechanism
- Bridge messages
- The bridge monitor
- The bridge environment
- The bridge exit
- The bridge exit area
- The bridge facility
- The FACILITYLIKE definition

The CICS 3270 bridge mechanism

The CICS 3270 bridge mechanism is the CICS function that allows a user transaction to be run without a VTAM 3270 terminal. Terminal input/output commands are intercepted by a ***bridge exit*** that emulates the terminal by passing the commands, packaged as ***messages*** to the end-user or ***client application***.

The user transaction

A user transaction is a 3270 CICS transaction.

The client application

A client application is a program, usually executing outside CICS, and possibly outside VSE/ESA, that uses the CICS 3270 bridge mechanism to run a user transaction.

The transport mechanism

A transport mechanism is used by the client application to pass ***messages*** to CICS. The Web; and CICS temporary storage and transient data, are all examples of transport mechanisms. Some transport mechanisms have separately definable queues.

Messages

A message contains information that provides all or part of the data needed to run a 3270 user transaction. Data originally written to the 3270 screen by the user transaction is packaged into messages and sent to the client application. Data originally read from the 3270 screen by the user transaction is obtained from messages sent by the client application.

The bridge monitor

A bridge monitor is usually a long-running CICS task that is associated with a specific transport mechanism, or a specific queue in a transport mechanism. A bridge monitor is not required if the Client application is a CICS task running in the same partition using TS or TD queues as the Client application can issue the START BREXIT command. When a message arrives requesting a CICS user transaction, the bridge monitor issues a START BREXIT TRANSID command to start the requested user transaction in a ***bridge environment*** where it will execute in association with the specified ***bridge exit***.

The bridge monitor must ensure that the requested transaction is started, and started only once. It can receive confirmation messages from the bridge exit after a successful start.

A bridge monitor can be is designed as a long running CICS task, to start any user transaction, or it can be designed to start only a single transaction.

The bridge environment

The bridge environment is established by CICS so that the ***user transaction*** can be executed and certain commands intercepted. A ***bridge exit*** and a ***bridge facility*** are essential components of the bridge environment.

The bridge exit

A bridge exit is a user-replaceable program that emulates the 3270 terminal API.

It does this by packaging data originally intended for the 3270 screen into messages and passing them to the transport mechanism, for delivery to the client application. Response messages from the client application are returned to the user transaction to satisfy terminal requests.

A bridge exit is usually designed to work with a specific transport mechanism.

A bridge exit can be generic if it is designed to run any user transaction, or specific if it is designed to work with a single user transaction.

In CICS Transaction Server for VSE/ESA Release 1.1.1, all 3270 terminal API requests are passed to the bridge exit. This provides a simple interface for specific bridge exits, but is very complicated in the generic case.

To reduce the complexity, the bridge exit can be designed to handle some of the requests, with all the terminal API requests being passed to another user-replaceable program, known as a **formatter**.

If used, the name of the formatter is obtained from the BRXA_FORMATTER field in the BRXA and the bridge exit is only called for requests that require input or output of data. If a formatter is not specified, the bridge exit is called for all requests.

The bridge exit is always called for the following requests:

- User transaction initialization
- User transaction bind
- User transaction termination
- User transaction abnormal termination
- Read and Write message
- Syncpoint (optional)

A formatter is called for the following requests:

- SEND (Terminal Control and BMS)
- RECEIVE (Terminal Control and BMS)
- CONVERSE
- FREE
- ISSUE DISCONNECT
- ISSUE ERASEAUP
- RETRIEVE (in some cases)

All requests made in a bridge exit are run as part of the same unit of work as the user transaction. Therefore, any recoverable requests made by the bridge exit are committed or rolled back at the same time as the user transaction resources.

The abend termination handler in the bridge exit is called when the user transaction terminates abnormally, so that the client application can be informed of the abend. The bridge exit can issue only non-recoverable requests in this call.

The interface between the bridge exit (the BRXA) and CICS is described fully in “Bridge exit area (BRXA)” on page 84. This interface is defined by CICS and must be used by all bridge exits.

The messaging interface between the bridge exit and the remote resource or the end-user is not formally defined. You may define this interface to suit your own environment. A sample interface has been defined that is used by the CICS TS/TD supplied exits. This interface is described in “Message data format” on page 59. You can use this interface definition as a basis for your own implementation using other transport mechanisms.

Formatter

A formatter is a user-replaceable program that can (optionally) be used in association with a bridge exit. The formatter performs all the message building and analysis functions otherwise done in the bridge exit. Separation of this function simplifies the logic of the bridge exit and allows the formatting code to be used by many different bridge exits.

The bridge exit area

The bridge exit area (BRXA) is the communication area between the bridge exit and CICS. It is a CICS COMMAREA (subject to normal length constraints) containing a number of sub-areas that are used by the bridge exit to process each call, and retain information between calls.

It contains the following subareas:

Header

This area contains version information and pointers to some of the following areas.

Transaction area

This area is used by bridge exit initialization processing. It contains information about the user transaction that CICS runs, and the real 3270 that it expects to use.

Command area

This area provides details of the command request. For CICS API requests it provides a simplified description of the command and response fields.

User area

This area is used to store data between calls to the bridge exit. It acts as a user input area to store the messages needed to satisfy RECEIVE and RETRIEVE requests, and also as a user output area to store the messages from SEND requests so that they can all be sent together when the user transaction terminates.

ADS descriptor

This area contains the ADS descriptor for BMS SEND MAP and RECEIVE MAP requests.

If the user application issues a BMS command, the bridge exit is called and passed (in the user area) the BMS Application Data Structure (ADS). This 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 this gives the client application a simplified interface to the terminal data, without the need to understand 3270 data streams.

The ADS descriptor allows the exit to interpret the BMS Application Data Structure (ADS), without requiring that the copybook for the Application Data Structure (or DSECT) be included in the source for the exit program at compile time.

The ADS descriptor is generated as part of the mapset load module produced by the map definition macros, provided these are assembled using the CICS Transaction Server for VSE/ESA Release 1.1.1 or later macro library. Mapsets generated with previous CICS releases do not contain the ADS descriptor. If the mapset does not contain the ADS descriptor, a null pointer is set in the bridge exit area.

The ADS descriptor can be generated in short or long form, by specifying the DSECT option on the DFHMSD macro. The long form, ADSL, contains all fields aligned on 4-byte boundaries. This is required if the transport mechanism that you are using is MQSeries, or cross-platform.

Note: If you are unable to reassemble the mapset because you do not have the source, you can use the DFHBMSUP utility provided by CICS Transaction Server for VSE/ESA Release 1 to recreate source statements from your mapset load module. See the *CICS Operations and Utilities Guide* for information about DFHBMSUP.

During initialization, the bridge exit can set an indicator in the bridge exit area to control whether or not the ADS descriptor should be passed to the exit on BMS SEND MAP and RECEIVE MAP calls. (If the ADS descriptor is required, the CICS interface code has to load the mapset and locate the descriptor, thus increasing the path-length.)

The format of the BRXA is defined in “Bridge exit area (BRXA)” on page 84.

The bridge facility

The bridge facility is a virtual terminal, replacing a real 3270, which is visible only to the user transaction and does not appear in response to CEMT I TERM or CEMT I TASK.

It has a dynamically created TERMID that can be used, for example, as the basis of a unique name for a TD or TS queue.

The bridge facility emulates a real terminal in the following EXEC CICS interfaces:

- ASSIGN
- Terminal control and BMS API
- EIB
- INQUIRE TASK
- INQUIRE TERMINAL

Note: EXEC CICS INQUIRE TERMINAL and INQUIRE TASK return information about a bridge facility only if issued from within the user transaction. The bridge facility is not visible outside the user transaction.

The bridge facility is discarded at the end of the user transaction when the bridge exit (during termination processing) sets the **keep time** to zero. The keep time defines how long the bridge facility should be retained after the transaction terminates. The bridge exit must specify a non-zero value if the bridge facility is to be kept for the next part of a pseudoconversation.

The bridge facility can have a TCTUA (Terminal Control Table User Area), which can be accessed by EXEC CICS ADDRESS TCTUA in the normal way. The TCTUA is initialized to nulls when the bridge facility is created.

A global user exit (GLUE) called XFAINTU is called when a bridge facility is created and discarded. XFAINTU is passed the address of the TCTUA, so you can use this exit to initialize the TCTUA.

The characteristics of a bridge facility are copied from a FACILITYLIKE definition, with the addition of preset security.

The FACILITYLIKE definition

FACILITYLIKE is the name of a real terminal 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):

- The bridge exit can return the FACILITYLIKE name in the BRXA_FACILITYLIKE parameter of the bridge exit initialization call.
- The parameter can be obtained from the PROFILE 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 a pseudoconversation, CICS does not search for a new FACILITYLIKE value.

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.

Bridge implementations provided

The following bridge programs are provided:

DFH0CBRE
A COBOL bridge exit program that uses CICS temporary storage or transient data queues to pass messages (in MQCIH format) to the end user application (another CICS application).

DFH0CBRF
A COBOL bridge exit formatter designed to work with DFH0CBRE. This builds and interprets BRMQ message vectors.

DFHWBLT
An object code bridge exit that allows you to access a CICS transaction from the World Wide Web. This exit uses the CICS Web Interface support described in the *CICS Internet Guide*.

Copybooks

The following copybooks are provided:

DFH0CBRD
COBOL copybook used by DFH0CBRE, and DFH0CBAE.

DFH0CBRU
COBOL copybook used by DFH0CBRF

DFHBRSDx
Copybooks in all supported languages defining the interface between the bridge monitor and the bridge exit.

DFHBRMHx
Copybooks in all supported languages defining the message header included in all messages passed between the supplied bridge exit and the client application. Constants are also supplied for these copybooks.

DFHBRMQx
Copybooks in all supported languages defining the command vectors in the messages passed between the supplied bridge exit and the client application. Constants are also supplied for these copybooks.

See “Data formats for the supplied bridge exits” on page 57 for more information about the formats of message headers and vectors.

Resource definitions

The following resource definition groups are provided:

DFH$BR
resource definitions to support the DFH0CBRE bridge implementation.

The CA21 SupportPak

The CA21 SupportPak 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, using the CICS 3270 Bridge facility rather than standard CICS terminal control function. You can then 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 created by the bridge exit, and log them for further analysis. You can then use this information to write your own end-user application to drive the CICS 3270 transaction instead of a real 3270 terminal.

The CA21 SupportPak can be obtained from the Web, at the following URL:

```
http://www.software.ibm.com/ts/cics/txppacs
```

Running 3270 transactions in a bridge environment

A user transaction is started directly by a bridge monitor transaction using the START BREXIT TRANSID command.

The user transaction is initialized in a bridge environment; all 3270 terminal commands are intercepted and passed to the named bridge exit that emulates the 3270 terminal by packaging the commands into messages and passing them to the transport mechanism for delivery to a client application.

A formatter may be used to package the commands into messages, to simplify the role of the bridge exit as a message handler only.

The bridge monitor transaction is normally a long running task associated with a message queue. It looks at the contents of each message to search for requests for new work, and identifies the name of the user transaction to run. It then starts the requested transaction and checks that it has started successfully.

The client application, which may be executing anywhere accessible by the transport mechanism, extracts the 3270 data from the messages and constructs reply messages, containing 3270 data and commands, to pass to the bridge exit to satisfy the 3270 terminal commands.

Simple terminal transactions

A simple terminal transaction is one in which all user data required to run the transaction is available before the transaction has started. There is a single input screen and one or more output screens. It may issue recoverable requests.

The following example of a simple inquiry or update transaction shows how a bridge exit is called to process various requests, and is passed a structured COMMAREA containing the full description of the request.

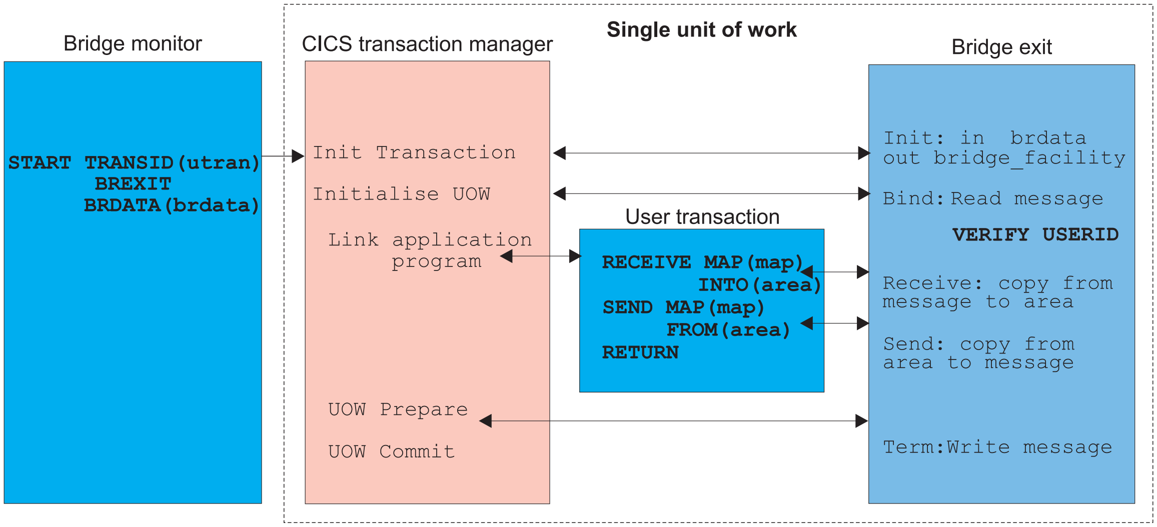


Figure 8. A simple terminal transaction

The following example shows a simple inquiry or update transaction when a formatter is used.

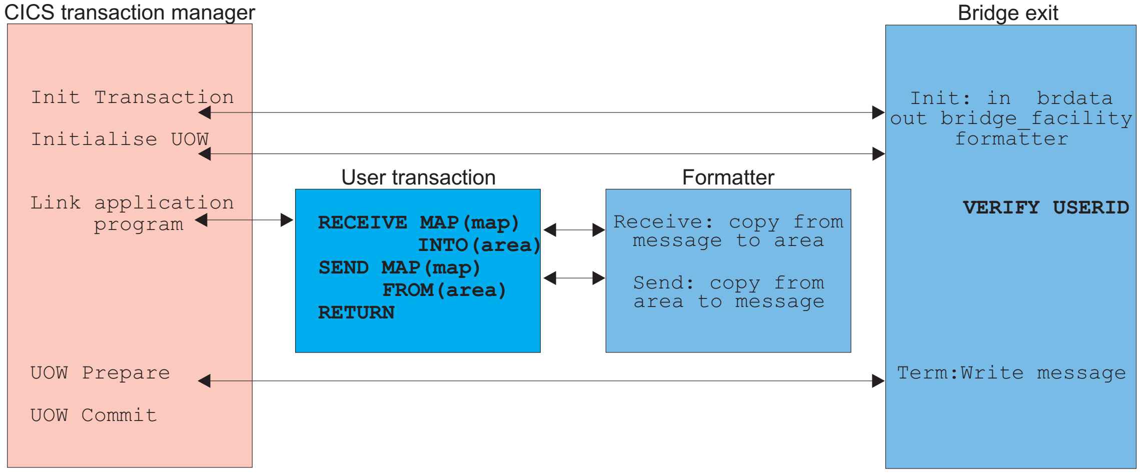


Figure 9. A simple terminal transaction using a formatter

Pseudoconversational transactions

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

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

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

Commands 1-4 all cause the bridge mechanism to pass the next transaction identifier and the START code in the bridge exit area (BRXA) on the termination and abend calls, with an indicator showing the source of the next TRANSID. This indicator can have 3 settings:

IMMEDIATE
The next TRANSID value came from a RETURN IMMEDIATE.

STARTED
The next TRANSID value came from a START TERMID.

NORMAL
The next TRANSID value came from a RETURN TRANSID or SET command.

The BRXA_STARTCODE field is also set to the start code appropriate for the next transaction.

At transaction termination, the bridge exit is called and passed the BRXA containing the next TRANSID and START code, and the indicator. It can then issue an EXEC CICS START BREXIT command for the next TRANSID, or return the next transaction information to the client application.

You can design the bridge exit to provide any of the following options for the client application, when issuing a message for a subsequent transaction:

- Copy the next TRANSID in the output message to the TRANSID field in the new input message. In this case the specified next TRANSID (from whatever source) will be run.
- Put a different TRANSID in the TRANSID field of the new input message. This provides a mechanism for cancelling the existing flow; the equivalent of logging or powering the terminal off. The existing next transaction and all the start requests are discarded. An exception trace entry is written.

To preserve the pseudoconversational environment, the bridge exit requests that its bridge facility be kept by specifying a **keep time** value. A bridge facility token is returned to the client application. Subsequent transaction requests in the pseudoconversation must contain this token.

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

Pseudoconversational and terminal information (the COMMAREA and TCTUA) are saved and associated with the bridge facility.

The TERMID is preserved for the lifetime of the bridge facility. This means that transactions that set up TS or TD queues using a TERMID as part of the name can be run in a bridge environment.

Note: If the next TRANSID is set (or defaulted) by the bridge exit to the same value as the next TRANSID saved in the bridge facility, CICS treats the transaction as part of a pseudoconversation. The user application can issue EXEC CICS INQUIRE TASK STARTCODE to find out if it is part of a pseudoconversation. The startcode ’ TO’ is returned for the first transaction and ’TP’ for subsequent transactions in the pseudoconversation.

Conversational transactions

The bridge exit can handle conversational user transactions that involve multiple terminal input screens resulting from the transaction issuing multiple RECEIVEs. These transactions can by handled by the bridge mechanism in two ways:

- The client application provides the bridge exit with all of the input necessary to satisfy all of the RECEIVE requests. In effect, this turns a conversational transaction into a non-conversational transaction. This is possible only if all of the data necessary to satisfy all input requests is known before starting the transaction.
- It is more usual for the end-user to analyze information from a previous SEND before responding to a subsequent RECEIVE. To support this the bridge exit sends a message requesting more data to the client application.

Implementing a 3270 bridge environment

The bridge mechanism is very flexible, but most implementations fall into a few basic models.

This section tells you how to identify the model that fits the requirements of your system and applications. It then presents examples of each model, which you can use as checklists when preparing your own client programs, bridge exits and monitors.

It covers the following topics:
- "Determining the bridge environment model"
- "Using the long running monitor model to implement the 3270 bridge" on page 31
- "Using the two task model to implement the 3270 bridge" on page 33
- "Using a single message model to implement the 3270 bridge" on page 35
- "Using the direct model to implement the 3270 bridge" on page 36

Determining the bridge environment model

There are basically four models:

- Long running monitor
- Two-task model
- Single message monitor
- Direct (without a bridge monitor)

From the questions asked in the following sections, you can determine what kind of model is required to run the bridge in your implementation:

Is the client application a CICS transaction?

YES use the **Direct** model. The client program can issue the EXEC CICS START BREXIT command itself, and does not need a bridge monitor.

Can the bridge exit send and receive messages directly from the client application?

NO use the **Two-task** model.

Is a new transaction started when a message arrives in CICS?

YES use a **Single message** model.

Otherwise, a **Long-running monitor** model should be used.

Using the long running monitor model to implement the 3270 bridge

In this model, the client can be anywhere, but is usually outside CICS, possibly on a workstation platform. The client application sends a message to CICS, and waits for a response.

The bridge monitor:

- Identifies that there is a new message (It could be POSTed, or browse a queue)
- Browses the message to obtain the user TRANSID and FACILITYTOKEN, if present
- Issues an EXEC CICS START TRANSID() BREXIT() BRDATA()

This runs the user transaction. The bridge exit is called to read and write messages directly from/to the client application. The bridge monitor does not (usually) need any further involvement in this request, but remains waiting for new requests.

Note: The bridge monitor must not get a lock on the message as the message must be readable by the bridge exit. The messages can be recoverable.

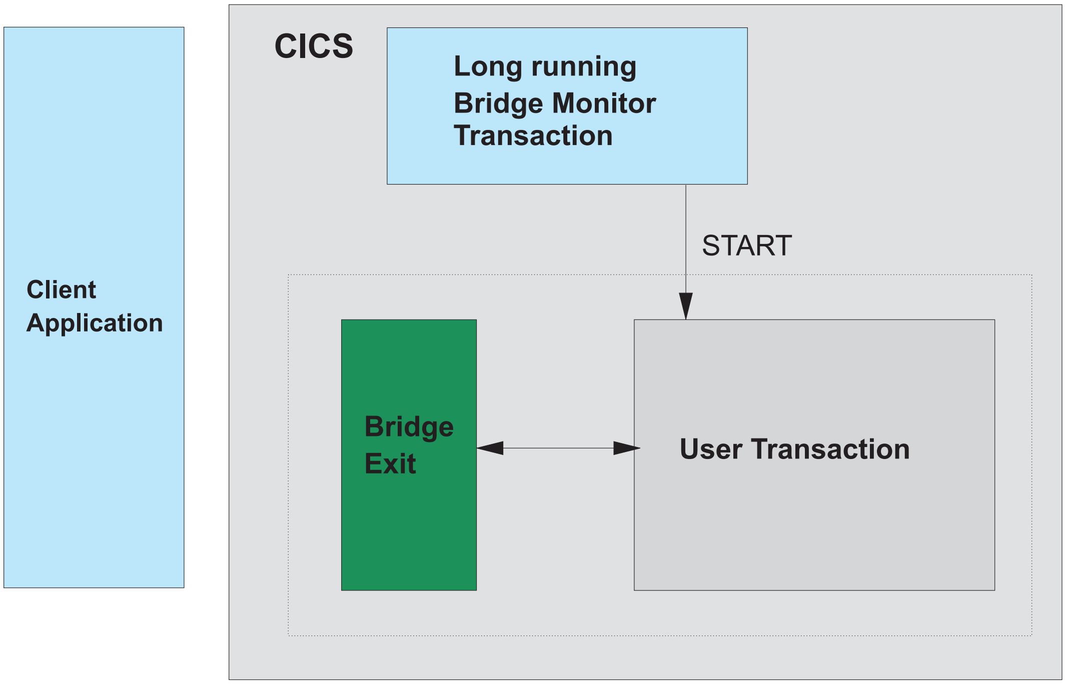


Figure 10. The long running monitor model

Client application design

1. Set initial FACILITYTOKEN to nulls
2. Create a message header containing:
   - TRANSID
   - FACILITYTOKEN
   - FACILITYLIKE
   - USERID (optional)
   - output message and/or queue identifier
3. Loop until end of transaction and NEXTTRANSID is blank:
   - Create message vectors containing information to satisfy each expected CICS API request (this is usually just a RECEIVE or RETRIEVE vector)

- Create a message containing the message header plus zero or more vectors
- Output message
- Input message reply (either with a wait option, or a loop)
- Copy the input message header to the next output message
- Extract message vectors from the input message
- Process these and get futher input from the user if necessary
- If next transid is set then copy next transid to TRANSID

4. End of loop

Bridge monitor design

1. Get startup information (for example, queue identifier)
2. Initialize queues
3. Loop until shutdown request
   - Wait for new message
   - Browse new message header (without locking it)
   - Extract following from the message header
     - TRANSID
     - FACILITYTOKEN
     - FACILITYLIKE (optional)
     - USERID (optional)
     - output message and/or queue identifier
   - Create brdata containing (see DFHBRSD for example)
     - input message and/or queue identifier
     - output message and/or queue identifier
     - FACILITYTOKEN or nulls if new request
     - FACILITYLIKE if new request (default to blanks)
   - EXEC CICS START TRANSID(transid) BREXIT(bridge exit) BRDATA(brdata)
   - Check if task shutdown
4. End of loop
5. Shutdown process

Bridge exit design

The supplieds DFH0CBRE and DFH0CBRF can be used with the following changes (if necessary):

- Change the transport mechanism specific calls
- If a message header other than MQCIH is used change this in DFH0CBRF
- If message vectors other than the BRMQ vectors are used, change these in DFH0CBRF

Using the two task model to implement the 3270 bridge

In this model, the client can be anywhere, but is usually outside CICS, possibly on a workstation platform The bridge monitor is started by a message arriving in CICS. The bridge monitor then:

- Browses the message to obtain the user TRANSID and FACILITYTOKEN, if present
- Issues an EXEC CICS START TRANSID() BREXIT() BRDATA()

The bridge exit cannot read or write the messages directly using the transport mechanism, so the bridge monitor and bridge exit communicate with each other using ECBs. When the bridge exit wants to write a message, it stores the message in memory, or on a non recoverable TS queue, and posts the bridge monitor. The bridge monitor then gets the message and sends it to the client. This model is the most complex, and the messages are not recoverable.

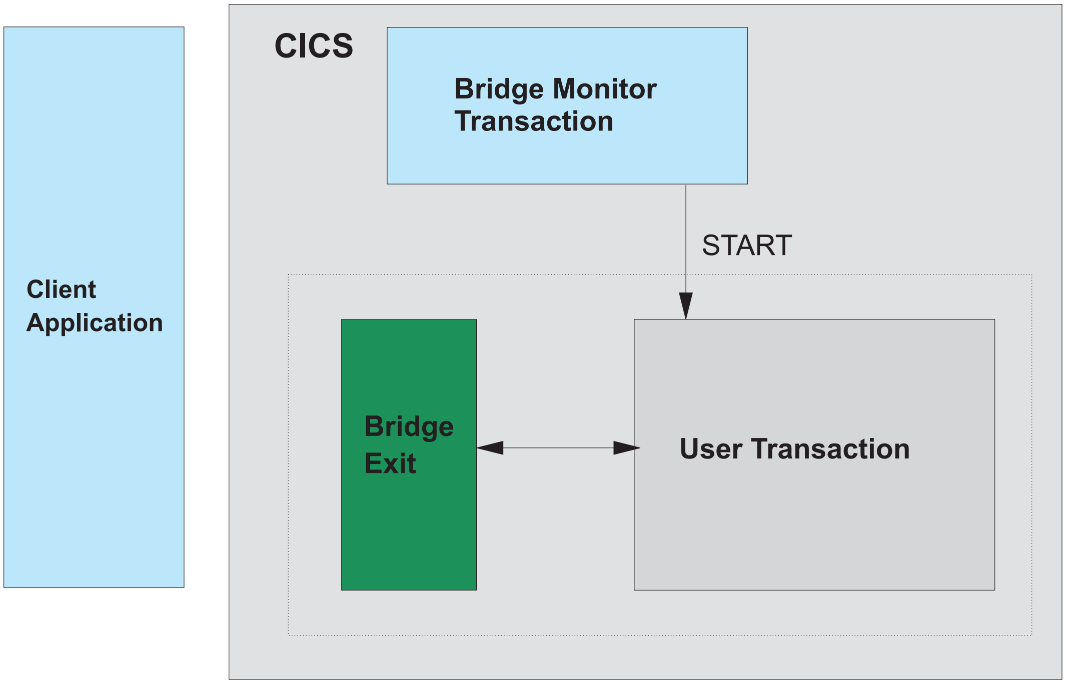


Figure 11. The two-task model

Bridge monitor design

1. Get startup instance information (for example, message and queue identifier)
2. Get shared storage for message buffers Initialise queues
3. Read new message header (if possible without locking it)
4. Extract following from the message header
   - TRANSID
   - FACILITYTOKEN
   - FACILITYLIKE (optional)
   - USERID (optional)
   - output message and/or queue identifier
5. Create brdata containing (see DFHBRSD for example)
   - input message buffer address
   - output message buffer address
   - FACILITYTOKEN or nulls if new request
   - FACILITYLIKE if new request (default to blanks)
   - ecb address

   - message buffer address
6. EXEC CICS START TRANSID(transid) BREXIT(bridge exit) BRDATA(brdata)
7. Loop until message indicates end of transaction
   - Wait on ecb
   - Get message from bridge exit
   - Write message
   - If message is a request for more data:
     – Read next message (wait for response from client)
     – Send message to bridge exit
     – Post ecb

Bridge exit design

The supplieds DFH0CBRE and DFH0CBRF can be used with the following changes (if necessary):

- Change the transport mechanism specific calls to use the post/wait mechanism
- If a message header other than MQCIH is used change this in DFH0CBRF
- If message vectors other than the BRMQ vectors are used, change these in DFH0CBRF

Using a single message model to implement the 3270 bridge

In this model, the client can be anywhere, but is usually outside CICS, possibly on a workstation platform The bridge monitor is started by a message arriving in CICS. The bridge monitor may read the message recoverably, but in this case should issue a syncpoint before issuing the START request. The bridge monitor then:

- Browses the message to obtain the user TRANSID and FACILITYTOKEN, if present
- Issues an EXEC CICS START TRANSID() BREXIT() BRDATA()

After the monitor has issued the START it has no further involvement, so can terminate.

When the bridge exit is called to read and write messages, it communicates directly with the client application. The messages can be recoverable.

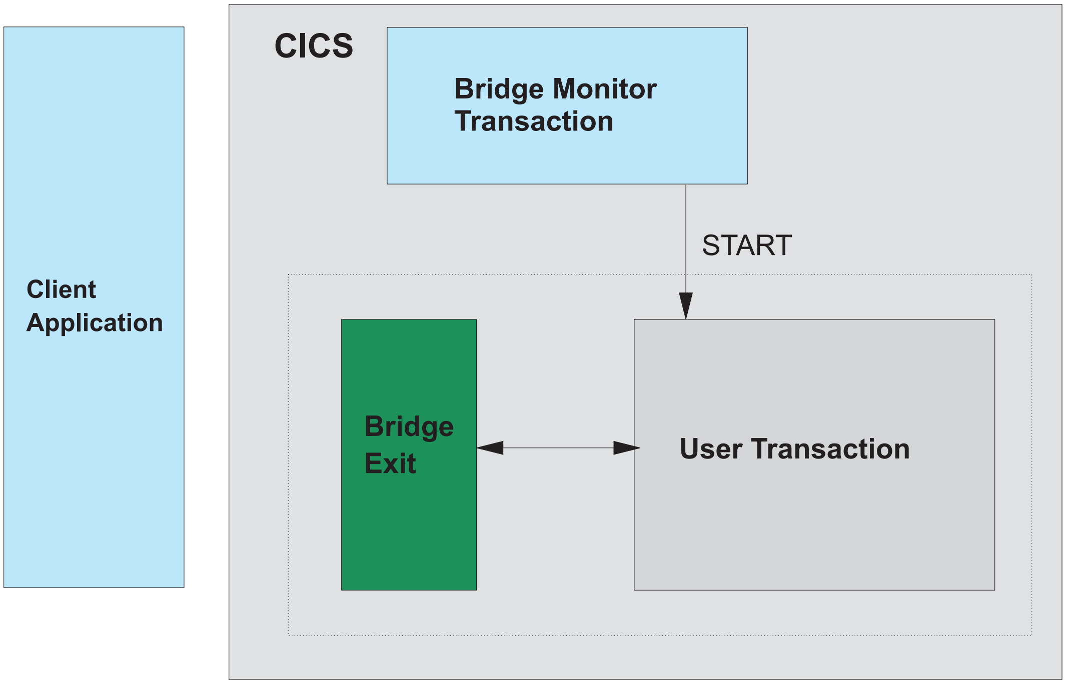


Figure 12. The single message model

Bridge monitor design

1. Get start-up instance information (for example, message and queue identifier)
2. Initialize queues
3. Read new message header (if possible without locking it)
4. Extract the following from the message header:
   - TRANSID
   - FACILITYTOKEN
   - FACILITYLIKE (optional)
   - USERID (optional)
   - output message and/or queue identifier
5. Create brdata containing (see DFHBRSD for example):
   - input message and/or queue identifier
   - output message and/or queue identifier
   - FACILITYTOKEN or nulls if new request
   - FACILITYLIKE if new request (default to blanks)
6. EXEC CICS START TRANSID(transid) BREXIT(bridge exit) BRDATA(brdata)

Using the direct model to implement the 3270 bridge

In this model, a CICS application directly starts a user transaction running in the bridge environment. This model has been known in the past as the 'user bridge'.

The client application, which is a CICS transaction:

- Writes a message on a queue (probably a TS queue)
- Creates brdata (probably consisting of the TS queue name and a null FACILITYTOKEN)
- Issues an EXEC CICS START TRANSID() BREXIT() BRDATA()

The client application then waits for data on the output queue. When the bridge exit is called to read and write messages, it communicates directly with the client application.

If Temporary Storage (TS) queues are used for the transport mechanism, they must not be recoverable as they are used by both the client and the bridge exit. If the same exit is used, communication between the client and bridge exit is synchronized by repeated EXEC CICS DELAY commands to wait for messages on the TS queue. If this is not efficient enough, this can be changed to an ECB mechanism.

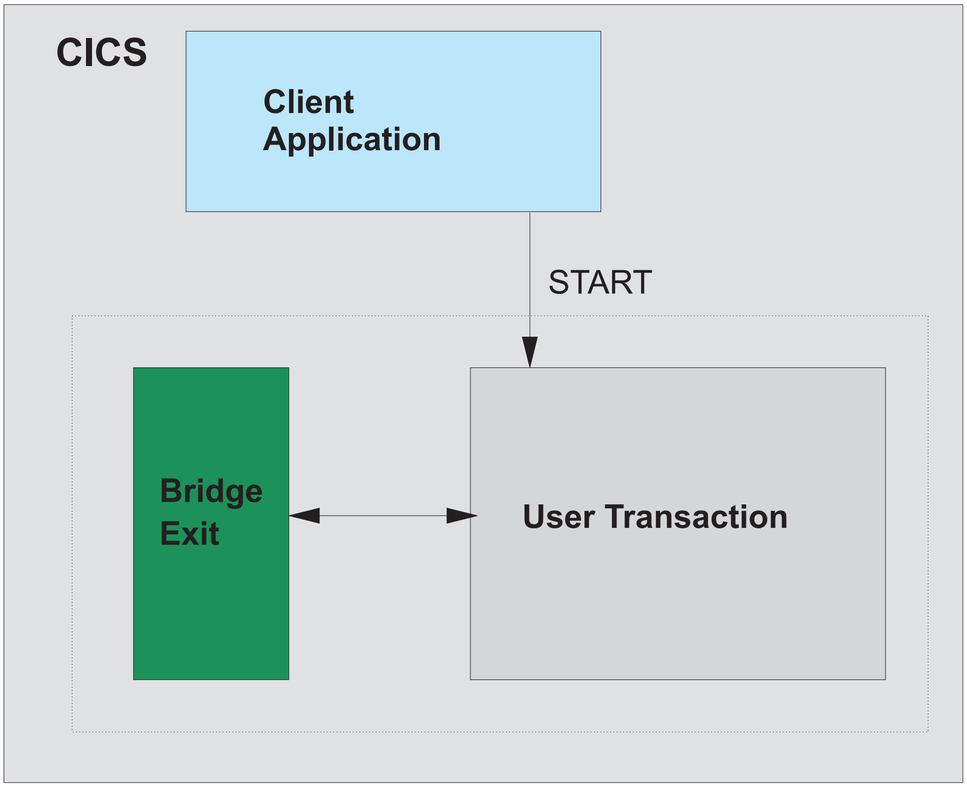


Figure 13. The Direct model

Client Application (CICS transaction) design

1. Create new message
   - First transid in pseudo conversation
   - Null FACILITYTOKEN
   - FACILITYLIKE or blank

2. Loop until end of pseudo conversation:
    - Write message to TS queue
    - Create brdata (using DFHBRSD) containing:
        - Input and output queue names
        - FACILITYLIKE and FACILITYTOKEN from message
    - EXEC CICS START TRANSID(from message) BREXIT(bridge exit) BRDATA(dfhbrsd)
    - Loop until end of transaction:
        - Loop until message obtained:
            - Read message from TS queue
            - Wait a second
        - Process message (such as send response to client)
        - If request for more data
            - Create next message
            - Write message to TS queue
    - If there is a next TRANSID
        - Copy next TRANSID to TRANSID in message

Bridge exit design

DFH0CBRE and DFH0CBRF can be used unchanged.

Chapter 4. The Bridge environment

The bridge environment is established when CICS receives a START BREXIT TRANSID call. The transaction specified by TRANSID is associated with the bridge exit specified by BREXIT (or specified on the TRANSACTION resource definition), and a virtual 3270 terminal, the bridge facility, is created.

The transaction executes in the usual way, but all terminal commands are intercepted and passed to the bridge exit.

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.

This chapter describes these special characteristics of the user transaction. It covers the following topics:

- "User transaction programming considerations"
- "Defining the user transaction" on page 44
- "Inquiring about the bridge environment" on page 45
- "The bridge facility" on page 49

User transaction programming considerations

The user transaction runs unchanged, with the following limitations.

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 TERMID. The name is not visible outside the user transaction, and may contain characters that are not allowed in a VTAM netname definition.

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, and BRXA_LOAD_ADS_DESCRIPTOR is set to Y in the bridge exit area. See"Bridge exit area (BRXA)" on page 84 for details of the fields in the bridge exit area.

BMS requests

The 3270 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:

Command	**Calls bridge exit**
RECEIVE MAP TERMINAL	YES
RECEIVE MAP FROM	NO
RECEIVE MAP MAPPINGDEV	NO

Note: TERMINAL is implied if neither TERMINAL nor FROM is specified.

SEND commands:

Command	ACCUM supported	Calls bridge exit
SEND MAP TERMINAL	NO	YES
SEND TEXT TERMINAL	NO	YES
SEND TEXT NOEDIT TERMINAL	NO	YES
SEND TEXT MAPPED TERMINAL	NO	YES
SEND CONTROL TERMINAL	NO	YES
SEND MAP SET	YES	NO
SEND TEXT SET	YES	NO
SEND TEXT NOEDIT SET	YES	NO
SEND TEXT MAPPED SET	YES	NO
SEND CONTROL SET	YES	NO
SEND MAP MAPPINGDEV	NO	NO

Note: 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
The RCT allows AUTHTYPE=TERMID or OPID which means that security

checking is done against the corresponding name. This fails in a bridge environment, and AUTHTYPE=USERID must be used instead. This is the preferred method in all environments.

External security customization
TERMID/OPID/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 (TCAM).

XTCIN after an input event (TCAM).

XTCOUT
before an output event (TCAM).

XZCATT
before a task attach (VTAM).

XZCIN after an input event (VTAM).

XZCOUT
before an output event (VTAM).

XZCOUT1
before a message is broken into RUs (VTAM).

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

Monitoring
A 3270 bridge transaction identifier has been added to 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. The USERID is the value returned on the bridge exit initialization call. If no value is returned, the USERID with which the transaction was started is used. 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 if the user transaction ends and the bridge exit specifies a keep time value.

When the bridge exit initialization call returns a bridge facility token, a check is made that the USERID is the same as that specified when the bridge facility was created. No other security check is made. If a greater level of security is required, such as validation of every message, this can be done in the bridge exit by issuing a VERIFY PASSWORD command.

START

The user transaction can issue EXEC CICS START requests for its own bridge facility. This allows existing menu-driven and pseudo-conversational applications that use this interface to work in a bridge environment. See "Pseudoconversational transactions" on page 28 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. If any of these options are specified, they are saved with the other START data and passed in the BRXA to the TERM call of the bridge exit. It is then the responsibility of the bridge exit to take account of them. They could also be passed back to the end user application and used there in some way.

The INTERVAL and AFTER values can be used 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 issued to the user transaction's own terminal are not fully supported because they are not required in the bridge environment:

TERMID

You can only specify the name of the bridge facility for this transaction. Any other name will result in a TERMIDERR.

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.

ATTACH

This option applies only to non-terminal starts. The CICS translator rejects START ATTACH requests when TERMID is specified.

BREXIT

The BREXIT option applies only to non-terminal starts. The CICS translator rejects START BREXIT requests when TERMID is specified.

STARTed transactions

Some menu applications use START to initiate subsequent transactions.

Started transactions are identified by specifying the appropriate value in BRXA_STARTCODE the bridge exit initialization call. This value is used 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. If there is no data, CICS does not immediately return ENDDATA, but calls the bridge exit, which can then provide data to satisfy the request, or return the ENDDATA condition.

Storage violation counts

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

SYNCPOINT

If the user transaction issues an explicit SYNCPOINT, or an implicit SYNCPOINT occurs, as in CREATE, DISCARD CONNECTION, DISCARD TDQUEUE, DISCARD TERMINAL, or in a DLI TERM psb, the bridge exit is called (if the BRXA_CALL_FOR_SYNC parameter is set to BRXA_YES), and reissues the SYNCPOINT, or ignores it. The bridge exit is permitted to do recoverable work before and/or after the SYNCPOINT, so it can do recoverable work in either unit of work. If the bridge exit does not reissue the SYNCPOINT, the logic of the transaction could be affected.

There is no specific call to the bridge exit for the implicit syncpoint at the end of the user transaction; this is handled by the termination and/or abend calls. The exit does not need to issue a SYNCPOINT request in the termination call, and must not issue a SYNCPOINT request in the abend call.

TCTUA

The TCTUA is available after the INIT call to the bridge exit.

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 supported.

TWA

The TWA is available after the INIT call to the bridge exit.

WAIT TERMINAL

The 3270 bridge does not support the WAIT TERMINAL command. INVREQ is returned..

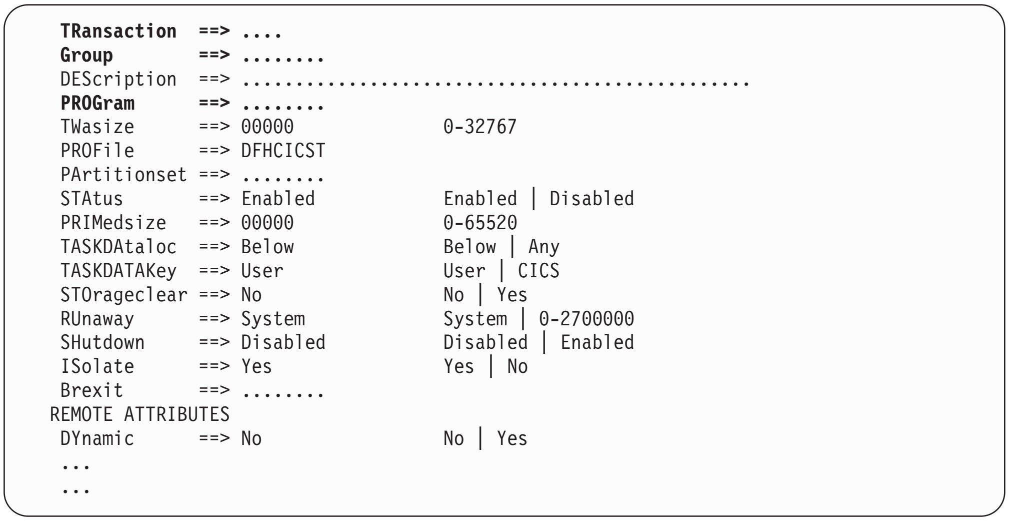
Defining the user transaction

Keywords are provided in the following resource definitions to define the default bridge exit and facility.

- TRANSACTION
- PROFILE

TRANSACTION resource definition

A user transaction definition can have an additional parameter, BREXIT, to define a default bridge exit. The named bridge exit is used when the transaction is specified in a START TRANSID BREXIT command, where the BREXIT name is blank. When the transaction is executed in the usual way, BREXIT is ignored.



```
 TRansaction  ==> ....
 Group        ==> ........
 DEScription  ==> ................................................
 PROGram      ==> ........
 TWasize      ==> 00000              0-32767
 PROFile      ==> DFHCICST
 PArtitionset ==> ........
 STAtus       ==> Enabled            Enabled | Disabled
 PRIMedsize   ==> 00000              0-65520
 TASKDAtaloc  ==> Below              Below | Any
 TASKDATAKey  ==> User               User | CICS
 STOrageclear ==> No                 No | Yes
 RUnaway      ==> System             System | 0-2700000
 SHutdown     ==> Disabled           Disabled | Enabled
 ISolate      ==> Yes                Yes | No
 Brexit       ==> ........
REMOTE ATTRIBUTES
 DYnamic      ==> No                 No | Yes
 ...
 ...
```

Figure 14. The DEFINE panel for the TRANSACTION resource definition

BREXIT

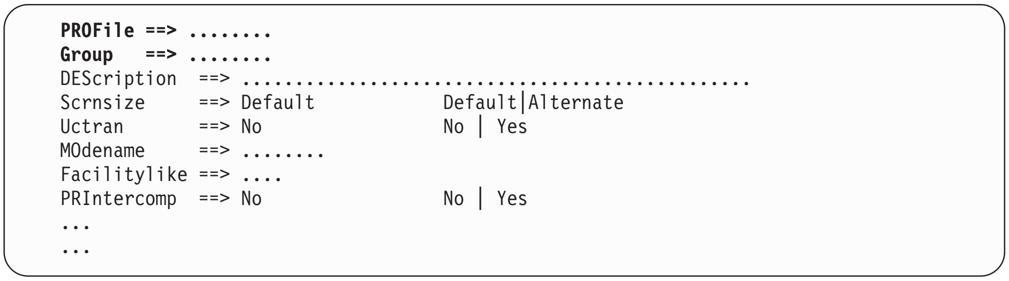
This is an optional parameter that defines the name of the default bridge exit to be associated with this transaction, if it is started in the 3270 bridge environment with a START BREXIT command, and BREXIT specifies no name.

The name may be up to 8 characters in length.

If BREXIT is defined, REMOTESYSTEM, REMOTENAME, DYNAMIC(YES), and RESTART(YES) should not be specified, and are ignored.

PROFILE resource definition

The PROFILE provides terminal-related information for a transaction, including the FACILITYLIKE parameter. The PROFILE of a user transaction can specify FACILITYLIKE to define the default terminal definition values to be used for the bridge facility.



```
PROFile ==> ........
Group   ==> ........
DEScription  ==> ................................................
Scrnsize     ==> Default            Default|Alternate
Uctran       ==> No                 No | Yes
MOdename     ==> ........
Facilitylike ==> ....
PRIntercomp  ==> No                 No | Yes
...
...
```

Figure 15. The DEFINE panel for the PROFILE resource definition

FACILITYLIKE

This is an optional parameter that specifies the name of an installed terminal resource definition to be used as a template for the bridge facility. It can be overridden by specifying FACILITYLIKE in the bridge exit.

There is no default value for this parameter, but if it is not defined here or in the bridge exit area, CICS uses CBRF.

If you are running in a CICS system started with the VTAM=NO System initialization (SIT) parameter, the resource definition specified by FACILITYLIKE must be defined as a remote terminal.

Inquiring about the bridge environment

You can use the following commands and interfaces to determine whether a transaction or task is executing in a bridge environment, and if so, to obtain information about the bridge monitor transaction that issued a START TRANSID BREXIT command to start the user transaction and its associated exit:

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

ASSIGN command

Function

Request values from outside the application program local environment.

Syntax

ASSIGN

```
►►──ASSIGN──┬───────────────────────────────┬──────────────────────────►◄
            └─BRIDGE(4-character data-area)─┘
```

Figure 16. ASSIGN (extract)

Options

BRIDGE

This parameter returns the transaction name (TRANSID) of the bridge monitor transaction that initiated the user transaction issuing this request.

A value of blanks is returned if :

- The user transaction was not started by a bridge monitor transaction.
- This command was issued by a program started by a distributed program link (DPL) request.

Conditions

Unchanged.

INQUIRE TASK command

Function

The INQUIRE TASK command returns information about a given task.

Syntax

INQUIRE TASK

```
                           ┌───────────────────────────┐
                           ▼                           │
►►──INQUIRE TASK(data-value)───┬──────────────────────┬┴──────────────►◄
                               ├─BRIDGE(data-area)────┤
                               └─IDENTIFIER(data-area)┘
```

Figure 17. INQUIRE TASK (extract)

Options

BRIDGE*(data-area)*

returns the 4-character name of the bridge monitor transaction that issued a START BREXIT TRANSID command to start this task. If this task is not currently running in the 3270 bridge environment, blanks are returned.

IDENTIFIER*(data-area)*

returns a 48-character field containing user data provided by the bridge exit, if the task was initiated in the bridge environment, or blanks, otherwise. This field is intended to assist in online problem resolution.

For example, it could contain the MQ correlator for the MQ bridge, or a Web token.

Conditions

Unchanged.

INQUIRE TRANSACTION command

Function

The INQUIRE TRANSACTION command returns information about a named transaction.

Syntax

INQUIRE TRANSACTION

```
                                      ┌──────────────────────────┐
                                      ▼                          │
►►──INQUIRE TRANSACTION(data-value)───┬────────────────────────┬─┴───────►◄
                                      ├─BREXIT(data-area)──────┤
                                      └─FACILITYLIKE(data-area)┘
```

Figure 18. INQUIRE TRANSACTION (extract)

Options

BREXIT*(data-area)*
returns the 8-character name of the bridge exit defined by the BREXIT parameter of the named transaction resource definition.

If BREXIT is not defined, blanks are returned.

FACILITYLIKE*(data-area)*
returns 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.

Conditions

Unchanged

CEMT INQUIRE TASK

Function

CEMT INQUIRE TASK returns information about a given task.

Syntax

CEMT INQUIRE TASK

```
►►──CEMT Inquire TAsk──┬───────────────┬──┬───────────────────┬──────────►◄
                       └─BRidge(value)─┘  └─IDentifier(value)─┘
```

Figure 19. CEMT INQUIRE TASK (extract)

Options

BRidge*(value)*
returns the 4-character name of the bridge monitor transaction that issued a START BREXIT TRANSID command to start this task. Otherwise, blanks are returned.

IDentifier*(value)*
returns a 48-character field containing user data provided by the bridge exit, if the task was initiated in the bridge environment, or blanks, otherwise. This field is intended to assist in online problem resolution.

For example, it could contain the MQ correlator for the MQ bridge, or a Web token.

This field can contain hexadecimal values. The *CICS Supplied Transactions* manual tells you how to display these fields and provides more information about CEMT.

CEMT INQUIRE TRANSACTION

Function

CEMT INQUIRE TRANSACTION returns information about a named transaction.

Syntax

CEMT INQUIRE TRANSACTION

```
►►──CEMT Inquire TRAnsaction──┬─────────────┬──┬───────────────────┬──────────►◄
                              └─BRexit(value)─┘  └─FAcilitylike(value)─┘
```

Figure 20. CEMT INQUIRE TRANSACTION (extract)

Options

BRexit*(value)*
returns the 8-character name of the bridge exit defined by the BREXIT parameter of the named transaction resource definition.

If BREXIT is not defined, blanks are returned.

FAcilitylike*(value)*
returns 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.

XPI commands

INQUIRE_TRANDEF

The parameter BREXIT is provided on the INQUIRE_TRANDEF function, returning the following value:

BREXIT(name8)
returns the name of the bridge exit program. If there is no bridge exit, blanks are returned.

name8
The name of an 8-byte location to receive the name of the bridge exit program.

INQUIRE_CONTEXT

A new function, INQUIRE_CONTEXT has been created, returning the following values:

BRIDGE_EXIT_PROGRAM(name8)

returns the name of the bridge exit program used by this task. If CONTEXT returns NORMAL, the contents of this field are meaningless.

name8

The name of an 8-byte location to receive the name of the bridge exit program.

BRIDGE_FACILITY_TOKEN(name4)

returns a token that contains the address of the bridge facility used by this task. This has the same format as a TCTTE and can be mapped using the DSECT DFHTCTTE. If CONTEXT returns NORMAL, the contents of this field are meaningless.

name4

The name of a 4-byte location to receive the token.

BRIDGE_TRANSACTION_ID(name4)

returns the name of the bridge monitor transaction used to start this user transaction. If CONTEXT returns NORMAL, the contents of this field are meaningless.

name4

The name of a 4-byte location to receive the name of the bridge transaction.

BRXA_TOKEN(name4)

returns a token that contains the address of the bridge exit area (BRXA) used by this task. The format of BRXA is defined by the DFHBRARx copy book. If CONTEXT returns NORMAL, the contents of this field are meaningless.

name4

The name of a 4-byte location to receive the token.

CONTEXT(byte1)

returns, in a 1-byte location (*byte1*), the type of environment in which the transaction is running.

NORMAL

A transaction that is not running in a bridge environment.

BRIDGE

A user transaction that was started using a bridge.

BREXIT

A bridge exit program.

See the *CICS Customization Guide* for more information about the XPI.

The bridge facility

The user transaction can retrieve information about the bridge facility using INQUIRE and ASSIGN.

A user transaction can issue INQUIRE TERMINAL or INQUIRE NETNAME for its bridge facility, or can issue INQUIRE TASK for itself. The TERMID can be obtained from EIBTRMID or from ASSIGN FACILITY, and the NETNAME can be obtained from ASSIGN NETNAME. Any other task issuing these commands for the bridge transaction facility receives TERMIDERR.

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 2. INQUIRE TERMINAL values

Keyword	Returned value
ACQSTATUS	ACQUIRED
ACCESSMETHOD	VTAM
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

Table 3. INQUIRE TASK values

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

QUERY

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

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

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

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

SET TERMINAL/NETNAME

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

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

Bridge facility global user exit

When enabled, XFAINTU (FAcility INitialization and Tidy Up) is called when a bridge facility is created or deleted:

- Just after a new bridge facility has been built.

- Just before a bridge facility is deleted. This may be at the end of a task when zero keep time is specified, or when a keep time expires before the facility is reused.

XFAINTU is needed to carry out any auditing or initialization that is normally done at LOGON/LOGOFF or AUTOINSTALL/DELETE. This could be TCTUA setup or data collection that the 3270 user transaction relies upon.

The initialization call to XFAINTU is made before the CICS Recovery Manager is available. You should not invoke any services from the bridge exit that use recoverable resources. .

See the *CICS Customization Guide* for more information about global user exits.

XFAINTU

When invoked

Just after a bridge facility is created and just before it is freed.

Exit-specific parameters

UEPFAREQ

Address of a 1-byte field that indicates why the exit has been called. Possible values are:

UEPFAIN

Initialization.

UEPFATU

Tidy-up.

UEPFATUT

Address of a 1-byte field that indicates the type of tidy-up required. Possible values are:

UEPFANTU

Normal tidy-up.

UEPFAETU

Expired tidy-up.

UEPFANAM

Address of the bridge facility name.

UEPFATYP

Address of a 1-byte field that indicates the facility type. The value is always:

UEPFABR

3270 bridge facility.

UEPFAUAA

Address of the bridge facility user area (TCTUA).

UEPFAUAL

Address of a one-byte field containing the length of the bridge facility user area.

Return codes

UERCNORM

Continue processing.

XPI calls

All can be used, except those that use Recovery Manager services.

API calls

All can be used except those that invoke task-related user exits, or use Recovery Manager services.

Chapter 5. Supplied 3270 bridge exits

This section tells you about the IBM supplied bridge solutions listed in "Bridge implementations provided" on page 25. It covers the following topics:

- "The TS/TD supplied bridge exit"
- "The Web bridge exit" on page 56
- "Data formats for the supplied bridge exits" on page 57

The TS/TD supplied bridge exit

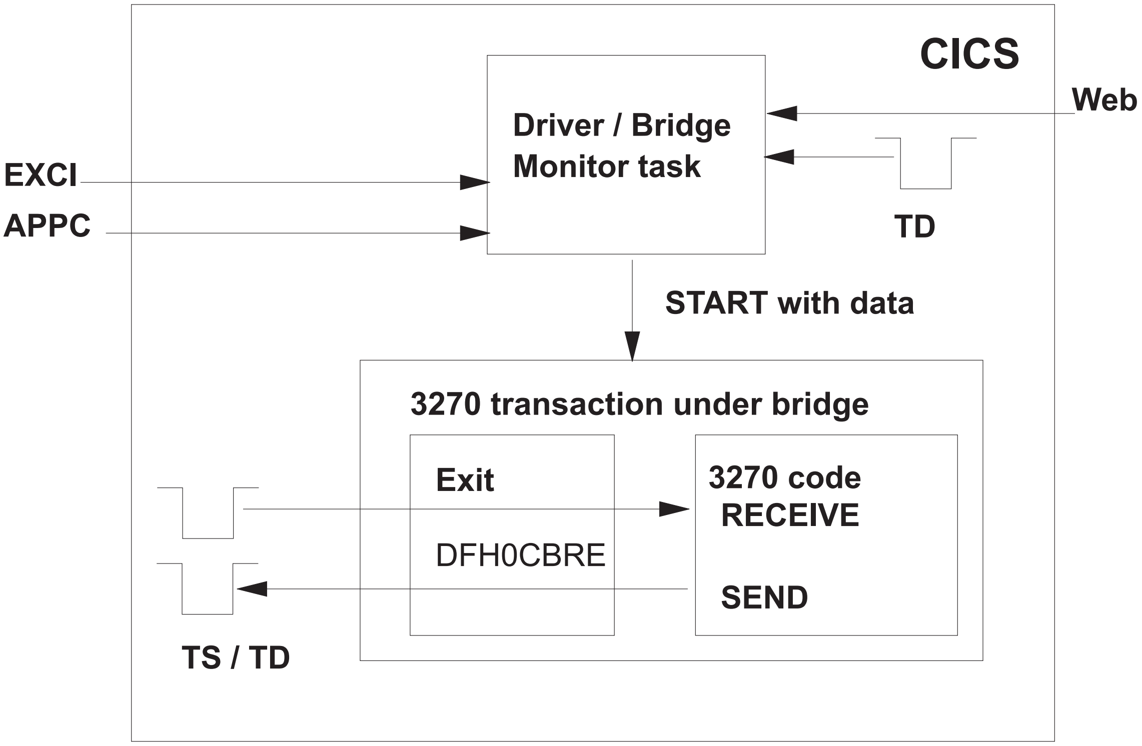


Figure 21. The supplied TS/TD bridge exit

This exit, DFH0CBRE (and its associated formatter, DFH0CBRF),is supplied in COBOL source. It uses CICS temporary storage (TS) or transient data (TD) queues to pass input and output from and to the client application (another CICS application).

You can modify this exit to support other transport mechanisms.

DFH0CBRE is the most general of the supplied exits. To run a transaction using this exit, you simply issue a START TRANSID() BRDATA() BREXIT(DFH0CBRE) command. The formats of the interfaces and messages used in this bridge exit are described in"Data formats for the supplied bridge exits" on page 57. The TS/TD supplied bridge exit is an example of the Direct model.

Using the DFH0CBRE exit

Before using the DFH0CBRE bridge exit to run an existing CICS transaction, you need to perform the following steps:

1. Translate and compile DFH0CBRE and DFH0CBRF using an appropriate COBOL compiler.
2. Link DFH0CBRE and DFH0CBRF as standard CICS application programs into a CICS load library used by the CICS system on which you will run the user transaction.
3. Define DFH0CBRE and DFH0CBRF as programs to CICS. Sample definitions are supplied in the DFH$BR group.
4. Define and install transient data queues if required.
5. Write a client requester program. This is an application program, written by you in any of the supported languages (Assembler, COBOL, PLI, C).

   It should issue a START TRANSID BREXIT BRDATA command to start the required user transaction, passing BRDATA data formatted as described in "BRDATA format" on page 58.

   It should write data to the TS/TD queue named, to provide 3270 terminal data for the 3270 user transaction, formatted as described in "Message data format" on page 59, and read responses back from the queue.
6. Define and install the end-user requester program and transaction.
7. Run the end-user requester transaction.

The Web bridge exit

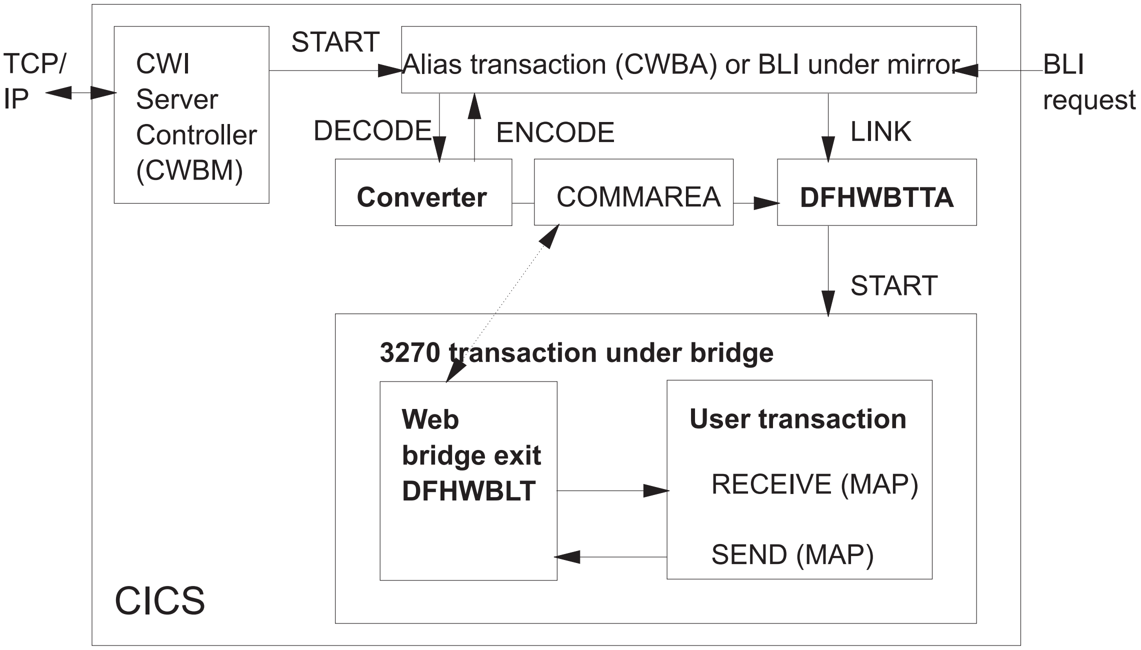


Figure 22. The Web supplied bridge exit

This exit, DFHWBLT, is an object code exit that allows you to access a CICS transaction from the World Wide Web. It uses the CICS Web support described in the *CICS Internet Guide*.

DFHWBLT works in conjunction with a long running server transaction, CWBM, which monitors the TCP/IP interface for incoming requests, and a bridge monitor, DFHWBTTA. The formats of the interfaces and messages used in this bridge exit are the same as in the TS/TD bridge and are described in “Data formats for the supplied bridge exits”. The Web bridge exit is an example of the two task monitor model.

Using the Web bridge exit

Use of this exit is fully described in the *CICS Internet Guide*. After installation, you do not need to provide any user code to use this solution, but it can be customized.

A more detailed implementation description can be found in the Redbook *CICS Transaction server for OS/390: Web Interface and 3270 Bridge*, order number SG24-5243.

Data formats for the supplied bridge exits

There are two interfaces between the supplied bridge exits and the client application. They are:

BRDATA

BRDATA is passed to the bridge exit when the user transaction and its associated bridge exit are started with a START TRANSID BREXIT BRDATA command. The main purpose of BRDATA is to pass information required for bridge exit initialization (facility token and facilitylike) and data required to identify the message queue and message. The names of the parameters and constants in the BRDATA data, translated into appropriate forms for the different programming languages supported, are defined in the following copybook files supplied as part of the 3270 bridge.

Table 4. START data copybooks

Language	Definition	Constants
Assembler	DFHBRSDD	DFHBRSCD
C	DFHBRSDH	DFHBRSCH
PL/I	DFHBRSDL	DFHBRSCL
COBOL	DFHBRSDO	DFHBRSCO

Messages

The messages passed between the bridge exit and the client application via TS and TD queues, to satisfy SEND and RECEIVE requests from the 3270 user application. The names of the parameters and constants in the message data, translated into appropriate forms for the different programming languages supported, are defined in the following copybook files supplied as part of the 3270 bridge.

Table 5. Message data copybooks

Language	Definition
Assembler	DFHBRMQD
C	DFHBRMQH
PL/I	DFHBRMQL
COBOL	DFHBRMQO

All user messages must begin with a standard message header, MQCIH. The names of the parameters and constants in MQCIH, translated into appropriate forms for the different programming languages supported, are defined in the following copybook files supplied as part of the 3270 bridge.

Table 6. MQCIH message header copybooks

Language	Definition	Constants
Assembler	DFHBRMHD	DFHBRMCD
C	DFHBRMHH	DFHBRMCH
PL/I	DFHBRMHL	DFHBRMCL
COBOL	DFHBRMHO	DFHBRMCO

BRDATA format

An EXEC CICS START BREXIT() BRDATA() command is issued to run a transaction in the bridge environment. To use the supplied bridge exits, the BRDATA data must conform to the following format.

Note that the parameter names shown are in COBOL format with a dash '-' name separator, rather than the underscore '_' required for other languages.

Offset Hex	Type	Len	Name
(0)	STRUCTURE	72	DFHBRSD
(0)	STRUCTURE	16	BRSD-HEADER-DATA
(0)	CHARACTER	4	BRSD-STRUCID
(4)	FULLWORD	4	BRSD-VERSION
(8)	FULLWORD	4	BRSD-STRUCLENGTH
(C)	CHARACTER	4	reserved
(10)	STRUCTURE	40	BRSD-QUEUE-NAMES
(10)	STRUCTURE	20	BRSD-OUTPUT-QUEUE
(10)	CHARACTER	2	BRSD-OUTPUT-TYPE
(12)	CHARACTER	2	reserved
(14)	STRUCTURE	16	BRSD-TS-OUTPUT-QUEUE
(14)	CHARACTER	4	BRSD-TD-OUTPUT-QUEUE
(18)	CHARACTER	12	reserved
(24)	STRUCTURE	20	BRSD-INPUT-QUEUE
(24)	CHARACTER	2	BRSD-INPUT-TYPE
(26)	HALFWORD	2	BRSD-INPUT-ITEM
(28)	STRUCTURE	16	BRSD-TS-INPUT-QUEUE
(28)	CHARACTER	4	BRSD-TD-INPUT-QUEUE
(2C)	CHARACTER	4	reserved
(30)	CHARACTER	8	reserved
(38)	CHARACTER	8	BRSD-FACILITY-TOKEN
(40)	CHARACTER	4	BRSD-FACILITYLIKE
(44)	CHARACTER	4	reserved

BRSD-STRUCID
An eye-catcher. This must be set to the constant brsd-struc-id in the DFHBRSCx copy book.

BRSD-VERSION
The version number of the start data. This must be set to the constant brsd-version-2 in the DFHBRSCx copy book.

BRSD-STRUCLENGTH
The length of the start data. This must be set to the constant brsd-length-2

BRSD-OUTPUT-TYPE
An indicator showing whether the output queue is temporary storage (TS) or Transient data (TD). This must be set to either the constant BRSD-TS or BRSD-TD in the DFHBRSCx copy book.

BRSD-TS-OUTPUT-QUEUE
The 8 or 16-byte name of the queue, if the output queue is a TS queue.

BRSD-TD-OUTPUT-QUEUE
The 4 byte name of the queue, if the output queue is a TD queue.

BRSD-INPUT-TYPE
The queue type, either TS or TD. This must be set to the constant BRSD-TS or BRSD in the DFHBRSCx copy book.

BRSD-INPUT-ITEM
The first item number in the TS queue. A value of binary zeroes means 'next'.

BRSD-TS-INPUT-QUEUE
The 8 or 16-byte name of the queue, if the input queue is a TS queue.

BRSD-TD-INPUT-QUEUE
The 4 byte name of the queue, if the input queue is a TD queue.

BRSD-FACILITY-TOKEN
The 8-byte bridge facility token. This value is nulls for a new facility.

BRSD-FACILITYLIKE
The name of an installed terminal that is to be used as a model for the bridge facility, when BRSD-FACILITY-TOKEN is nulls.

Message data format

Messages sent to the supplied bridge exits by the client application must all conform to the following format:

All messages must begin with a message header, MQCIH, followed by a variable number of self defining *vectors*. Each vector contains a vector header that identifies the vector type and length so that the bridge exit can interpret the content of each message.

There are three types of vector:

OUTBOUND REPLY
This type of vector flows from the user transaction to the end-user transaction carrying a reply. No further input is requested.

OUTBOUND REQUEST
This type of vector flows from the user transaction to the end-user transaction requesting further input. There is only one in each message and it must be last.

INBOUND
This type of vector flows from the end-user transaction to the user transaction carrying data to satisfy a user transaction RECEIVE or RETRIEVE. Processing is more efficient if the order of RETRIEVE and RECEIVE vectors in the first message matches the order of CICS commands in the user transaction, but this is not essential.

RETRIEVE vectors can only flow in the first message. If there are more RETRIEVEs than can fit in a single message, they can be sent in multiple messages , provided that:

- All of the RETRIEVE vectors are sent before any RECEIVE vectors
- All of the messages are written to the queue before the START BREXIT is issued.

Note: All fields have a minimum length of four bytes. Data is left justified and padded with blanks if necessary. Field names are shown with '-' (dash) separators, as used in COBOL. Other languages require '_' (underscore) separators.

Header MQCIH	Vector#1 BRMQ	...	Vector#n BRMQ

Figure 23. Message format

MQCIH Message header:

Offset Hex	Type	Len	Name
(0)	STRUCTURE	180	MQCIH
(0)	CHARACTER	4	MQCIH-STRUCID
(4)	FULLWORD	4	MQCIH-VERSION
(8)	FULLWORD	4	MQCIH-STRUCLENGTH
(C)	BINARY	8	reserved
(14)	CHARACTER	8	reserved
(1C)	BINARY	20	reserved
(30)	FULLWORD	4	MQCIH-GETWAITINTERVAL
(34)	BINARY	8	reserved
(3C)	FULLWORD	4	MQCIH-FACILITYKEEPTIME
(40)	FULLWORD	4	MQCIH-ADSDESCRIPTOR
(44)	FULLWORD	4	MQCIH-CONVERSATIONALTASK
(48)	FULLWORD	4	MQCIH-TASKENDSTATUS
(4C)	CHARACTER	8	MQCIH-FACILITY
(54)	CHARACTER	4	reserved
(58)	CHARACTER	4	MQCIH-ABENDCODE
(5C)	CHARACTER	8	MQCIH-AUTHENTICATOR
(64)	CHARACTER	24	reserved
(7C)	CHARACTER	4	MQCIH-TRANSACTIONID
(80)	CHARACTER	4	MQCIH-FACILITYLIKE
(84)	CHARACTER	4	MQCIH-ATTENTIONID
(88)	CHARACTER	4	MQCIH-STARTCODE
(8C)	CHARACTER	4	MQCIH-CANCELCODE
(90)	CHARACTER	4	MQCIH-NEXTTRANSACTIONID
(94)	CHARACTER	16	reserved
(A4)	FULLWORD	4	MQCIH-CURSORPOSITION
(A8)	FULLWORD	4	MQCIH-ERROROFFSET
(AC)	FULLWORD	4	MQCIH-INPUTITEM
(B0)	BINARY	4	reserved

MQCIH-STRUCID
The identifier for the CICS information header structure. This is an input field.

MQCIH-VERSION
The version number for the CICS information header structure. This must be MQCIH-VERSION-2. This is an input field.

MQCIH-STRUCLENGTH
The length of the CICS information header structure. This must be MQCIH-LENGTH-2. This is an input field.

MQCIH-GETWAITINTERVAL
The maximum wait interval for message input (in milliseconds). This is an input field.

MQCIH-FACILITYKEEPTIME
The length of time that the bridge facility will be kept after the user transaction has ended (in seconds). This is an input field.

MQCIH-ADSDESCRIPTOR
An indicator specifying whether ADS descriptors should be sent on SEND and RECEIVE BMS requests. The MQCADSD-MSGFORMAT value indicates that the ***long***form of the ADSD is used. Valid values are:

MQCADSD-NONE
MQCADSD-SEND
MQCADSD-RECV
MQCADSD-SEND+MQCADSD-RECV
MQCADSD-SEND+MQCADSD-RECV+MQCADSD-MSGFORMAT

This is an input field.

MQCIH-CONVERSATIONALTASK
An indicator specifying whether the task should be allowed to issue requests for more information, or should abend. Valid values are:

MQCCT-YES
MQCCT-NO

This is an input field.

MQCIH-TASKENDSTATUS
The value returned on output messages showing the status of the user transaction. One of the following values will be returned:

MQCTES-NOSYNC
The user transaction has not yet completed, and has not syncpointed.

MQCTES-COMMIT
The user transaction has not yet completed, but has syncpointed the first unit of work.

MQCTES-BACKOUT
The user transaction has not yet completed. The current unit of work will be backed out.

MQCTES–ENDTASK
The user transaction has ended (or abended).

This is an output field.

MQCIH-FACILITY
The 8-byte bridge facility token. This value is returned on output messages when a keep time is specified. This is an input/output field.

MQCIH-ABENDCODE
The abend code returned if the transaction abends, otherwise this field is set to blanks. This is an output field.

MQCIH-AUTHENTICATOR
The password or passticket for the specified USERID. This is an input field.

MQCIH-TRANSACTIONID
The transaction identifier of the user transaction.

MQCIH-FACILITYLIKE
The name of an installed terminal that is to be used as a model for the bridge facility. A value of blanks means that the FACILITYLIKE is taken from the bridge transaction profile definition, or a default value is used. This is an input field.

MQCIH-ATTENTIONID
The initial value of the AID key when the transaction is started. This is a 1-byte value, left justified. It is an input field.

MQCIH-STARTCODE
An indicator set on output from CICS with the start code that is appropriate for the next transaction. Valid values are:

MQCSC-START
MQCSC-STARTDATA
MQCSC-TERMINPUT
MQCSC-NONE

This is an input field.

MQCIH-CANCELCODE
The abend code to be used to terminate the transaction (normally a conversational transaction that is requesting more data). Otherwise this field is set to blanks. This is an input field.

MQCIH-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. This is an output field.

MQCIH-CURSORPOSITION
The initial cursor position when the transaction is started. Subsequently, for conversational transactions, the cursor position is in the RECEIVE vector. This is an input field.

MQCIH-ERROROFFSET
The position of invalid data detected by the bridge exit. This field provides the offset from the start of the message to the location of the invalid data.

MQCIH-INPUTITEM
The current TS queue item number being processed by the bridge exit.

Standard header for all vectors:

Offset Hex	Type	Len	Name
(0)	STRUCTURE	16	BRMQ-VECTOR-HEADER
(0)	FULLWORD	4	BRMQ-VECTOR-LENGTH
(4)	CHARACTER	4	BRMQ-VECTOR-DESCRIPTOR
(8)	CHARACTER	4	BRMQ-VECTOR-TYPE
(C)	CHARACTER	4	BRMQ-VECTOR-VERSION

BRMQ-VECTOR-LENGTH
The length of the vector. On output, this is always rounded up to the next multiple of 4, to facilitate full word alignment of subsequent vectors in the message. On input to the bridge exit, it is advisable to round up to the next multiple of 4 for the same reason.

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

0402 RECEIVE

0404 SEND

0406 CONVERSE

0418 ISSUE ERASEAUP

100A RETRIEVE

1802 RECEIVE MAP

1804 SEND MAP

1806 SEND TEXT

1812 SEND CONTROL

BRMQ-VECTOR-TYPE
The vector type. Valid values are:

I Inbound to the bridge exit.

O Outbound from the bridge exit.

BRMQ-VECTOR-VERSION
The vector version number. Valid values are:

X'00000000'
The first version.

Outbound reply vectors

Outbound vectors carry reply messages flowing from the user transaction to the end-user transaction.

SEND:

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

BRMQ-SE-ERASE-INDICATOR
The type of ERASE specified by the CICS SEND command that caused the exit to be called. Valid character values, left justified, are:

N No ERASE.

E ERASE.

A ERASE ALTERNATE.

D ERASE DEFAULT.

BRMQ-SE-CTLCHAR
The CTLCHAR value specified by the SEND command that caused the exit to be called. If CTLCHAR is not specified, the default X'C3' is sent.

BRMQ-SE-STRFIELD-INDICATOR
The presence of STRFIELD on the SEND command. Valid character values, left justified, are:

Y STRFIELD specified.

N STRFIELD not specified.

BRMQ-SE-DEFRESP-INDICATOR
The presence of DEFRESP on the SEND command that caused the exit to be called. Valid character values, left justified, are:

Y DEFRESP specified.

N DEFRESP not specified.

BRMQ-SE-INVITE-INDICATOR
The presence of INVITE on the send command that caused the exit to be called. Valid character values, left justified, are:

Y INVITE specified.

N INVITE not specified.

BRMQ-SE-LAST-INDICATOR
The presence of LAST on the SEND command that caused the exit to be called. Valid character values, left justified, are:

Y LAST specified.

N LAST not specified.

BRMQ-SE-WAIT-INDICATOR
The presence of WAIT on the SEND command that caused the exit to be called. Valid character values, left justified, are:

Y WAIT specified.

N WAIT not specified.

BRMQ-SE-DATA-LEN
The length of the data associated with the FROM option of the SEND command that caused the exit to be called. This is explicitly defined in the LENGTH or FLENGTH option, or derived from the length of the field.

data
Character field of length BRMQ-SE-DATA-LEN containing the data addressed by the FROM option of the SEND command.

SEND CONTROL: The fields in this vector are included also in SEND MAP and SEND TEXT.

Offset Hex	Type	Len	Name
(0)	STRUCTURE	52	BRMQ-SEND-CONTROL
(0)	CHARACTER	16	Header
(10)	CHARACTER	4	BRMQ-SC-ERASE-INDICATOR
(14)	CHARACTER	4	BRMQ-SC-ERASEAUP-INDICATOR
(18)	CHARACTER	4	BRMQ-SC-FREEKB-INDICATOR

Offset Hex	Type	Len	Name
(1C)	CHARACTER	4	BRMQ-SC-ALARM-INDICATOR
(20)	CHARACTER	4	BRMQ-SC-FRSET-INDICATOR
(24)	CHARACTER	4	BRMQ-SC-LAST-INDICATOR
(28)	CHARACTER	4	BRMQ-SC-WAIT-INDICATOR
(2C)	FULLWORD	4	BRMQ-SC-CURSOR
(30)	CHARACTER	4	BRMQ-SC-MSR-DATA

BRMQ-SC-ERASE-INDICATOR

The type of ERASE specified by the CICS BMS SEND command that caused the exit to be called. Valid character values, left justified, are:

N No ERASE

E ERASE

A ERASE ALTERNATE

D ERASE DEFAULT

BRMQ-SC-ERASEAUP-INDICATOR

The presence of ERASEAUP on the BMS SEND command that caused the exit to be called. Valid character values, left justified, are:

Y ERASEAUP specified.

N ERASEAUP not specified.

BRMQ-SC-FREEKB-INDICATOR

The presence of FREEKB on the BMS SEND command that caused the exit to be called. Valid character values, left justified, are:

Y FREEKB specified.

N FREEKB not specified.

BRMQ-SC-ALARM-INDICATOR

The presence of ALARM on the BMS SEND command that caused the exit to be called. Valid values are:

Y ALARM specified.

N ALARM not specified.

BRMQ-SC-FRSET-INDICATOR

The presence of FRSET on the BMS SEND command that caused the exit to be called. Valid character values, left justified, are:

Y FRSET specified.

N FRSET not specified.

BRMQ-SC-LAST-INDICATOR

The presence of LAST on the BMS SEND command that caused the exit to be called. Valid character values, left justified, are:

Y LAST specified.

N LAST not specified.

BRMQ-SC-WAIT-INDICATOR

The presence of WAIT on the BMS SEND command that caused the exit to be called. Valid character values, left justified, are:

Y WAIT specified.

N WAIT not specified.

BRMQ-SC-CURSOR
The presence of CURSOR or CURSOR(*data-value*) on the BMS SEND command that caused the exit to be called. Valid character values, left justified, are:

-1 CURSOR specified with dynamic cursor positioning.

-2 Neither CURSOR nor CURSOR(*data-value*) specified.

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

BRMQ-SC-MSR-DATA
The value of the MSR option specified on the BMS SEND command that caused the exit to be called. Valid values are:

X'00000000'
MSR option not specified.

other The value of the MSR option specified.

SEND MAP:

Offset Hex	Type	Len	Name
(0)	STRUCTURE	88	BRMQ-SEND-MAP
(0)	CHARACTER	16	Header
(10)	CHARACTER	4	BRMQ-SC-ERASE-INDICATOR
(14)	CHARACTER	4	BRMQ-SC-ERASEAUP-INDICATOR
(18)	CHARACTER	4	BRMQ-SC-FREEKB-INDICATOR
(1C)	CHARACTER	4	BRMQ-SC-ALARM-INDICATOR
(20)	CHARACTER	4	BRMQ-SC-FRSET-INDICATOR
(24)	CHARACTER	4	BRMQ-SC-LAST-INDICATOR
(28)	CHARACTER	4	BRMQ-SC-WAIT-INDICATOR
(2C)	FULLWORD	4	BRMQ-SC-CURSOR
(30)	CHARACTER	4	BRMQ-SC-MSR-DATA
(34)	CHARACTER	8	BRMQ-SM-MAPSET
(3C)	CHARACTER	8	BRMQ-SM-MAP
(44)	CHARACTER	4	BRMQ-SM-DATA-INDICATOR
(48)	FULLWORD	4	BRMQ-SM-DATA-LEN
(4C)	FULLWORD	4	BRMQ-SM-DATA-OFFSET
(50)	FULLWORD	4	BRMQ-SM-ADSD-LEN
(54)	FULLWORD	4	BRMQ-SM-ADSD-OFFSET
(58)	CHARACTER		variable length data

Fields **BRMQ-SC-ERASE-INDICATOR** to **BRMQ-SC-MSR-DATA** are defined in "SEND CONTROL" on page 64.

BRMQ-SM-MAPSET
The value of the MAPSET option specified by the SEND MAP command that caused the exit to be called.

BRMQ-SM-MAP
The value of the MAP option specified by the SEND MAP command that caused the exit to be called.

BRMQ-SM-DATA-INDICATOR
The presence of MAPONLY and DATAONLY options on the SEND MAP command that caused the exit to be called. Valid character values, left justified, are:

D DATAONLY specified.

M MAPONLY specified.

N Neither DATAONLY nor MAPONLY specified.

BRMQ-SM-DATA-LEN
The length of the data associated with the FROM option on the SEND MAP command that caused the exit to be called. This is the length of the symbolic map or ADS (application data structure).

BRMQ-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 that caused the exit to be called.

BRMQ-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. See "ADS descriptor area" on page 100 for a description of the ADS.

BRMQ-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.

data
Character field of length BRMQ-SM-DATA-LEN containing the data specified by the FROM option of the SEND MAP command, in ADS (Application Data Structure) format. This is followed by an ADS descriptor for this data, of length BRMQ-SM-ADSD-LEN, if an ADS descriptor was requested.

SEND TEXT:

Offset Hex	Type	Len	Name
(0)	STRUCTURE	60	BRMQ-SEND-TEXT
(0)	CHARACTER	16	Header
(10)	CHARACTER	4	BRMQ-SC-ERASE-INDICATOR
(14)	CHARACTER	4	BRMQ-SC-ERASEAUP-INDICATOR
(18)	CHARACTER	4	BRMQ-SC-FREEKB-INDICATOR
(1C)	CHARACTER	4	BRMQ-SC-ALARM-INDICATOR
(20)	CHARACTER	4	BRMQ-SC-FRSET-INDICATOR
(24)	CHARACTER	4	BRMQ-SC-LAST-INDICATOR
(28)	CHARACTER	4	BRMQ-SC-WAIT-INDICATOR
(2C)	FULLWORD	4	BRMQ-SC-CURSOR
(30)	CHARACTER	4	BRMQ-SC-MSR-DATA
(34)	CHARACTER	4	BRMQ-ST-TEXT-TYPE
(38)	FULLWORD	4	BRMQ-ST-DATA-LEN
(3C)	CHARACTER		variable length data

Fields **BRMQ-SC-ERASE-INDICATOR** to **BRMQ-SC-MSR-DATA** are defined in "SEND CONTROL" on page 64.

BRMQ-ST-TEXT-TYPE
The presence of MAPPED or NOEDIT options on the SEND TEXT command that caused the exit to be called. Valid character values, left justified, are:

M MAPPED specified.

N NOEDIT specified.

blank Neither MAPPED nor NOEDIT specified.

BRMQ-ST-DATA-LEN

The length of the text associated with the FROM option of the SEND TEXT command that caused the exit to be called.

data

Character field of length BRMQ-ST-DATA-LEN containing the data specified by the FROM option of the SEND TEXT command that caused the exit to be called. For SEND TEXT MAPPED, the additional 4 bytes created by the SEND MAP command, are included in this field, but the additional length is not included in BRMQ-ST-DATA-LEN.

ISSUE ERASEAUP:

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

BRMQ-IE-WAIT-INDICATOR

The presence of the WAIT option on the ISSUE ERASEAUP command that caused the exit to be called. Valid character values, left justified, are:

Y WAIT specified.

N WAIT not specified.

Outbound request vectors

Outbound request vectors flow from the user transaction to the end-user transaction requesting further input. There is only one in each message and it must be last.

RECEIVE REQUEST:

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

BRMQ-RER-BUFFER-INDICATOR

The presence of the BUFFER option on the RECEIVE request that caused the exit to be called. Valid character values, left justified, are:

Y BUFFER specified.

N BUFFER not specified.

RECEIVE MAP REQUEST:

Offset Hex	Type	Len	Name
(0)	STRUCTURE	36	BRMQ-RECEIVE-MAP-REQUEST
(0)	CHARACTER	16	Header
(10)	CHARACTER	8	BRMQ-RMR-MAPSET
(18)	CHARACTER	8	BRMQ-RMR-MAP
(20)	FULLWORD	4	BRMQ-RMR-ADSD-LEN
(24)	CHARACTER		variable length data

BRMQ-RMR-MAPSET
The value of the MAPSET option on the RECEIVE MAP command that caused the exit to be called.

BRMQ-RMR-MAP
THE value of the MAP option on the RECEIVE MAP command that caused the exit to be called.

BRMQ-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 (MQCIH-ADSDESCRIPTOR set to MQCADSD-NONE).

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

CONVERSE REQUEST:

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

BRMQ-COR-ERASE-INDICATOR
The type of ERASE specified by the CICS SEND command that caused the exit to be called. Valid character values, left justified, are:

N No ERASE.

E ERASE.

A ERASE ALTERNATE.

D ERASE DEFAULT.

BRMQ-COR-CTLCHAR
The CTLCHAR value specified by the SEND command that caused the exit to be called. If CTLCHAR is not specified, the default X'C3' is sent.

BRMQ-COR-STRFIELD-INDICATOR
The presence of STRFIELD on the SEND command. Valid character values, left justified, are:

Y STRFIELD specified.

N STRFIELD not specified.

BRMQ-COR-DEFRESP-INDICATOR
The presence of DEFRESP on the SEND command that caused the exit to be called. Valid character values, left justified, are:

Y DEFRESP specified.

N DEFRESP not specified.

BRMQ-COR-DATA-LEN
The length of the data associated with the FROM option of the SEND command that caused the exit to be called. This is explicitly defined in the LENGTH or FLENGTH option, or derived from the length of the field.

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

Inbound vectors

Inbound vectors flow from the end-user transaction to the user transaction carrying data to satisfy a user transaction RECEIVE, CONVERSE, or RETRIEVE.

RECEIVE:

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

BRMQ-RE-TRANSMIT-SEND-AREAS
A flag indicating whether previously generated, but not yet transmitted, SEND areas are to be preserved. Valid character values, left justified, are:

Y Preserve untransmitted SEND areas.

N Delete untransmitted SEND areas.

BRMQ-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 character values, left justified, are:

Y Data in BUFFER format.

N Data not in BUFFER format.

BRMQ-RE-AID
The AID key that was simulated to generate data in response to the RECEIVE command. The first byte of this field contains equivalent values to EIBAID, as defined by DFHAID. The remaining three bytes are padded with blanks. This value is inserted into the EIBAID field.

BRMQ-RE-CPOSN
The offset of the cursor at the time the RECEIVE data was generated. This value is inserted in EIBCPOSN for the transaction issuing the EXEC CICS RECEIVE.

BRMQ-RE-DATA-LEN
The length of the data provided in this vector in response to the RECEIVE command. This value is copied into the LENGTH or FLENGTH field.

data

Character field of length BRMQ-RE-DATA-LEN to be copied into the INTO area, or referenced by the SET option, of the RECEIVE command that caused the exit to be called.

RECEIVE MAP:

Offset Hex	Type	Len	Name
(0)	STRUCTURE	48	BRMQ-RECEIVE-MAP
(0)	CHARACTER	16	Header
(10)	CHARACTER	4	BRMQ-RM-TRANSMIT-SEND-AREAS
(14)	CHARACTER	8	BRMQ-RM-MAPSET
(1C)	CHARACTER	8	BRMQ-RM-MAP
(24)	CHARACTER	4	BRMQ-RM-AID
(28)	FULLWORD	4	BRMQ-RM-CPOSN
(2C)	FULLWORD	4	BRMQ-RM-DATA-LEN
(30)	CHARACTER		variable length data

BRMQ-RM-TRANSMIT-SEND-AREAS

A flag indicating whether previously generated, but not yet transmitted, SEND areas are to be preserved. Valid character values, left justified, are:

Y Preserve untransmitted SEND areas.

N Delete untransmitted SEND areas.

BRMQ-RM-MAPSET

The name of the MAPSET to be used to present the data. If this is blank, the data is to be presented to the next RECEIVE MAP command, regardless of the MAPSET value specified by the command.

BRMQ-RM-MAP

The name of the MAP to be used to present the data. If this is blank, the data is to be presented to the next RECEIVE MAP command, regardless of the MAP value specified by the command.

BRMQ-RM-AID

The AID key that was simulated to generate data in response to the RECEIVE command. The first byte of this field contains equivalent values to EIBAID, as defined by DFHAID. The remaining three bytes are padded with blanks. This value will be inserted into the EIBAID field for the transaction issuing the EXEC CICS RECEIVE MAP.

BRMQ-RM-CPOSN

The offset of the cursor at the time the RECEIVE data was generated. This value will be inserted into the EIBCPOSN field for the transaction issuing the EXEC CICS RECEIVE MAP.

BRMQ-RM-DATA-LEN

The length of the data provided in this vector in response to the RECEIVE MAP command that caused the exit to be called. This value is copied into the LENGTH or FLENGTH field.

data

Character field of length BRMQ-RM-DATA-LEN, in ADS (Application Data Structure) format equivalent to the MAP and MAPSET specified by the RECEIVE command.

CONVERSE:

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

BRMQ-CO-TRANSMIT-SEND-AREAS
A flag indicating whether previously generated, but not yet transmitted, SEND areas are to be preserved. Valid character values, left justified, are:

Y Preserve untransmitted SEND areas.

N Delete untransmitted SEND areas.

BRMQ-CO-AID
The AID key that was simulated to generate data in response to the RECEIVE command. The first byte of this field contains equivalent values to EIBAID, as defined by DFHAID. The remaining three bytes are padded with blanks. This value is inserted into the EIBAID field.

BRMQ-CO-CPOSN
The offset of the cursor at the time the RECEIVE data was generated. This value is inserted in EIBCPOSN for the transaction issuing the EXEC CICS RECEIVE.

BRMQ-CO-DATA-LEN
The length of the data provided in this vector in response to the RECEIVE command. This value is copied into the LENGTH or FLENGTH field.

data
Character field of length BRMQ-CO-DATA-LEN to be copied into the INTO area, or referenced by the SET option, of the CONVERSE command that caused the exit to be called.

RETRIEVE:

Offset Hex	Type	Len	Name
(0)	STRUCTURE	36	BRMQ-RETRIEVE
(0)	CHARACTER	16	Header
(10)	CHARACTER	4	BRMQ-RT-RTRANSID
(14)	CHARACTER	4	BRMQ-RT-RTERMID
(18)	CHARACTER	8	BRMQ-RT-QUEUE
(20)	FULLWORD	4	BRMQ-RT-DATA-LEN
(24)	CHARACTER		variable length data

BRMQ-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.

BRMQ-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.

BRMQ-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.

BRMQ-RT-DATA-LEN
The length of the data provided in this vector in response to the RETRIEVE command that caused the exit to be called. This value is copied into the LENGTH or FLENGTH field.

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

Note: The RETRIEVE vector is only valid in the first inbound message. It is ignored in other messages.

Chapter 6. Writing your own bridge programs

If you do not want to use the IBM supplied bridge exits and monitors, you can modify them, or write your own. This chapter tells you how you can modify the supplied exits and about the interfaces that are defined by CICS that you must support in your own programs. It covers the following topics:

- “Designing your own bridge solution”
- “Writing your own bridge exit” on page 78
- “Writing your own formatter” on page 81
- “Bridge exit area (BRXA)” on page 84
- “Supplied copybooks” on page 105

Designing your own bridge solution

The bridge mechanism is very flexible, but most implementations fall into a few basic models.

“Implementing a 3270 bridge environment” on page 30 tells you how to identify the model that fits the requirements of your system and applications. It then presents examples of each model, which you can use as checklists when preparing your own client programs, bridge exits and monitors.

Is a new bridge exit needed?

If you want to use a transport mechanism other than TS, TD or the Web, or you want to use a message format that doesn’t have MQCIH as the message header, it may be simpler to modify DFH0CBRE, rather than write your own complete solution.

Changing the transport mechanism

The supplied exit has three sections marked ″transport mechanism specific″. These should be changed to replace the TS/TD code with transport specific code. The brdata may also need to change. This is contained in the DFHBRSCO and DFHBRSDO copybooks, and in the INIT routine.

Changing the message header

The only user of the MQCIH message header is the DFH0CBRE bridge exit. It is not used in the formatter, or the common code. Therefore if you want to simplify the MQCIH, for example to hard code most of the values in the exit, and have a reduced header, this can be done by changing the DFHMQMHO and DFHMQMCO copy books in the ″message header″ sections and routines marked as ″MSG-HDR SPECIFIC ROUTINES″.

If you want a much simpler interface which is tailored for a specific(type) of application, then you should consider writing your own bridge exit that is specifically designed for a single transaction (or a number of transactions with identical interfaces).

An example of this approach is described in the Dallas System Center Redbook.

Is a new formatter needed?

If you want to use a message format that doesn’t use BRMQ message vectors, it may be simpler to modify DFH0CBRF, rather than write your own complete solution.

Changing the message vectors

The only user of the BRMQ message vectors is the DFH0CBRF formatter. It is not used in the bridge exit, or the common code. Therefore if you want to simplify the BRMQ vectors, for example to hard code values in the exit, and have a reduced header, this can be done by changing the DFHBRMQ copy book in the ″message structures″ section, and occurances of the BRMQ field

If you want a much simpler interface which is tailored for a specific(type) of application, then you should consider writing your own formatter that is specifically designed for a single transaction (or a number of transactions with identical interfaces). The data can be read by the bridge exit as normal, but the formatter can be tailored for only those API interfaces that are specifically required.

BMS macro generation utility program (DFHBMSUP)

This chapter describes the BMS macro generation utility, DFHBMSUP, to recreate BMS macro statements from a mapset load module.

Overview

DFHBMSUP can recreate the original BMS macros that were assembled to produce a mapset load module, when the macro statements are no longer available.

The utility program generates map definition macros that are equivalent to the originals, and thus can be used to recreate symbolic maps if the original source has been lost. However, it is not possible to recover the original field names used. Field names are generated by the utility and you can then edit them.

DFHBMSUP sets a return code indicating success or failure.

Note: DFHBMSUP cannot process mapset load modules created on CICS/DOS/VS 1.7 and earlier releases.

Input

All input information is defined in the JCL.

DFHBMSUP requires the following inputs:

Input MAPSET
Name defined in the PARM field of the EXEC statement.

The library in which the mapset resides must be included in the LIBDEF search statement.

Output

DFHBMSUP provides the following outputs:

Output map
Name defined in the BMSOUT DLBL statement.

DLBL statement

This section describes the DLBL statement for the output file used by DFHBMSUP.

BMSOUT DLBL
Defines a file contain the BMS macro statements generated by the utility.

Return codes

DFHBMSUP sets one of the following return codes:

0 Utility executed successfully.

4 Input mapset could not be loaded.

8 Output mapset could not be opened.

Example of using DFHBMSUP

Figure 24 shows the statements required to process a BMS mapset load module. Macro statements are generated and written to the **MAPOUT** file.

```
/* *******************************
/*  Sample job for running the   *
/*  DFHBMSUP to create a BMS MAP *
/*  from the assembled Mapset    *
/*                               *
/* *******************************
// OPTION PARTDUMP
// DLBL BMSOUT,'output_file',0,SD
// EXTENT SYS019,volume,1,0,xxx,xx
// ASSGN SYS019,DISK,VOL-volume,SHR
// ASSGN SYS009,SYSLST
// LIBDEF *,SEARCH=search_list
// EXEC PGM=DFHBMSUP,SIZE=DFHBMSUP,parm='mapname'
/*
/&
* ££ EOJ
```

Figure 24. DFHBMSUP—generating BMS macro statements.

Example of DFHBMSUP output

The following macro statements were generated from the mapset load module, BMSET40. Note that the utility generates a new name for the mapset, @000001, so that you can assemble it without overwriting the existing mapset.

You can edit all the names to be more meaningful for your application.

```
* This is an unaligned mapset
*
         TITLE 'BMSET40 Mapset MACRO Definition Listing'
@000001  DFHMSD TYPE=DSECT,LANG=ASM,MODE=INOUT
*
BMAP400  DFHMDI SIZE=(1,80),CTRL=(FRSET,FREEKB),COLUMN=1,LINE=1,       *
               MAPATTS=(COLOR,HILIGHT)
         DFHMDF POS=0,LENGTH=4,ATTRB=(ASKIP,BRT),COLOR=PINK,           *
               HILIGHT=REVERSE,INITIAL='BM40'
         DFHMDF POS=5,LENGTH=1,COLOR=BLUE
FLD00001 DFHMDF POS=16,LENGTH=45,ATTRB=(ASKIP,BRT),COLOR=NEUTRAL
         DFHMDF POS=62,LENGTH=1,COLOR=BLUE
FLD00002 DFHMDF POS=78,LENGTH=1,COLOR=YELLOW
BMAP401  DFHMDI SIZE=(9,80),CTRL=(FRSET,FREEKB),COLUMN=1,LINE=2,       *
               MAPATTS=(COLOR,HILIGHT)
         DFHMDF POS=0,LENGTH=1,COLOR=BLUE,INITIAL=' '
         DFHMDF POS=80,LENGTH=1,COLOR=BLUE,INITIAL=' '
         DFHMDF POS=160,LENGTH=1,COLOR=BLUE,INITIAL=' '
         DFHMDF POS=240,LENGTH=1,COLOR=BLUE,INITIAL=' '
         DFHMDF POS=320,LENGTH=1,COLOR=BLUE,INITIAL=' '
         DFHMDF POS=400,LENGTH=1,COLOR=BLUE,INITIAL=' '
         DFHMDF POS=480,LENGTH=1,COLOR=BLUE,INITIAL=' '
         DFHMDF POS=560,LENGTH=1,COLOR=BLUE,INITIAL=' '
         DFHMDF POS=658,LENGTH=39,COLOR=TURQUOISE,                     *
               INITIAL='THIS SHOULD BE IN THE MIDDLE OF LINE 10'
*
BMAP402  DFHMDI SIZE=(1,80),CTRL=(FRSET,FREEKB),COLUMN=1,LINE=11,      *
               MAPATTS=(COLOR,HILIGHT)
         DFHMDF POS=0,LENGTH=1,COLOR=BLUE,INITIAL=' '
BMAP403  DFHMDI SIZE=(1,80),CTRL=(FRSET,FREEKB),COLUMN=1,LINE=11,      *
               MAPATTS=(COLOR,HILIGHT)
         DFHMDF POS=17,LENGTH=41,COLOR=TURQUOISE,                      *
```

```
               INITIAL='THIS TEXT SHOULD NOT APPEAR ON THE SCREEN'
*
BMAP404  DFHMDI SIZE=(10,80),CTRL=(FRSET,FREEKB),COLUMN=1,LINE=12,     *
               MAPATTS=(COLOR,HILIGHT)
         DFHMDF POS=18,LENGTH=39,COLOR=TURQUOISE,                      *
               INITIAL='THIS SHOULD BE IN THE MIDDLE OF LINE 12'
         DFHMDF POS=80,LENGTH=1,COLOR=BLUE,INITIAL=' '
         DFHMDF POS=160,LENGTH=1,COLOR=BLUE,INITIAL=' '
         DFHMDF POS=240,LENGTH=1,COLOR=BLUE,INITIAL=' '
         DFHMDF POS=320,LENGTH=1,COLOR=BLUE,INITIAL=' '
         DFHMDF POS=400,LENGTH=1,COLOR=BLUE,INITIAL=' '
         DFHMDF POS=480,LENGTH=1,COLOR=BLUE,INITIAL=' '
         DFHMDF POS=560,LENGTH=1,COLOR=BLUE,INITIAL=' '
         DFHMDF POS=640,LENGTH=1,COLOR=BLUE,INITIAL=' '
         DFHMDF POS=720,LENGTH=1,COLOR=BLUE,INITIAL=' '
*
BMAP405  DFHMDI SIZE=(3,80),CTRL=(FRSET,FREEKB),COLUMN=1,LINE=22,      *
               MAPATTS=(COLOR,HILIGHT)
FLD00003 DFHMDF POS=80,LENGTH=78,COLOR=BLUE
         DFHMDF POS=160,LENGTH=41,COLOR=BLUE,                          *
               INITIAL='PF1=HELP        PF3=EXIT      PF12=RETURN'
         DFHMDF POS=208,LENGTH=30,COLOR=BLUE,                          *
               INITIAL='ENTER=CONTINUE      CLEAR=EXIT'
@000001  DFHMSD TYPE=FINAL
         END
```

Writing your own bridge exit

Your bridge exit is always called for the following requests:
- User transaction initialization
- User transaction bind
- User transaction termination
- User transaction abnormal termination
- Syncpoint (optional)

If a formatter is specified in the INIT call, then the bridge exit is also called for Read and Write message requests.

If a formatter is specified in the INIT call, it is called for the following requests. Otherwise, if a formatter is not used, the bridge exit is also called for these requests:
- SEND (Terminal Control and BMS)
- RECEIVE (Terminal Control and BMS)
- CONVERSE
- FREE
- ISSUE DISCONNECT
- ISSUE ERASEAUP
- RETRIEVE (in some cases)

Transaction calls to the bridge exit

The following calls are made at transaction initialization, termination, syncpoint and abend:

INIT (Initialization) call

This call is made by the CICS transaction manager during the establishment of the bridge environment for the user transaction. The bridge facility is identified and a FACILITY_TOKEN created. If a null FACILITY_TOKEN is supplied, the bridge exit establishes a new bridge facility and creates a new token; if a valid FACILITY_TOKEN is supplied, the bridge exit uses the existing bridge facility, and the parameter FACILITYLIKE is ignored.

This call also processes the BRDATA passed on the START command, storing the passed data in the Bridge Exit Area (BRXA) for subsequent use during the execution of the user transaction.

The following values can be set in the transaction and common area of the BRXA on this call. Any other values are ignored.

Table 7. Init call parameters

Field Name	Default
BRXA_FACILITY_TOKEN	nulls
BRXA_FACILITYLIKE	blanks
BRXA_FORMATTER	blanks
BRXA_USER_ABEND_CODE	blanks
BRXA_CALL_EXIT_FOR_SYNCPOINT	BRXA_YES

Commands that cause an explicit or implicit syncpoint, or that use a resource of a type that can be defined as recoverable cannot be used in the the INIT call. Invalid commands are:

- File commands
- Temporary storage commands
- Transient data commands
- Task related user exit requests
- CREATE commands
- SYNCPOINT commands
- ENQ commands
- CICS Business Transaction Services (BTS) commands

BIND call

This call is made by the CICS transaction manager during task initalization, when the Unit of Work (UOW) is created.

It is used to open queues and possibly obtain the message to run the user transaction.

You can also use the EXEC CICS VERIFY command in this call to validate a password or pass-ticket in the message.

The following BRXA parameters can be set in the BIND call:

Table 8. Bind call parameters

Field Name	Default
BRXA_STARTCODE	BRXA_TERMINPUT (if BRXA_FACILITY_TOKEN is null) or value from the bridge facility
BRXA_USER_ABEND_CODE	blanks
BRXA_LOAD_ADS_DESCRIPTOR	BRXA_NO
BRXA_IDENTIFIER	nulls
BRXA_FORMATTER	blanks

SYNCPOINT call

This is only called if the field BRXA_CALL_EXIT_FOR_SYNCPOINT is set. BRXA_SYNC_COMMAND is set to indicate whether the SYNCPOINT is a rollback. This call allows the exit to write a request to the client before and/or after the syncpoint call. Note that if the call is a rollback, the write request should not be recoverable. The SYNCPOINT call must be reissued exactly the same as the original call.

TERM (termination) call

This call is made by the CICS transaction manager when the user transaction issues a RETURN command. On this call the bridge exit sends the response message back to the client application. This can be done by a direct interface to the transport mechanism or indirectly by passing the response message to the bridge monitor for transmission.

This call also identifies the next transaction to be run if this has been specified and can then issue an EXEC CICS START BREXIT command for the next TRANSID, or return the next transaction information to the client application.

The following values can be set in the transaction and common area of the BRXA on this call. Any other values are ignored.

- BRXA_FACILITY_KEEP_TIME
- BRXA_USER_ABEND_CODE

ABEND call

This call is made if the user transaction abends, so that the bridge exit can send non-recoverable messages to the client application. For example, a non-syncpointing MQPUT can be issued for the MQ bridge.

Recoverable requests cannot be made in this call.

BRXA_USER_ABEND_CODE can not be set in this call.

The following values can be set in the transaction and common areas on this call. Any other values are ignored.

- BRXA_FACILITY_KEEP_TIME

Message calls to the bridge exit

The following calls are made if a formatter is specified in the INIT call:

brxa-read-message-nowait

The purpose of this call is to read the next message if there already is one available. It is only used if the client application writes more than one message at a time. When a message is read, the bridge exit could check the password or pass-ticket in the message using an EXEC CICS VERIFY PASSWORD to ensure that the message came from an authorized source.

brxa-read-message-wait

The purpose of this call to to get the next message. This is usually done by sending a message to the client, requesting the next message, and waiting for a reply. When a message is read, the bridge exit could check the password or pass-ticket in the message using an EXEC CICS VERIFY PASSWORD to ensure that the message came from an authorized source.

brxa-write-message

The purpose of this call is to write a message. This is an intermediate message due to either a flush request, or the message buffer being full.

API calls to the bridge exit

These calls are only made if a formatter is not specified in the INIT call:

SEND call
This call is made to the bridge exit when the user transaction issues a SEND command. This is the reverse of what happens in a RECEIVE command. In this case the bridge exit copies data from the command area to a message to be returned to the client application.

RECEIVE call
This call is made to the bridge exit when the user transaction issues a RECEIVE command. Using the input obtained from BRDATA (or from a client application message), data is copied to the INTO or SET storage for the user transaction RECEIVE command. For RECEIVE MAP calls, this is the ADS. The bridge exit can set the following fields in the command area:

- The EIBRESP/EIBRESP2 values (defaults to NORMAL)
- The EIBAID value (defaults to ENTER)
- The EIBCPOSN value (defaults to 0)

CONVERSE
This call combines the function of SEND and RECEIVE.

FREE
After this call has been made, no further API calls should be made.

ISSUE DISCONNECT
After this call has been made, no further API calls should be made.

ISSUE ERASEAUP
This call causes the exit clear the screen data.

RETRIEVE
RETRIEVE calls are normally only used in the first leg of a pseudoconversation, when a startcode of SD is returned. All other RETRIEVE requests should return ENDDATA.

Writing your own formatter

If the bridge exit specifies a formatter, the formatter is called for the following requests:

- SEND (Terminal Control and BMS)
- RECEIVE (Terminal Control and BMS)
- CONVERSE
- FREE
- ISSUE DISCONNECT
- ISSUE ERASEAUP
- RETRIEVE (in some cases)

Calls to the formatter

Your formatter must handle the following calls:

RECEIVE call
This call is made to the formatter when the user transaction issues a RECEIVE command. Using the input obtained from BRDATA (or from a client application message), data is copied to the INTO or SET storage for the user transaction RECEIVE command. For RECEIVE MAP calls, this is the ADS. The bridge exit can set the following fields in the command area:

- The EIBRESP/EIBRESP2 values (defaults to NORMAL)
- The EIBAID value (defaults to ENTER)
- The EIBCPOSN value (defaults to 0)

If the RECEIVE does not have data to answer a receive request (for a conversational transaction), then if the client can send more than one message at a time, this call can return a response of BRXA-FMT-READ-MESSAGE-NOWAIT to CICS to ask if the next message has already arrived. CICS then calls the formatter again. If a message was available the formatter will be called again for the same request. This time there should be a information to process the command. If there is (still) nothing in the message to answer the receive request, this call can add a vector to the end of the message, requesting more data, and return a response of BRXA-FMT-REQUEST-NEXT-MESSAGE to CICS. The bridge exit will write the current message, and read the next message when it arrives. When the message is read, the formatter will be called again. This time there should be information to process the command.

SEND call
This call is made to the bridge exit when the user transaction issues a SEND command. This is the reverse of what happens in a RECEIVE command. In this case the bridge exit copies data from the command area to a message to be returned to the client application. If the message is too large to process, the send call returns a response of BRXA-FMT-OUTPUT-BUFFER-FULL to CICS. This results in the bridge exit being called to send the message and free the buffer space. The formatter will be called again for the same request. If the SEND results in the current message being flushed, this call should return a response of BRXA-FMT-WRITE-MESSAGE. The formatter will not be called again for the same request.

CONVERSE
This call combines the function of SEND and RECEIVE.

FREE
After this call has been made, no further API calls should be made.

ISSUE DISCONNECT
After this call has been made, no further API calls should be made.

ISSUE ERASEAUP

RETRIEVE
RETRIEVE calls are normally only used in the first leg of a pseudoconversation, when a startcode of SD is returned. All other RETRIEVE requests should return ENDDATA.

Return codes from the formatter

If the formatter cannot fully process a command, it passes a return code back to CICS in the field BRXA_FMT_RESPONSE. This gives the formatter the option of calling the bridge exit. CICS also passes state information to the formatter in fields BRXA_READ_NOWAIT_ISSUED and BRXA_REQUEST_NEXT_ISSUED.

BRXA_FMT_RESPONSE

The following output values can be returned to CICS in BRXA_FMT_RESPONSE

BRXA_FMT_NONE
(default) No action. The formatter has processed the request.

BRXA_FMT_OUTPUT_BUFFER_FULL
There is no room to add the next vector. Call the bridge exit to write the message, clear the buffer, then call the formatter again.

BRXA_FMT_WRITE_MESSAGE
The request required data to be flushed. Call the bridge exit to write the message.

BRXA_FMT_REQUEST_NEXT_MESSAGE
The formatter has processed all the data in the message. Call the bridge exit to read another message, then call the formatter again.

BRXA_FMT_READ_MESSAGE_NOWAIT
The formatter has processad all the data in the message. Check to see if there is a new message before requesting any further input. Call the bridge exit to read a message, then call the formatter again.

BRXA_READ_NOWAIT_ISSUED

The following values are passed to the formatter in BRXA_READ_NOWAIT_ISSUED. This field is used by the formatter to check if it has already returned a BRXA_FMT_READ_MESSAGE_NOWAIT for this command.

BRXA_NO
A BRXA_FMT_READ_MESSAGE_NOWAIT has not been returned for this command.

BRXA_YES
A BRXA_FMT_READ_MESSAGE_NOWAIT has been returned for this command.

BRXA_REQUEST_NEXT_ISSUED

The following values are passed to the formatter in BRXA_REQUEST_NEXT_ISSUED. This field is used by the formatter to check if it has already returned a BRXA_FMT_REQUEST_NEXT_MESSAGE for this command.

BRXA_NO
A BRXA_FMT_REQUEST_NEXT_MESSAGE has not been returned for this command.

BRXA_YES
A BRXA_FMT_REQUEST_NEXT_MESSAGE has been returned for this command.

Bridge exit area (BRXA)

This section contains Product-sensitive Programming Interface and Associated Guidance Information.

The bridge exit area (BRXA) is the interface between the bridge exit or the formatter and CICS.

This is an interface defined by CICS, and must be used by all bridge exits and formatters. It is a CICS COMMAREA, accessed by ADDRESS COMMAREA, which is passed to the bridge exit or formatter by CICS whenever it is called.

The names of the parameters and constants, translated into appropriate forms for the different programming languages supported, are supplied in the copybook files listed in “Supplied copybooks” on page 105.

The BRXA contains a number of sub-areas that are used by the bridge exit and formatter to process each call, and retain information between calls. It consists of the following sub-areas:

Header
This area contains version information and pointers to some of the following areas.

Transaction area
This area is used by bridge exit initialization processing. It contains information about the user transaction that CICS will run, and the real 3270 that it expects to use.

Command area
This area provides details of the command request. For CICS API requests it provides a simplified description of the command and response fields.

User area
This area is used to store data between calls to the bridge exit. It acts as a user input area to store the messages needed to satisfy RECEIVE and RETRIEVE requests, and also as a user output area to store the messages from SEND requests so that they can all be sent together when the user transaction terminates.

ADS descriptor
This area contains an ADS descriptor for BMS SEND MAP and RECEIVE MAP requests, if the mapset has been assembled with CICS Transaction Server for VSE/ESA Release 1.1.1 or later release and an ADS descriptor has been created.

The ADS descriptor is created by the BMS macros in either a ***long*** or a ***short*** form. Long data has the same content as short data, but the fields are word aligned to support those transport mechanisms that require all message fields to be word aligned.

BRXA header area

Offset Hex	Type	Len	Name
(0)	STRUCTURE	56	BRXA_HEADER
(0)	CHARACTER	8	BRXA_HEADER_EYECATCHER
(8)	FULLWORD	4	BRXA_HEADER_LENGTH
(C)	FULLWORD	4	BRXA_HEADER_VERSION_NO

Offset Hex	Type	Len	Name
(10)	ADDRESS	4	BRXA_TRANSACTION_AREA_PTR
(14)	FULLWORD	4	BRXA_TRANSACTION_AREA_LEN
(18)	ADDRESS	4	BRXA_COMMAND_AREA_PTR
(1C)	FULLWORD	4	BRXA_COMMAND_AREA_LEN
(20)	ADDRESS	4	BRXA_USER_AREA_PTR
(24)	FULLWORD	4	BRXA_USER_AREA_LEN
(28)	ADDRESS	4	BRXA_INPUT_MSG_PTR
(2C)	FULLWORD	4	BRXA_INPUT_MSG_LEN
(30)	ADDRESS	4	BRXA_OUTPUT_MSG_PTR
(34)	FULLWORD	4	BRXA_OUTPUT_MSG_LEN

The BRXA header contains the following fields:

BRXA_HEADER_EYECATCHER
An eye-catcher to identify the area as an BRXA. This is initialized by CICS to the value BRXA_HEADER_EYE ('>BRAREA '), which is defined in the DFHBRACx copy books.

BRXA_HEADER_LENGTH
The length of the header.

BRXA_HEADER_VERSION_NO
The version number of the BRXA. This allows future releases to extend the BRXA. This is initialized by CICS to the value of BRXA_CURRENT_VERSION_NO in the DFHBRACx copybook.

BRXA_TRANSACTION_AREA_PTR
The address of the transaction subarea, BRXA_TRANSACTION_AREA. This is set by CICS, and should not be modified by the bridge exit code.

BRXA_TRANSACTION_AREA_LEN
The length of the transaction subarea, BRXA_TRANSACTION_AREA. This is set by CICS, and should not be modified by the bridge exit code.

BRXA_COMMAND_AREA_PTR
The address of the command subarea, BRXA_COMMAND_AREA, This is set by CICS, and should not be modified by the bridge exit code.

BRXA_COMMAND_AREA_LEN
The length of the command subarea, BRXA_COMMAND_AREA. This is set by CICS, and should not be modified by the bridge exit code.

BRXA_USER_AREA_PTR
A field that allows the address of a user area to be saved across bridge exit calls within a task. The user area should be obtained using an EXEC CICS GETMAIN.

BRXA_USER_AREA_LEN
A field in which the exit can save the length of the user area.

BRXA_INPUT_MSG_PTR
A field used to save the address of an input message. This field is intended to be used in conjunction with a formatter.

BRXA_INPUT_MSG_LEN
A field used to save the current length of the input message.

BRXA_OUTPUT_MSG_PTR
A field used to save the address of an output message. This field is intended to be used in conjunction with a formatter.

BRXA_OUTPUT_MSG_LEN
A field used to save the current length of the output message.

BRXA transaction area

Offset Hex	Type	Len	Name
(0)	STRUCTURE	180	BRXA_TRANSACTION_AREA
(0)	CHARACTER	8	BRXA_TRAN_AREA_EYECATCHER
(8)	CHARACTER	4	BRXA_BRIDGE_TRANID
(C)	CHARACTER	4	BRXA_TRANID
(10)	CHARACTER	4	BRXA_NEXTTRANID
(14)	CHARACTER	4	BRXA_ABEND_CODE
(18)	CHARACTER	8	BRXA_CALLING_PROG
(20)	CHARACTER	8	BRXA_USERID
(28)	CHARACTER	8	reserved
(30)	CHARACTER	2	BRXA_STARTCODE
(32)	CHARACTER	1	BRXA_LOAD_ADS_DESCRIPTOR
(33)	CHARACTER	1	BRXA_TRACE
(34)	CHARACTER	4	BRXA_FACILITYLIKE
(38)	UNSIGNED	4	BRXA_FACILITY_KEEP_TIME
(3C)	CHARACTER	8	BRXA_FACILITY_TOKEN
(44)	HALFWORD	2	BRXA_SCREEN_HEIGHT
(46)	HALFWORD	2	BRXA_SCREEN_WIDTH
(48)	HALFWORD	2	BRXA_ALTERNATE_SCREEN_HEIGHT
(4A)	HALFWORD	2	BRXA_ALTERNATE_SCREEN_WIDTH
(4C)	CHARACTER	48	BRXA_IDENTIFIER
(7C)	CHARACTER	8	BRXA_FORMATTER
(84)	CHARACTER	1	BRXA_CALL_EXIT_FOR_SYNC
(85)	CHARACTER	1	BRXA_NEXTTRANID_SOURCE
(86)	CHARACTER	6	reserved
(8C)	CHARACTER	8	reserved
(94)	FULLWORD	4	BRXA_BRDATA_PTR
(98)	FULLWORD	4	BRXA_BRDATA_LEN
(9C)	CHARACTER	4	BRXA_INTERVAL
(A0)	CHARACTER	4	BRXA_TIME
(A4)	FULLWORD	4	BRXA_HOURS
(A8)	FULLWORD	4	BRXA_MINUTES
(AC)	FULLWORD	4	BRXA_SECONDS
(B0)	CHARACTER	1	BRXA_START_AFTER
(B1)	CHARACTER	1	BRXA_START_AT
(B2)	CHARACTER	2	reserved

The transaction area contains the following fields:

BRXA_TRAN_AREA_EYECATCHER
An eye-catcher to identify the area as a BRXA transaction area. This is set by CICS, before passing control to the bridge exit, to the value BRXA_TRAN_AREA_EYE ('>BRTRANA'), defined in the DFHBRACx copy books.

BRXA_BRIDGE_TRANID
The transaction identifier of the bridge monitor transaction that issued a START TRANSID BREXIT command to start this bridge exit and its associated user transaction.

BRXA_TRANID
The transaction identifier of the user transaction.

BRXA_NEXTTRANID
The transaction identifier of the next transaction. This is set by CICS from the TRANSID value provided in the final RETURN command of the user transaction; from the value provided by a SET TERMINAL NEXTTRANSID command, or from the TRANSID of the first START issued by the user transaction for the bridge facility. This field contains blanks (X'40') if no next transaction has been specified.

BRXA_ABEND_CODE
The abend code if the bridge transaction abends before initializing the user transaction, or if the user transaction abends. If the transaction has not abended, this field is blanks.

BRXA_CALLING_PROG
The name of the program in the user transaction which issued the command causing the bridge exit to be invoked. For the initialization, termination, and abend calls, this field is set to blanks.

BRXA_USERID
The USERID under which the user transaction is running.

BRXA_STARTCODE
This field is set to the start code appropriate to the next transaction returned in BRXA_NEXTTRANID. The following start codes are possible:

brxa_start
START command without data.

brxa_startdata
START command with data.

brxa_terminput
Terminal input (default).

The initial value of this field is blanks.

BRXA_LOAD_ADS_DESCRIPTOR
A 1-character field that tells CICS whether or not to provide the ADS descriptor on subsequent SEND MAP and RECEIVE MAP commands.

If this field is set to 'Y' (BRXA_YES) when the user transaction issues a SEND MAP and RECEIVE MAP command, CICS loads the mapset, locates the ADS descriptor for the map, and provides its address in BRXA_ADS_DESCRIPTOR_PTR in the command subarea.

The ADS descriptor format is explained in "ADS descriptor area" on page 100.

If this field has any value other than 'Y', then CICS does not attempt to load the mapset and locate the descriptor, and BRXA_ADS_DESCRIPTOR_PTR is set to null.

This value can only be set in the INIT call.

BRXA_TRACE

A 1-character field that is set to 'Y' (BRXA_YES) if level-2 tracing is set on for the bridge. The bridge can use this flag to trace input and output data, for example, for diagnostic purposes.

Note that for BR level tracing, the BRXA is already traced by CICS on input and output.

BRXA_FACILITYLIKE

The name of an installed 3270 terminal to be used as a template terminal definition for the bridge facility.

The exit sets this value during the initialization call. If the exit does not provide a value, CICS looks for a value specified as FACILITYLIKE in the user transaction's profile. If this value is also blanks, CICS uses the CICS-supplied definition CBRF (based on model DFHLU2).

If the specified FACILITYLIKE does not exist, CICS abends the transaction ABRJ.

It is not possible to change the FACILITYLIKE definition after the terminal has been created, so this parameter is ignored if BRXA_FACILITY_TOKEN is specified.

If the real FACILITYLIKE terminal is logged on when the bridge facility is created, any values returned by QUERY will apply also to the bridge facility.

BRXA_FACILITY_KEEP_TIME

The time (in seconds) that the bridge facility is kept after the user transaction terminates. If a non-zero value is set in this field the bridge facility and its pseudo-conversational data are retained.

CICS sets this value to zero before the bridge exit initialization call. The exit can set it at any time; CICS does not use the value until the exit returns from the task termination call. If the value is zero, CICS discards the bridge facility; if non-zero, CICS retains the facility and associated data.

The maximum value is one week (604800 seconds). If a value larger than this is specified, CICS retains the bridge facility for one week.

BRXA_FACILITY_TOKEN

A token representing the bridge facility to be used. CICS initializes this value to nulls but the exit can set it in the initialization call.

Specifying a value implies reusing a bridge facility kept from a previous transaction.

The default value of nulls results in CICS dynamically allocating a new bridge facility.

The name of the bridge facility is accessible to the user transaction in the EIBTRMID field of the EIB. No other TERMIDs in the system are the same, although the name may be reused almost immediately when the user transaction finishes, if BRXA_FACILITY_KEEP_TIME is set to zero.

BRXA_SCREEN_HEIGHT

The current screen height.

BRXA_SCREEN_WIDTH
The current screen width.

BRXA_ALTERNATE_SCREEN_HEIGHT
The alternate screen height.

BRXA_ALTERNATE_SCREEN_WIDTH
The alternate screen width.

BRXA_IDENTIFIER
A 48-character field provided by the bridge exit. The intended use of this field is for task-specific information to assist in on-line problem resolution. It could contain, for example, the MQ correlator for the MQ bridge, or a Web token.

BRXA_FORMATTER
An 8- byte character field to be used by the bridge exit to specify the name of a user replaceable program to be used as a formatter. If a program name is specified in this field, then the it is called for all BMS, terminal, and interval control requests. The bridge exit is only called for XM, SYNC and MSG requests.

BRXA_CALL_EXIT_FOR_SYNCPOINT
A 1-character field that tells CICS whether or not to call the bridge exit for SYNCPOINT requests.

If this field is set to ’Y’ (BRXA_YES), then the bridge exit is called; if it is set to ’N’ (BRXA_NO), then the bridge exit is not called.

This value can only be set in the INIT call.

BRXA_NEXTTRANID_SOURCE
A 1-character field that indicates how the next transaction was specified. This indicator can have three settings:

BRXA_IMMEDIATE
The next TRANSID value came from a RETURN IMMEDIATE command.

BRXA_STARTED
The next TRANSID value came from a START TERMID command.

BRXA_NORMAL
The next TRANSID value came from a RETURN TRANSID or SET TERMINAL/NETNAME command.

BRXA_BRDATA_PTR
A fullword (4-byte) field that contains the address of the data specified by the BRDATA parameter on the START TRANSID BREXIT command.

BRXA_BRDATA_LEN
A fullword (4-byte) field that contains the length of the data specified by the BRDATA parameter on the START TRANSID BREXIT command.

BRXA_INTERVAL
A 4-character field containing the INTERVAL value specified by the user transaction on a START for its bridge facility.

BRXA_TIME
A 4-character field containing the TIME value specified by the user transaction on a START for its bridge facility.

BRXA_HOURS
A fullword (4-byte) field containing the HOURS value specified by the user transaction on a START for its bridge facility.

BRXA_MINUTES
A fullword (4-byte) field containing the MINUTES value specified by the user transaction on a START for its bridge facility.

BRXA_SECONDS
A fullword (4-byte) field containing the SECONDS value specified by the user transaction on a START for its bridge facility.

BRXA_START_AFTER
A 1-character field containing the AFTER value specified by the user transaction on a START for its bridge facility.

BRXA_START_AT
A 1-character field containing the AT value specified by the user transaction on a START for its bridge facility.

BRXA command area

The command area contains information relating to the command that has caused the bridge exit to be called. Common fields appear in the first **common** section of the command area, and fields specific to a particular command, or group of commands, follow.

The following diagrams show the various fields in the BRXA_COMMAND_AREA, and the commands for which they are valid.

An 'I' in the table indicates that the field is an input parameter to the exit; it is set by CICS before passing control to the exit, and any changes to the value made by the exit are ignored by CICS.

An 'O' in the table indicates that the field is an output parameter from the exit; the exit must set this value before return (unless the default value is acceptable), because CICS uses the value in completing the command.

A '-' or a blank in the table indicates that the field is not applicable to that command. The values on entry to the exit are undefined, and CICS ignores any value set in the field by the exit.

Table 9. BRXA command area field usage

Field name	SEND	RECEIVE	CONVERSE	ISSUE ERASEAUP	ISSUE DISCONNECT	FREE	SEND MAP	SEND CONTROL	SEND TEXT	RECEIVE MAP	RECV PARTN	RETRIEVE	Syncpoint
brxa_command_common													
brxa_function_code	I	I	I	I	I	I	I	I	I	I	I	I	I
brxa_command_code	I	I	I	I	I	I	I	I	I	I	I	I	I
brxa_user_abend_code	O	O	O	O	O	O	O	O	O	O	O	O	O
brxa_from_ptr	I	-	I	-	-	-	I	-	I	-	-	-	-

Table 9. BRXA command area field usage (continued)

Field name	SEND	RECEIVE	CONVERSE	ISSUE ERASEAUP	ISSUE DISCONNECT	FREE	SEND MAP	SEND CONTROL	SEND TEXT	RECEIVE MAP	RECV PARTN	RETRIEVE	Syncpoint
brxa_from_len	I	-	I	-	-	-	I	-	I	-	-	-	-
brxa_into_ptr	-	O	O	-	-	-	-	-	-	O	O	O	-
brxa_into_len	-	O	O	-	-	-	-	-	-	O	O	O	-
brxa_resp	O	O	O	O	O	O	O	O	O	O	O	O	O
brxa_resp2	O	O	O	O	O	O	O	O	O	O	O	O	O
brxa_cposn	-	O	O	-	-	-	-	-	-	O	O	-	-
brxa_aid	-	O	O	-	-	-	-	-	-	O	O	-	-
brxa_erase_indicator	I	-	I	-	-	-	I	I	I	-	-	-	-
brxa_last_indicator	I	-	I	-	-	-	I	I	I	-	-	-	-
brxa_wait_indicator	I	-	I	I	-	-	I	I	I	-	-	-	-
brxa_tc_command													
brxa_ctlchar	I	-	I	-	-	-							
brxa_buffer_indicator	-	I	-	-	-	-							
brxa_strfield_indicator	I	-	I	-	-.	-							
brxa_defresp_indicator	I	-	I	-	-	-							
brxa_invite_indicator	I	-	-	-	-	-							
brxa_bms_command													
brxa_mapset							I	-	-	I	-		
brxa_map							I	-	-	I	-		
brxa_ads_descriptor_ptr							I	-	-	I	-		
brxa_cursor							I	I	I	-	-		
brxa_msr_data							I	I	I	-	-		
brxa_data_indicator							I	-	-	-	-		
brxa_eraseaup_indicator							I	I	-	-	-		
brxa_freekb_indicator							I	I	I	-	-		
brxa_alarm_indicator							I	I	I	-	-		
brxa_msr_indicator							I	I	I	-	-		
brxa_frset_indicator							I	I	-	-	-		
brxa_text_type							-	-	I	-	-		
brxa_ic_command													
brxa_rtermid												O	
brxa_rtransid												O	
brxa_queue												O	
brxa_sync_command													

Table 9. BRXA command area field usage (continued)

Field name	SEND	RECEIVE	CONVERSE	ISSUE ERASEAUP	ISSUE DISCONNECT	FREE	SEND MAP	SEND CONTROL	SEND TEXT	RECEIVE MAP	RECV PARTN	RETRIEVE	Syncpoint
brxa_explicit													I
brxa_rollback													I

BRXA command area - common

Offset Hex	Type	Len	Name
(0)	STRUCTURE	48	BRXA_COMMAND_COMMON
(0)	CHARACTER	8	BRXA_COMMAND_AREA_EYECATCHER
(8)	CHARACTER	2	BRXA_FUNCTION_CODE
(A)	CHARACTER	2	BRXA_COMMAND_CODE
(C)	CHARACTER	4	BRXA_USER_ABEND_CODE
(10)	ADDRESS	4	BRXA_FROM_PTR
(14)	FULLWORD	4	BRXA_FROM_LEN
(18)	ADDRESS	4	BRXA_INTO_PTR
(1C)	FULLWORD	4	BRXA_INTO_LEN
(20)	FULLWORD	2	BRXA_RESP
(22)	HALFWORD	2	BRXA_RESP2
(24)	HALFWORD	2	BRXA_CPOSN
(26)	CHARACTER	1	BRXA_AID
(27)	CHARACTER	1	BRXA_ERASE_INDICATOR
(28)	CHARACTER	1	BRXA_LAST_INDICATOR
(29)	CHARACTER	1	BRXA_WAIT_INDICATOR
(2A)	CHARACTER	1	BRXA_FMT_RESPONSE
(2B)	CHARACTER	1	BRXA_READ_NOWAIT_ISSUED
(2C)	CHARACTER	1	BRXA_REQUEST_NEXT_ISSUED
(2D)	CHARACTER	3	(reserved)

BRXA_COMMAND_AREA_EYECATCHER

An eye-catcher to identify the area as a bridge command area. This is set by CICS, before passing control to the bridge exit, to the value BRXA_COMMAND_AREA_EYE ('>BRCOMMA'), which is defined in the DFHBRACx copy book.

BRXA_FUNCTION_CODE

A 2-character code identifying the CICS function for which the bridge exit was called. For calls before and after the user transaction runs, this is '00'. For all other requests, this is the 2-digit value in the first byte of EIBFN converted to character form. Valid EBCDIC characters are used for the function and command code to simplify testing of the values in user transaction exit programs written in all the supported languages, and to simplify passing of the codes to other systems. A constant with a meaningful name is provided for all the supported languages to simplify testing. The value is:

```
BRXA_XM
BRXA_TC
BRXA_IC
BRXA_SYNC
BRXA_BMS
BRXA_MSG
```

BRXA_COMMAND_CODE

A two-character code identifying the CICS command for which the bridge exit was called. For transaction initialization this is ’00’, for transaction bind this is ’02’, for transaction termination this is ’04’, and for transaction abend this is ’06’. For all other requests, this is the value in the second byte of EIBFN converted to character form.

See the *CICS Application Programming Reference* for information about EIBFN values. Valid EBCDIC characters are used for the function and command code to simplify testing of the values in user transaction exit programs written in all the supported languages, and to simplify passing of the codes to other systems. Constants with meaningful names are provided for all the supported languages to simplify testing. The values are:

```
BRXA_INIT
BRXA_BIND
BRXA_TERM
BRXA_ABEND
*   tc
BRXA_RECEIVE
BRXA_SEND
BRXA_CONVERSE
BRXA_ISSUE_DISCONNECT
BRXA_ISSUE_ERASEAUP
BRXA_FREE
*   bms
BRXA_RECEIVE_MAP
BRXA_SEND_MAP
BRXA_SEND_TEXT
BRXA_SEND_CONTROL
*   ic
BRXA_RETRIEVE
*   sync
BRXA_SYNCPOINT
*   msg
BRXA_READ_MESSAGE_NOWAIT
BRXA_READ_MESSAGE_WAIT
BRXA_WRITE_MESSAGE
```

BRXA_USER_ABEND_CODE

The abend code. CICS initializes this value to blanks. If the exit changes it to any other value, CICS generates a transaction abend with this code.

BRXA_FROM_PTR

The address of the FROM data in SEND, CONVERSE, SEND MAP, SEND TEXT, and START commands. This is zero for other commands, or if FROM is not specified on the command.

BRXA_FROM_LEN

The length of the FROM data in SEND, CONVERSE, SEND MAP, SEND TEXT, and START commands. This is zero for other commands, or if FROM is not specified on the command.

BRXA_INTO_PTR

The address of the INTO data in RECEIVE, CONVERSE, RECEIVE MAP and RETRIEVE commands. This must be set by the bridge exit, and CICS copies data from this address into the INTO area specified on the command, or copies the address into the SET parameter specified on the command.

Note: The exit must GETMAIN storage for INTO input because local storage could be reused on return from the bridge exit.

BRXA_INTO_LEN
The length of the INTO data in RECEIVE, CONVERSE, RECEIVE MAP, and RETRIEVE commands. This must be set by the user transaction exit, and CICS copies this value into the LENGTH, FLENGTH, or INTOLENGTH parameter specified on the command, and uses the value when copying data into the INTO area.

Note: CONVERSE is the only command which has both FROM and INTO, and the BRXA_FROM_PTR and BRXA_INTO_PTR (and corresponding lengths) could be replaced by a single BRXA_DATA_PTR (and BRXA_DATA_LEN). For CONVERSE, the exit replaces the FROM address and length by the INTO address and length.

BRXA_RESP
The resp code to be set (by CICS) in EIBRESP. This will be set to zero by CICS before calling the exit, and the exit must set this value if anything other than a normal response is required. CICS will generate an ABRN transaction abend if the value returned is not one that could normally be produced by CICS for this command.

If this value is zero on return, CICS may itself set the EIBRESP value and raise a condition.

BRXA_RESP2
The RESP2 code CICS returns and stores in EIBRESP2. This is set to zero by CICS before calling the exit, and the exit must change this value if anything other than a normal response is required.

CICS does not check the value specified for consistency with the command. If this value is zero on return, CICS may itself set the EIBRESP2 value and raise a condition.

BRXA_CPOSN
The cursor position to be set (by CICS) in EIBCPOSN for RECEIVE, CONVERSE, and RECEIVE MAP commands. This is set to zero by CICS before calling the exit, and the exit must change this value if the user transaction uses the value in EIBCPOSN.

BRXA_AID
The attention identifier (PF key code) to be set (by CICS) in EIBAID for RECEIVE, CONVERSE, and RECEIVE MAP commands. This is set to ENTER (X’7D’) by CICS before calling the exit, and the exit must change this value if the user transaction expects another value in EIBAID. The exit can use the values defined in DFHAID copy books to set the value (these are EBCDIC values of the 3270 AID characters).

BRXA_ERASE_INDICATOR
A 1-character value which is set (by CICS) to indicate whether ERASE, ERASE ALTERNATE, or ERASE DEFAULT is specified on SEND, CONVERSE SEND MAP, SEND TEXT, or SEND CONTROL commands. Constants with meaningful names are provided for all languages to allow the bridge exit to test this value if necessary. The values are:

```
BRXA_ERASE
BRXA_ERASE_ALTERNATE
BRXA_ERASE_DEFAULT
```

BRXA_LAST_INDICATOR

A 1-character field indicating whether LAST is specified on a SEND command. Valid values are 'Y' or 'N'; the following constants are provided for the exit to test this field:

```
BRXA_YES
BRXA_NO
```

BRXA_WAIT_INDICATOR

A one-character field indicating whether WAIT is specified on a SEND command, or on a RETRIEVE command. Valid values are 'Y' or 'N'; the following constants are provided for the exit to test this field:

```
BRXA_YES
BRXA_NO
```

BRXA_FMT_RESPONSE

This field is used by the formatter to tell the CICS that the bridge exit should be called to read or write a message. Possible values are:

BRXA_FMT_NONE

No action. The formatter has processed the request.

BRXA_FMT_OUTPUT_BUFFER_FULL

There is no room to add the next vector. Call the bridge exit to write the message, clear the buffer, then call the formatter again.

BRXA_FMT_WRITE_MESSAGE

The request required data to be flushed. Call the bridge exit to write the message.

BRXA_FMT_READ_MESSAGE_NOWAIT

The formatter has processed the data in the message. Check to see if there is a new message before requesting any further input. Call the bridge exit to read a message, then call the formatter again.

BRXA_FMT_REQUEST_NEXT_MESSAGE

The formatter has processed the data in the message. Call the bridge exit to read a message, then call the formatter again.

BRXA_READ_NOWAIT_ISSUED

This field is used by the formatter to check if it has already returned a BRXA_FMT_READ_MESSAGE_NOWAIT for this command. Possible values are:

BRXA_NO

BRXA_FMT_READ_MESSAGE_NOWAIT has not been returned for this command.

BRXA_YES

BRXA_FMT_READ_MESSAGE_NOWAIT has been returned for this command.

BRXA_REQUEST_NEXT_ISSUED

This field is used by the formatter to check if it has already returned a BRXA_FMT_REQUEST_NEXT_MESSAGE for this command. Possible values are:

BRXA_NO

BRXA_FMT_REQUEST_NEXT_MESSAGE has not been returned for this command.

BRXA_YES
BRXA_FMT_REQUEST_NEXT_MESSAGE has been returned for this command.

BRXA command area - terminal control

The terminal control command area defines some terminal control specific parameters.

Commands supported are SEND, RECEIVE, and CONVERSE.

Offset Hex	Type	Len	Name
(0)	STRUCTURE	53	BRXA_TC_COMMAND
(0)	common	48	
(2C)	CHARACTER	1	BRXA_CTLCHAR
(2D)	CHARACTER	1	BRXA_BUFFER_INDICATOR
(2E)	CHARACTER	1	BRXA_STRFIELD_INDICATOR
(2F)	CHARACTER	1	BRXA_DEFRESP_INDICATOR
(30)	CHARACTER	1	BRXA_INVITE_INDICATOR

BRXA_CTLCHAR
The 3270 Write Control Character (WCC) passed on SEND and CONVERSE commands as CTLCHAR. If not specified on the command, the default value (X'C3'- unlock keyboard, reset MDT flags) is passed to the exit.

BRXA_BUFFER_INDICATOR
A 1-character field indicating whether BUFFER was specified on a RECEIVE command. Valid values are 'Y' or 'N'; the following constants are provided for the exit to test this field:

```
BRXA_YES
BRXA_NO
```

BRXA_STRFIELD_INDICATOR
A 1-character field indicating whether STRFIELD was specified on a SEND or CONVERSE command. Valid values are 'Y' or 'N'; the following constants are provided for the exit to test this field:

```
BRXA_YES
BRXA_NO
```

BRXA_DEFRESP_INDICATOR
A 1-character field indicating whether DEFRESP was specified on a SEND or CONVERSE command. Valid values are 'Y' or 'N'; the following constants are provided for the exit to test this field:

```
BRXA_YES
BRXA_NO
```

BRXA_INVITE_INDICATOR
A 1-character field indicating whether INVITE was specified on a SEND command. Valid values are 'Y' or 'N'; the following constants are provided for the exit to test this field:

```
BRXA_YES
BRXA_NO
```

BRXA command area - BMS

The BMS command interface defines some BMS specific parameters.

Commands supported are SEND MAP, SEND TEXT, SEND CONTROL, and RECEIVE MAP.

Offset Hex	Type	Len	Name
(0)	STRUCTURE	77	BRXA_BMS_COMMAND
(0)	common	44	
(2C)	CHARACTER	7	BRXA_MAPSET
(33)	CHARACTER	1	(reserved)
(34)	CHARACTER	7	BRXA_MAP
(3B)	CHARACTER	1	(reserved)
(3C)	ADDRESS	4	BRXA_ADS_DESCRIPTOR_PTR
(40)	HALFWORD	2	BRXA_CURSOR
(42)	CHARACTER	4	BRXA_MSR_DATA
(46)	CHARACTER	1	BRXA_DATA_INDICATOR
(47)	CHARACTER	1	BRXA_ERASEAUP_INDICATOR
(48)	CHARACTER	1	BRXA_FREEKB_INDICATOR
(49)	CHARACTER	1	BRXA_ALARM_INDICATOR
(4A)	CHARACTER	1	BRXA_FRSET_INDICATOR
(4B)	CHARACTER	1	BRXA_MSR_INDICATOR
(4C)	CHARACTER	1	BRXA_TEXT_TYPE

BRXA_MAPSET
The (unsuffixed) mapset name specified on SEND MAP or RECEIVE MAP.

BRXA_MAP
The map name specified on SEND MAP or RECEIVE MAP.

BRXA_ADS_DESCRIPTOR_PTR
The address of the ADS descriptor for BMS SEND MAP and RECEIVE MAP commands. This is set by CICS, if the bridge exit has set the flag indicating that the descriptor should be loaded, and if the relevant mapset has been reassembled under CICS Transaction Server for VSE/ESA Release 1 to include the descriptor. Otherwise this pointer is set to 0.

BRXA_CURSOR
A halfword value containing the CURSOR position specified on SEND MAP, SEND TEXT, or SEND CONTROL command, which identifies where the cursor is to be positioned on the 3270 screen. A value of -1 is passed if the application specified CURSOR with no value on SEND MAP command, indicating that symbolic cursor positioning is required, that is, that the cursor is to be positioned in the first field in the application data structure that has a value of -1 in the corresponding length field. A value of -2 is passed if the application did not specify CURSOR on the SEND MAP command.

BRXA_MSR_DATA
The 4-character value specified in MSR on a SEND MAP, SEND CONTROL, or SEND TEXT command. Constants are provided in the copy book DFHMSRCA that allow the exit to test the values specified.

Note: If you assume that a BFB will always be constructed as if its TYPETERM were defined with MSRCONTROL(NO), then this parameter could be omitted, because BMS ignores the MSR field specified on the command for a 3270 terminal for which MSRCONTROL(NO) is specified.

BRXA_DATA_INDICATOR
A 1-character field indicating whether DATAONLY, MAPONLY, or neither is

specified on the SEND MAP command. Valid values are ’D’ (DATAONLY), ’M’ (MAPONLY) or ’N’ (neither specified); the following constants are provided for the exit to test this field:

```
BRXA_DATAONLY
BRXA_MAPONLY
BRXA_NEITHER
```

(Note that if MAPONLY is specified, the FROM pointer and length are zero, because there is no application data structure in this case.)

BRXA_ERASEAUP_INDICATOR
A 1-character field indicating whether ERASEAUP is specified on a SEND MAP or SEND CONTROL command. Valid values are ’Y’ or ’N’; the following constants are provided for the exit to test this field:

```
BRXA_YES
BRXA_NO
```

BRXA_FREEKB_INDICATOR
A 1-character field indicating whether FREEKB is specified on a SEND MAP, SEND TEXT, or SEND CONTROL command. Valid values are ’Y’ or ’N’; the following constants are provided for the exit to test this field:

```
BRXA_YES
BRXA_NO
```

BRXA_ALARM_INDICATOR
A 1-character field indicating whether ALARM is specified on a SEND MAP, SEND TEXT, or SEND CONTROL command. Valid values are ’Y’ or ’N’; the following constants are provided for the exit to test this field:

```
BRXA_YES
BRXA_NO
```

BRXA_MSR_INDICATOR
A 1-character field indicating whether MSR is specified on a SEND MAP, SEND TEXT, or SEND CONTROL command. Valid values are ’Y’ or ’N’; the following constants are provided for the exit to test this field:

```
BRXA_YES
BRXA_NO
```

BRXA_FRSET_INDICATOR
A 1-character field indicating whether FRSET is specified on a SEND MAP or SEND CONTROL command. Valid values are ’Y’ or ’N’; the following constants are provided for the exit to test this field:

```
BRXA_YES
BRXA_NO
```

BRXA_TEXT_TYPE
A 1-character field indicating whether NOEDIT or MAPPED is specified on a SEND TEXT command. Valid values are blank (neither NOEDIT nor MAPPED specified), ’N’ (NOEDIT specified) and ’M’ (MAPPED specified); the following constants are provided for the exit to test this field:

```
BRXA_TEXT_NORMAL
BRXA_TEXT_MAPPED
BRXA_TEXT_NOEDIT
```

BRXA command area - interval control

The interval control command area defines some interval control specific parameters.

The only command supported is RETRIEVE.

Offset Hex	Type	Len	Name
(0)	STRUCTURE	60	BRXA_IC_COMMAND
(0)	common	44	
(2C)	CHARACTER	4	BRXA_RTERMID
(30)	CHARACTER	4	BRXA_RTRANSID
(34)	CHARACTER	8	BRXA_QUEUE

BRXA_RTERMID
The value of RTERMID specified on a START command. For the RETRIEVE command, this is a field that the bridge exit can set to pass the RTERMID value back to the application issuing the RETRIEVE.

BRXA_RTRANSID
The value of RTRANSID specified on a START command. For the RETRIEVE command, this is a field that the bridge exit can set to pass the RTRANSID value back to the application issuing the RETRIEVE.

BRXA_QUEUE
The value of QUEUE specified on START command. For the RETRIEVE command this is a field in which the bridge exit can set the QUEUE value to be used by the application issuing the RETRIEVE,

BRXA command area - syncpoint

The syncpoint command area defines actions at SYNCPOINT and SYNCPOINT ROLLBACK. BRXA_EXPLICIT is used to indicate that this request originated from an explicit EXEC CICS SYNCPOINT command, or that it is an implicit syncpoint generated by CICS. It is set to 'Y' or 'N' before the exit is invoked; the following constants are provided for the exit to test this field:

```
BRXA_YES
BRXA_NO
```

Valid values for BRXA_ROLLBACK are 'Y' or 'N'; the following constants are provided for the exit to test this field:

```
BRXA_YES
BRXA_NO
```

Offset Hex	Type	Len	Name
(0)	STRUCTURE	46	BRXA_SYNC_COMMAND
(0)	common	44	
(2C)	CHARACTER	1	BRXA_EXPLICIT
(2D)	CHARACTER	1	BRXA_ROLLBACK

BRXA command area - MSG

This command area defines actions when the bridge exit is called to read or write a message. These functions are only used if the bridge exit specified a formatter on initialization.

This command area defines actions at initialization. Relevant fields are in the common part of the command area. The layout of the MSG section of the command area is :

Offset Hex	Type	Len	Name
(0)	STRUCTURE	48	BRXA_MSG_COMMAND

ADS descriptor area

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 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 (using CICS Transaction Server for VSE/ESA Release 1.1.1 or a later release) to include the descriptor, and CICS attempts to locate the descriptor only if the BRXA_LOAD_ADS_DESCRIPTOR indicator is set to BRXA_YES in the bridge exit initialization call.

The ADS descriptor is created by the BMS macros in either a ***long*** or a ***short*** form. Long data has the same content as short data, but the fields are aligned on 4-byte boundaries to support those transport mechanisms that require all message fields to be word aligned. nly the long form is supported in the C language. The ADS descriptor is defined below in both short and long format.

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.

Short form:

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. This is needed to determine the HTML template corresponding to the map.

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 each field of the ADS; 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 ADSD_FIRST_FIELD can be used as the initial value of the pointer for the field descriptor (unless ADSD_FIELD_COUNT is 0).

Long form:

Offset	Type	Len	Name
Hex			
(0)	STRUCTURE	60	ADS_LONG_DESCRIPTOR
(0)	FULLWORD	42	ADSDL_LENGTH
(4)	CHARACTER	4	ADSDL_EYECATCHER
(8)	FULLWORD	4	ADSDL_MAP_INDEX
(C)	FULLWORD	4	ADSDL_FIELD_COUNT
(10)	FULLWORD	4	ADSDL_STRUCTURE_LENGTH
(14)	FULLWORD	4	ADSDL_ATTRIBUTE_NUMBER
(18)	CHARACTER	12	ADSDL_ATTRIBUTE_TYPE_CODES
(24)	CHARACTER	1	ADSDL_MAP_JUSTIFY_HOR
(25)	CHARACTER	1	ADSD_MAP_JUSTIFY_VER
(26)	CHARACTER	2	reserved
(28)	FULLWORD	4	ADSDL_MAP_STARTING_LINE
(2C)	FULLWORD	4	ADSDL_MAP_STARTING_COLUMN
(30)	FULLWORD	4	ADSDL_MAP_LINES
(34)	FULLWORD	4	ADSDL_MAP_COLUMNS
(38)	CHARACTER	1	ADSDL_WRITE_CONTROL_CHARACTER
(39)	CHARACTER	3	(reserved)
(3C)	STRUCTURE	*	ADSDL_FIRST_FIELD

ADSDL_LENGTH
The length of the ADS descriptor.

ADSDL_EYECATCHER
An eye-catcher (’ADSL’) to identify this as an ADS descriptor.

ADSDL_MAP_INDEX
The index number of the map within the mapset. This is needed to determine the HTML template corresponding to the map.

ADSDL_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.

ADSDL_STRUCTURE_LENGTH
The length of the short application data structure.

ADSDL_ATTRIBUTE_NUMBER
The number of extended attributes in each field of the ADS; that is, the number of attributes specified in DSATTS in the map definition.

ADSDL_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

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

ADSDL_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).

ADSDL_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.)

ADSDL_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.)

ADSDL_MAP_LINES
The number of lines in the map from the ’SIZE=’ operand.

ADSDL_MAP_COLUMNS
The number of columns in the map from the ’SIZE=’ operand.

ADSDL_WRITE_CONTROL_CHAR
The 3270 encoded WCC derived from the ’CONTROL=’ operand.

ADSDL_FIRST_FIELD
The first field descriptor. The address of ADSD_FIRST_FIELD can be used as the initial value of the pointer for the field descriptor (unless ADSD_FIELD_COUNT is 0).

ADS field descriptor

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

Short form:

Offset Hex	Type	Len	Name
(0)	STRUCTURE	42	ADS_FIELD_DESCRIPTOR
(0)	CHARACTER	32	ADSD_FIELD_NAME
(20)	HALFWORD	2	ADSD_FIELD_NAME_LEN
(22)	HALFWORD	2	ADSD_OCCURS_INDEX
(24)	HALFWORD	2	ADSD_FIELD_OFFSET
(26)	HALFWORD	2	ADSD_FIELD_DATA_LEN
(28)	CHARACTER	1	ADSD_FIELD_JUSTIFY
(29)	CHARACTER	1	ADSD_FIELD_FILL_CHAR
(2A)	CHARACTER	*	ADSD_NEXT_FIELD

ADSD_FIELD_NAME
The unsuffixed field name padded with blanks.

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 ADSDL. The offset is to the beginning of the (fullword) length field, and you must add 4 (for the length field) + 4 (for the 3270 attribute) + 8 fullwords for the extended attributes 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.

Long form:

Offset	Type	Len	Name
Hex			
(0)	STRUCTURE	52	ADS_LONG_FIELD_DESCRIPTOR
(0)	CHARACTER	32	ADSDL_FIELD_NAME
(20)	FULLWORD	4	ADSDL_FIELD_NAME_LEN
(24)	FULLWORD	4	ADSDL_OCCURS_INDEX
(28)	FULLWORD	4	ADSDL_FIELD_OFFSET
(2C)	FULLFWORD	4	ADSDL_FIELD_DATA_LEN
(30)	CHARACTER	1	ADSDL_FIELD_JUSTIFY
(31)	CHARACTER	1	ADSDL_FIELD_FILL_CHAR
(32)	CHARACTER	2	reserved
(34)	CHARACTER	*	ADSD_NEXT_FIELD

ADSD_LONG_FIELD_NAME
The unsuffixed field name padded with blanks.

ADSDL_FIELD_NAME_LEN
The number of characters in the field name.

ADSDL_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.

ADSDL_FIELD_OFFSET
The offset of the field within the ADS. The offset is to the beginning of the (halfword) length field, and you must add 2 (for the length field) + 1 (for the 3270 attribute) + attribute_number (for the extended attributes specified in DSATTS) to obtain the offset of the data part of the field.

ADSDL_FIELD_DATA_LEN
The length of the field in the ADS.

ADSDL_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.

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

ADSDL_NEXT_FIELD
The next field descriptor. The address of ADSD_NEXT_FIELD can be used to update a pointer for the field descriptor.

Supplied copybooks

The names of the parameters and constants, translated into appropriate forms for the different programming languages supported, are defined in files supplied as part of the 3270 bridge. The files or copybooks for the various languages are listed in the following table.

Table 10. BRXA copybooks

Language	Area definition	Area constants
Assembler	DFHBRARD	DFHBRACD
C	DFHBRARH	DFHBRACH
PL/I	DFHBRARL	DFHBRACL
COBOL	DFHBRARO	DFHBRACO

Copybook example (DFHBRACD)

The following constants are provided for use by an assembler language bridge exit to set and interpret values in the BRXA.

API function codes are character equivalent constants of the first byte of the EIBFN values documented in the *CICS Application Programming Reference* manual.

```
 * function codes
BRXA_XM   EQU   C'00'
BRXA_TC   EQU   C'04'
BRXA_IC   EQU   C'10'
BRXA_SYNC EQU   C'16'
BRXA_BMS  EQU   C'18'
BRXA_MSG  EQU   C'01'
```

```
* indicator values
BRXA_YES             EQU   C'Y'
BRXA_NO              EQU   C'N'
BRXA_ERASE           EQU   C'E'
BRXA_ERASE_ALTERNATE EQU   C'A'
BRXA_ERASE_DEFAULT   EQU   C'D'
BRXA_DATAONLY        EQU   C'D'
BRXA_MAPONLY         EQU   C'M'
BRXA_NEITHER         EQU   C'N'
BRXA_TEXT_NORMAL     EQU   C' '
BRXA_TEXT_MAPPED     EQU   C'M'
BRXA_TEXT_NOEDIT     EQU   C'N'
BRXA_IMMEDIATE       EQU   C'I'
BRXA_STARTED         EQU   C'S'
BRXA_NORMAL          EQU   C'B'
* Start codes
BRXA_START           EQU   C'S '
BRXA_STARTDATA       EQU   C'SD'
BRXA_TERMINPUT       EQU   C'TD'
```

```
* formatter response values
BRXA_FMT_NONE                 EQU   C'N'
BRXA_FMT_OUTPUT_BUFFER_FULL   EQU   C'F'
BRXA_FMT_WRITE_MESSAGE        EQU   C'W'
BRXA_FMT_REQUEST_NEXT_MESSAGE  EQU   C'Q'
BRXA_FMT_READ_MESSAGE_NOWAIT   EQU   C'R'
```

API command codes are character equivalent constants of the second byte of the EIBFN values documented in the *CICS Application Programming Reference* manual.

```
*   xm
BRXA_INIT            EQU  C'00'
BRXA_BIND            EQU  C'02'
BRXA_TERM            EQU  C'04'
BRXA_ABEND           EQU  C'06'
*   tc
BRXA_RECEIVE         EQU  C'02'
BRXA_SEND            EQU  C'04'
BRXA_CONVERSE        EQU  C'06'
BRXA_ISSUE_DISCONNECT EQU  C'14'
BRXA_ISSUE_ERASEAUP   EQU  C'18'
BRXA_FREE            EQU  C'22'
*   bms
BRXA_RECEIVE_MAP     EQU  C'02'
BRXA_SEND_MAP        EQU  C'04'
BRXA_SEND_TEXT       EQU  C'06'
BRXA_SEND_CONTROL    EQU  C'12'
*   ic
BRXA_RETRIEVE        EQU  C'0A'
*   sync
BRXA_SYNCPOINT       EQU  C'02'
*   msg (new for CTS 1.3)
BRXA_READ_MESSAGE_NOWAIT   EQU C'02'
BRXA_READ_MESSAGE_WAIT     EQU C'04'
BRXA_WRITE_MESSAGE         EQU C'06'
```

Chapter 7. Problem determination

This chapter contains Diagnosis, Modification, or Tuning Information.

This chapter helps you debug problems in the CICS 3270 bridge exit user-replaceable program, the IBM-supplied parts of the CICS 3270 bridge, and in the system setup of the CICS 3270 bridge. If you suspect that you have a problem in another part of CICS, refer to the *CICS Problem Determination Guide*.

The 3270 user program should be tested with a real 3270 terminal before transferring to a bridge environment.

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
The CICS 3270 bridge provides CICS messages. These messages are listed in *CICS Messages and Codes*.

Trace The CICS 3270 bridge produces system trace entries containing all the important information required for problem diagnosis.

Dump Dump formatting is provided for data areas relating to the CICS 3270 bridge.

Abend codes
Transaction abend codes are standard 4-character names. The abend codes produced by the CICS 3270 bridge are listed in *CICS Messages and Codes*.

Troubleshooting

This section provides some hints on troubleshooting. 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. 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
   - Release of any other products providing transport mechanisms for the 3270 bridge, such as MQSeries.
   - LE/370 release
   - VSE/ESA release
2. When did the problem first occur?
3. What were you trying to accomplish at the time the problem occurred?
4. What changes were made to the system before the occurrence of the problem?
   - To the VSE/ESA system
   - To the bridge exit
   - To the CICS user program being accessed by the bridge
   - To the end-user program
   - To the transport mechanisms

   - To CICS Transaction Server
5. What is the problem?
   - Incorrect output
   - Hang/Wait: Use CEMT INQUIRE to display details of the transaction.
   - Loop: Use CEMT INQUIRE to display the details of the transaction.
   - Abend in the bridge exit
   - Abend in a CICS program
   - Abend in the IBM-supplied part of the CICS 3270 bridge
   - Performance problem
   - Storage violation
   - Logic Error
6. At what point in the processing did the problem occur?

Documentation about the problem

To investigate most problems, you must look at the dumps, traces, and logs provided with VSE and CICS.

- System Dump: This contains the CICS internal trace
- CICS auxiliary trace, if enabled
- TCP/IP for VSE trace, if relevant
- GTF trace, if enabled
- Console log
- CSMT log
- CICS job log

To identify which are likely to be useful for your problem, try to work out the area of the CICS 3270 bridge giving rise to the problem.

Using messages and codes

CICS 3270 bridge messages have identifiers of the form DFHBR*nnnn*, where *nnnn* are four numeric characters. These numbers indicate which component generated the message, as shown in *CICS Messages and Codes*.

When the CICS 3270 bridge issues a message as a result of an error, it also makes an exception trace entry. The CICS 3270 bridge also generates information messages, for instance during enable processing and disable processing.

CICS 3270 bridge messages are supplied in supported National Languages. The CICS message editing utility can be used to translate them into other languages supported by CICS.

Using Trace

The CICS 3270 bridge creates CICS system trace entries, which can be formatted using software supplied as part of CICS.

You can request level 2 tracing using SET TRACETYPE or the CETR supplied transaction. This gives a full trace of data being transmitted between the user transaction and the end-user application.

You should request level 2 tracing for the bridge by specifying BR in the SET TRACETYPE or CETR command. CICS sets BRXA_TRACE to ’Y’ if level 2 tracing is requested, but the bridge exit should create exception trace entries even if this flag is not set.

CICS trace output is described in the *CICS Problem Determination Guide*, and details of the contents of each trace points are given in the *CICS Trace Entries Handbook*.

Dump and trace formatting

To control dump formatting of CICS 3270 bridge data areas, you can change the parameters ofr DFHPD410 to include BR=0|1|2 dump formatting as follows:

The parameters have these meanings:

BR=0 Suppress system dump for the 3270 bridge.

BR=1 Produce system dump summary listing for the 3270 bridge.

BR=2 Produce system dump for the 3270 bridge.

BR=3 Produce system dump summary listing and a system dump for the 3270 bridge.

Details on specifying parameters and anlysing dumps are described in the *CICS Problem Determination Guide*.

CICS 3270 bridge output in the formatted dump consists of the major control blocks of the CICS 3270 bridge, with interpretation of some of the fields. The CICS 3270 bridge output can be found in the IPCS output by searching for `==BR`. It is under the heading `BRIDGE FACILITY SUMMARY`.

Debugging the bridge exit

The following aids are provided to help you resolve problems occurring in the bridge exit while bridging to a 3270 user transaction:

IDENTIFIER

The bridge exit constructs a 48-byte identifier field, containing information to aid problem determination. This can contain relevant fields taken from the START data. You can access the identifier with INQUIRE TASK, or CEMT INQUIRE TASK.

EDF

The CEDX transaction provides EDF for non-terminal tasks. This allows you to use EDF with the bridge exit (which is a non terminal task). You should issue CEDX against the bridge transaction to see the initialization call to the bridge exit, otherwise you should issue CEDX against the user transaction.

Note that bridge facilities are not EDF-able.

The supplied bridge exit program, DFH0CBRE, is defined with EDF disabled; you will need to modify this is you intend to use CEDX.

See the *CICS Supplied Transactions* for more information about the use of CEDX.

Trace

Trace records are written during execution of the bridge exit. See the *CICS Enhancements Guide* for a description of the trace entries.

Level 2 trace will show you the messages transmitted, so you can verify that the user transaction and end-user application are cooperating correctly.

You can also use the ENTER TRACENUM command in the bridge exit to write user records to the CICS trace.

Debugging the supplied bridge exit

The supplied bridge exit, DFH0CBRE, is very well commented. You are recommended to read the source carefully before use. The following aids are provided to help you resolve problems:

Abend codes and Trace

The supplied exit creates trace and exception trace entries during execution. It also returns some abend codes (ABXx). The bridge mechanism validates correct use of BRXA, and ABRx abends result from incorrect usage. These codes and the trace points are all fully documented in the comments within DFH0CBRE.

Note: If you change the supplied exit, you can change the ABR prefix.

You need to activate the level-2 tracing with SET TRACETYPE or CETR, specifying the BR component.

Message validation

If CICS detects an invalid character in an input message, the MQCIH-ERROROFFSET field is set to the offset of the invalid character, counted from the start of the message.

Part 3. External CICS Interface

This part of the book describes the external CICS interface.

This part contains:

- “Chapter 8. Introduction to the external CICS interface” on page 113
- “Chapter 9. The EXCI CALL interface” on page 119
- “Chapter 10. The EXEC CICS interface” on page 143
- “Chapter 11. Defining connections to CICS” on page 151
- “Chapter 12. The EXCI user-replaceable module” on page 157
- “Chapter 13. External CICS interface options table, DFHXCOPT” on page 159
- “Chapter 14. Compiling and link-editing external CICS interface client programs” on page 161
- “Chapter 15. Security” on page 169
- “Chapter 16. Problem determination” on page 173
- “Chapter 17. Response and reason codes returned on EXCI calls” on page 189
- “Chapter 18. Messages and Codes” on page 199.

Chapter 8. Introduction to the external CICS interface

This chapter gives a brief overview of the external CICS interface (EXCI), covering the following topics:

- "Overview"
- "Benefits of the external CICS interface" on page 117
- "EXCI" on page 10
- "Requirements for the external CICS interface" on page 117

Overview

The external CICS interface is an application programming interface that enables a non-CICS program (a client program) running in VSE to call a program (a server program) running in a CICS region and to pass and receive data by means of a communications area. The CICS application program is invoked as if linked-to by another CICS application program.

This programming interface allows a user to allocate and open sessions (or pipes[1]) to a CICS region, and to pass distributed program link (DPL) requests over them. The multiregion operation (MRO) facility of CICS interregion communication (IRC) facility supports these requests, and each pipe maps onto one MRO session, with a limit of 25 pipes per EXCI partition.

The client program and the CICS server region (the region where the server program runs or is defined) must be in the same VSE image.

A client program that uses the external CICS interface can operate multiple sessions for different users (either under the same or separate TCBs) all coexisting in the same VSE partition without knowledge of, or interference from, each other.

Where a client program attaches another client program, the attached program runs under its own VSE task.

The programming interfaces

The external CICS interface provides two forms of programming interface: the EXCI CALL interface and the EXEC CICS interface.

The EXCI CALL interface: This interface consists of six commands that allow you to:

- Allocate and open sessions to a CICS system from non-CICS programs running under VSE
- Issue DPL requests on these sessions from the non-CICS programs
- Close and deallocate the sessions on completion of the DPL requests.

The six EXCI commands are:

- Initialize-User
- Allocate_Pipe
- Open_Pipe
- DPL call

[1] pipe. A one-way communication path between a sending process and a receiving process. In an external CICS interface implementation, each pipe maps onto one MRO session, where the client program represents the sending process and the CICS server region represents the receiving process.

- Close_Pipe
- Deallocate_Pipe

The EXEC CICS interface: The external CICS interface provides a single, composite command—EXEC CICS LINK PROGRAM—that performs all six commands of the EXCI CALL interface in one invocation.

This command is similar but not identical to the distributed program link command of the CICS command-level application programming interface.

API restrictions for server programs

A CICS server program invoked by an external CICS interface request is restricted to the DPL subset of the CICS application programming interface. This subset (the DPL subset) of the API commands is the same as for a CICS-to-CICS server program.

See the *CICS Application Programming Guide* for details of the DPL subset for server programs.

Choosing between the EXEC CICS and the CALL interface

As illustrated in the various language versions of the CICS-supplied sample client program (see “Sample application programs” on page 162 for details), you can use both the CALL interface (all six commands) and the EXEC CICS LINK command in the same program, to perform separate requests. As a general rule, it is unlikely that you would want to do this in a production program.

Each form of the external CICS interface has its particular benefits.

- For low-frequency or single DPL requests, you are recommended to use the EXEC CICS LINK command.

  It is easier to code, and therefore less prone to programming errors.

  Note that each invocation of an EXEC CICS LINK command causes the external CICS interface to perform all the functions of the CALL interface, which results in unnecessary overhead.
- For multiple or frequent DPL requests from the same client program, you are recommended to use the EXCI CALL interface.

  This is more efficient, because you need only perform the Initialize_User and Allocate_Pipe commands once, at or near the beginning of your program, and the Deallocate_Pipe once on completion of all DPL activity. In between these functions, you can open and close the pipe as necessary, and while the pipe is opened, you can issue as many DPL calls as you want.

Illustrations of the external CICS CALL interface

The diagrams in Figure 25 on page 115 through Figure 28 on page 116 illustrate the external CICS interface using the EXCI CALL interface.

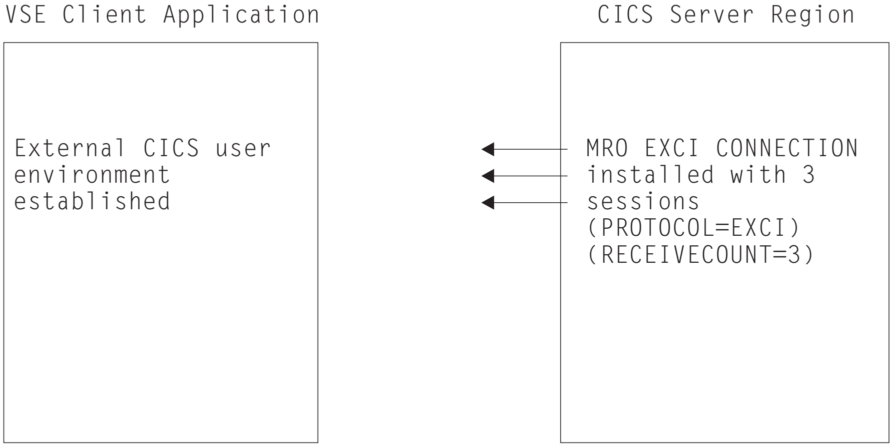

VSE Client Application
CICS Server Region
External CICS user environment established
MRO EXCI CONNECTION installed with 3 sessions (PROTOCOL=EXCI) (RECEIVECOUNT=3)

Figure 25. Stage 1: Status after an INITIALIZE_USER call

Notes:

1. In Figure 25, the target CICS region is running with IRC open, and one EXCI connection with three sessions installed, at the time the client application program issues an INITIALIZE_USER call.
2. The client application program address space is initialized with the EXCI user environment. There is no MRO activity at this stage, and no pipe exists.

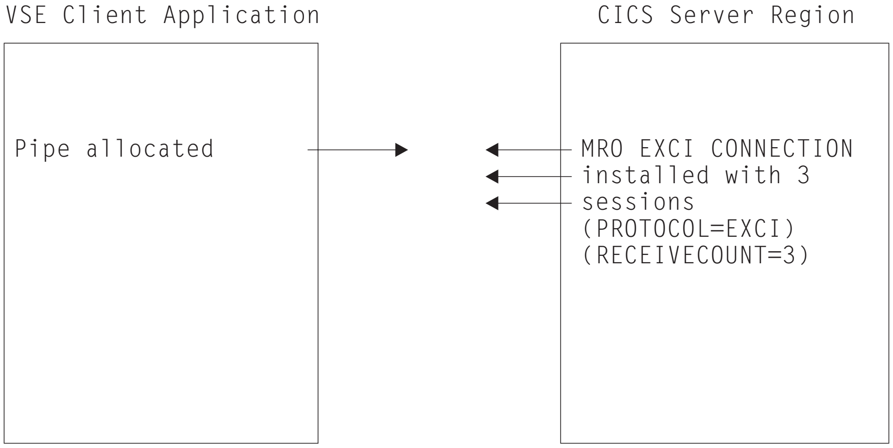

VSE Client Application
CICS Server Region
Pipe allocated
MRO EXCI CONNECTION installed with 3 sessions (PROTOCOL=EXCI) (RECEIVECOUNT=3)

Figure 26. Stage 2: Status after the first ALLOCATE_PIPE call

Note: In Figure 26, the external CICS interface logs on to MRO, identifying the target CICS server region.

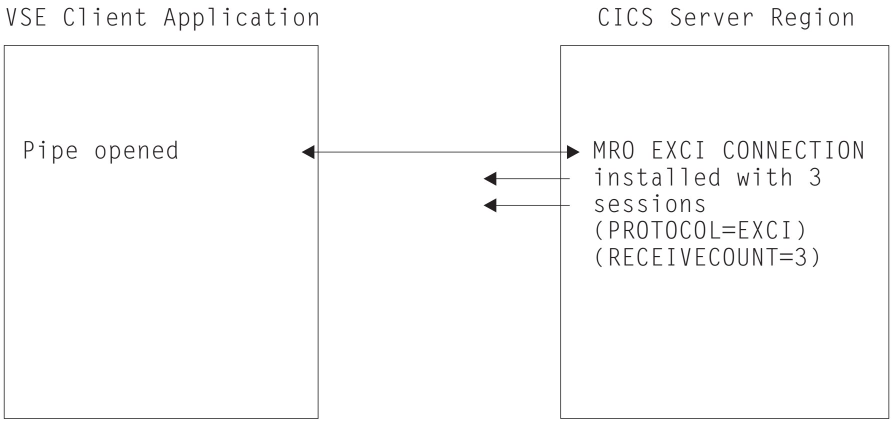


Figure 27. Stage 3: Status after the OPEN_PIPE call

Notes:

1. In Figure 27, the external CICS interface connects to the CICS server region, and the pipe is now available for use.
2. The remaining two EXCI sessions are free, and can be used by further open pipe requests from the same, or a different, client application program (provided the connection is generic).

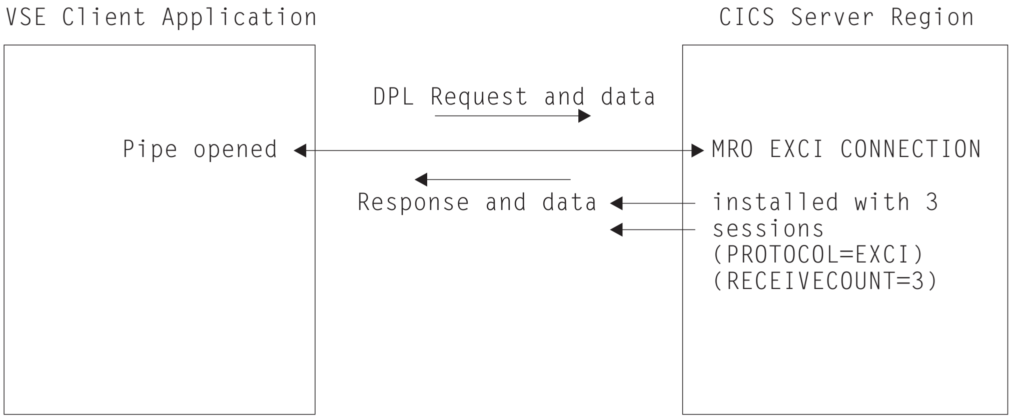


Figure 28. Stage 4: Status with one open pipe, processing a DPL call

Note: In Figure 28, the external CICS interface passes the DPL request over the open pipe, with any associated data. The CICS server region returns a response and data over the open pipe.

Closing pipes: When the client application program closes a pipe, it remains allocated ready for use by the same user, and the status is as shown in Figure 26 on page 115. At this stage, the MRO session is available for use by another open pipe request, from the same or from a different client application program (provided the connection is generic).

Deallocating pipes: When the client application program deallocates a pipe, it logs off from MRO and frees all the storage associated with the session. This leaves the status as shown in Figure 25 on page 115.

Illustration of the EXCI EXEC CICS interface

Figure 29 illustrates the EXEC CICS interface, and how it resolves to the six EXCI CALLs.

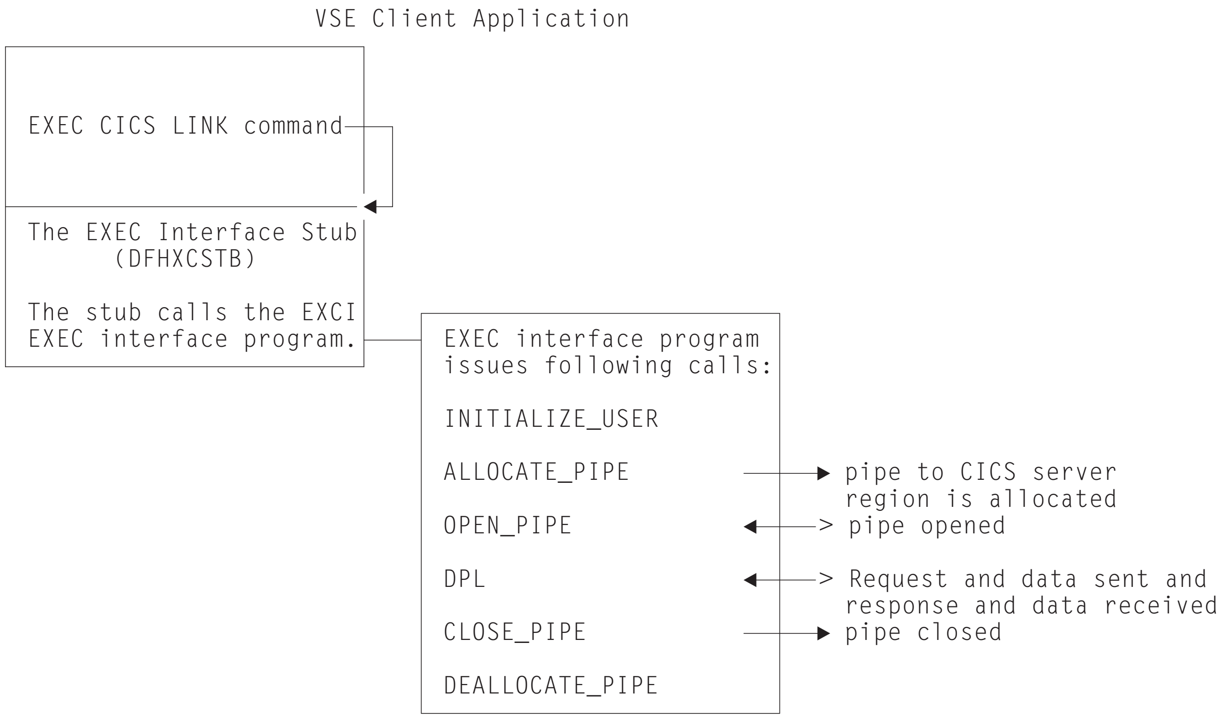

VSE Client Application
EXEC CICS LINK command
The EXEC Interface Stub (DFHXCSTB)
The stub calls the EXCI EXEC interface program.
EXEC interface program issues following calls:
INITIALIZE_USER
ALLOCATE_PIPE
OPEN_PIPE
DPL
CLOSE_PIPE
DEALLOCATE_PIPE
pipe to CICS server region is allocated
pipe opened
Request and data sent and response and data received
pipe closed

Figure 29. Illustration of the external CICS interface using the EXEC CICS command

Benefits of the external CICS interface

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

Programs running in VSE/ESA can issue an EXEC CICS LINK PROGRAM command, or use the EXCI CALL interface, to call a CICS application program running in a CICS server region.

The provision of this programming interface means that, for example, VSE programs can:

- Update resources with integrity while CICS is accessing them.
- Take CICS resources offline, and back online, at the start and end of a VSE 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).

Requirements for the external CICS interface

Client programs running in a VSE partition can communicate only with CICS server partitions running under CICS Transaction Server for VSE/ESA, Release 1 or a later.

Also, the client program can connect to the server CICS region only through the CICS Transaction Server for VSE/ESA Release 1, or later, interregion communication program, DFHIRP.

For information about DFHIRP, and its requirement to be installed in the VSE shared virtual area (SVA), see the *CICS System Definition Guide*.

Chapter 9. The EXCI CALL interface

The EXCI CALL interface consists of six commands that allow you to:

- Allocate and open sessions to a CICS system from non-CICS programs running under MVS
- Issue distributed program link (DPL) requests on these sessions from the non-CICS programs

The six EXCI commands are:

- Initialize_User
- Allocate_Pipe
- Open_Pipe
- DPL_Request
- Close_Pipe
- Deallocate_Pipe

The application program stub, DFHXCSTB: The EXCI commands invoke the external CICS interface via an application programming stub provided by CICS, called DFHXCSTB. You must include this stub when you link-edit your non-CICS program.

The CALL interface commands

In the description of each command that follows, the syntax box illustrates the assembler form of the command. The commands are also supported by C for VSE, COBOL for VSE, and PL/I for VSE programming languages, using the CALL conventions appropriate for the languages.

There are examples of these CALLs, in all the supported languages, in the sample client programs provided by CICS. See “Sample application programs” on page 162 for information about these.

Initialize_User

Function

Initialize the user environment including obtaining authority to use IRC facilities. The environment is created for the lifetime of the VSE task, so the command needs to be issued only once per user per VSE task. Further commands from this user must be issued under the same VSE task.

Note: A *user* is a program that has issued an Initialize_user request (or for which an Initialize_User request has been issued), with a unique name per VSE task. For example:

- A simple client program running under VSE can be a single user of the external CICS interface.
- A client program running under VSE can open several pipes and issue external CICS interface calls over them sequentially, on behalf of different vendor packages. In this case, from the viewpoint of the client program, each of the packages is a user, identified by a unique user name. Thus a single client program can operate on behalf of multiple users.
- A program running under VSE can attach several VSE subtasks, under each of which a vendor package issues external CICS interface calls on its own behalf. Each package is a client program in its own right, and runs under its own VSE subtask. Each is also a user, with a unique user name.

Syntax

CALL DFHXCIS

```
►►──CALL DFHXCIS,(version_number,──return_area,──user_token,──call_type,──────────►

►──user_name),──VL,MF=(E,(1))─────────────────────────────────────────────────────►◄
```

Parameters

version_number
A fullword binary input area indicating the version of the external CICS interface parameter list being used. It must be set to 1 in the client program.

The equated value for this parameter in the CICS-supplied copybook DFHXCPL*x* (where *x* indicates the language) is VERSION_1. See page 139 for copybook details.

return_area
A 5-word output area to receive response and reason codes, and a message pointer field. For more details see "Return area for the EXCI CALL interface" on page 138.

user_token
A 1-word output area containing a 32-bit token supplied by the CICS external interface to represent the client program.

The user token corresponds to the *user-name* parameter. The client program must pass this token on all subsequent external CICS interface commands made for the user defined on the *user_name* parameter.

call_type
A 1-word input area indicating the function of the command. It must be set to 1 in the client program to indicate that this is an Initialize_User command.

The equated value for this call in the CICS-supplied copybook DFHXCPL*x* (where *x* indicates the language) is INIT_USER. See page 139 for copybook details.

user_name
An input area holding a name that identifies the user of the external CICS interface. Generally, this is the client program. If this user is to use a specific pipe, then the value in *user_name* must match that of the NETNAME attribute of the CONNECTION definition for the specific pipe.

Responses and reason codes

For all non-zero response codes, a unique reason code value identifies the reason for the response.

Note: All numeric response and reason code values are in decimal.

The following is a summary of the response and reason codes that the external CICS interface can return on the Initialize_User call:

Response OK
Command executed successfully (RC 0). Reason code:

0	Normal response

Response WARNING
The command executed successfully, but with an error (RC 4). Reason codes:

3	VERIFY_BLOCK_FM_ERROR
4	WS_FREEMAIN_ERROR

Response RETRYABLE
The command failed because of setup errors but can be reissued (RC 8). Reason code:

201	NO_CICS_IRC_STARTED

Response USER_ERROR
The command failed because of an error in either the client or the server (RC 12). Reason codes:

401	INVALID_CALL_TYPE
402	INVALID_VERSION_NUMBER
403	INVALID_USER_NAME
410	DFHMEBM_LOAD_FAILED
411	DFHMET4E_LOAD_FAILED
412	DFHXCURM_LOAD_FAILED
413	DFHXCTRA_LOAD_FAILED
419	CICS_AFCB_PRESENT
420	DFHXCOPT_LOAD_FAILED

Response SYSTEM_ERROR
The command failed (RC 16). Reason codes:

601	WS_GETMAIN_ERROR
602	XCGLOBAL_GETMAIN_ERROR
603	XCUSER_GETMAIN_ERROR
605	VERIFY_BLOCK_GM_ERROR
606	SSI_VERIFY_FAILED
607	CICS_SVC_CALL_FAILURE

622	ESTAE_SETUP_FAILURE
623	ESTAE_INVOKED
627	INCORRECT_SVC_LEVEL

For more information about response codes, see “Response code values” on page 138.

For information about the reason codes, see “Chapter 17. Response and reason codes returned on EXCI calls” on page 189.

Allocate_Pipe

Function

Allocate a single session, or pipe, to a CICS region. This command does not connect the client program to a CICS region; this happens on the Open_Pipe command. You can allocate up to 25 pipes in an EXCI partition.

Syntax

CALL DFHXCIS

```
►►──CALL DFHXCIS,(version_number,──return_area,──user_token,──call_type,──────────►

►─pipe_token,──┬─CICS_applid,─┬──allocate_opts),──VL,MF=(E,(1))──────────────────►◄
               └─null_ptr,────┘
```

Parameters

version_number
: A fullword binary input area indicating the version of the external CICS interface parameter list being used. It must be set to 1 in the client program.

  The equated value for this parameter in the CICS-supplied copybook DFHXCPL*x* (where *x* indicates the language) is VERSION_1. See page 139 for copybook details.

return_area
: A 5-word output area to receive response and reason codes, and a message pointer field. For more details see “Return area for the EXCI CALL interface” on page 138.

user_token
: The 1-word token returned on the Initialize_User command.

call_type
: A 1-word input area indicating the function of the command. It must be set to 2 in the client program to indicate that this is an Allocate_Pipe command.

  The equated value for this call in the CICS-supplied copybook DFHXCPL*x* (where *x* indicates the language) is ALLOCATE_PIPE. See page 139 for copybook details.

pipe_token
: A 1-word output area. CICS returns a 32-bit token in this area to represent the allocated session. This token must be used on any subsequent command that uses this session.

CICS_applid (or null_ptr)
: An 8-byte input area containing the generic applid of the CICS system to which the allocated session is to be connected.

  Although an applid is required to complete the Allocate_Pipe function, this parameter is optional on the Allocate_Pipe call. You can either specify the applid on this parameter to the Allocate_Pipe call, or in the user-replaceable module, DFHXCURM, using the URMAPPL parameter (DFHXCURM is always invoked during Allocate_Pipe processing). You can also use the URMAPPL parameter in DFHXCURM to override an applid specified on the Allocate_Pipe call. See “Chapter 12. The EXCI user-replaceable module” on page 157 for information about the URMAPPL parameter.

If you omit the applid from the call, you must ensure that the CALL parameter list contains a null address for *CICS_applid*. How you do this depends on the language you are using for the non-CICS client program. For an example of a call that omits an optional parameter, see “Example of EXCI CALLs with null parameters” on page 140.

allocate_opts

A 1-byte input area to represent options specified on this command. The options specify which type of session is to be used—specific or generic. X'00' represents a specific session. X'80' represents a generic session.

The equated value for these options in the CICS-supplied copybook DFHXCPL*x* (where *x* indicates the language) are SPECIFIC_PIPE and GENERIC_PIPE. See page 139 for copybook details.

Responses and reason codes

For all non-zero response codes, a unique reason code value identifies the reason for the response.

Note: All numeric response and reason code values are in decimal.

The following is a summary of the response and reason codes that the external CICS interface can return on the Allocate_Pipe call:

Response OK

Command executed successfully (RC 0). Reason code:

0	Normal response

Response USER_ERROR

The command failed because of an error in either the client or the server (RC 12). Reason codes:

401	INVALID_CALL_TYPE
402	INVALID_VERSION_NUMBER
404	INVALID_USER_TOKEN

Response SYSTEM_ERROR

The command failed (RC 16). Reason codes:

604	XCPIPE_GETMAIN_ERROR
608	IRC_LOGON_FAILURE
622	ESTAE_SETUP_FAILURE
623	ESTAE_INVOKED
628	IRP_LEVEL_CHECK_FAILURE

For information about response codes, see “Response code values” on page 138.

For information about the reason codes, see “Chapter 17. Response and reason codes returned on EXCI calls” on page 189.

Open_Pipe

Function

Cause IRC to connect an allocated pipe to a receive session of the appropriate connection defined in the CICS region named either on the Allocate_Pipe command, or in DFHXCURM. The appropriate connection is either:

- The EXCI connection with a NETNAME value equal to the *user_name* parameter on the Initialize_User command (that is, you are using a specific connection, dedicated to this client program), or
- The EXCI connection defined as generic.

Note: This command should be used only when there is a DPL call ready to be issued to the CICS region. When not in use, EXCI sessions should not be left open.

If sessions are left open, CICS may not be able to shut its IRC facility in an orderly manner. A normal shutdown of CICS without the support of the shutdown assist transaction waits if any EXCI sessions are not closed. CICS issues message DFHIR2321 indicating the following information:

- The netname of the session if it is on a specific connection
- The word GENERIC if the open sessions are on a generic connection.

Provided that at least one DPL_Request call has been issued on the session, message DFHIR2321 also shows the job name, step name, and procedure name of the client job that is using the session, and the ID of the VSE image on which the client program is running.

Syntax

CALL DFHXCIS

```
►►──CALL DFHXCIS,(version_number,──return_area,──user_token,──call_type,──────────►

►─pipe_token),──VL,MF=(E,(1))──────────────────────────────────────────────────►◄
```

Parameters

version_number
A fullword binary input area indicating the version of the external CICS interface parameter list being used. It must be set to 1 in the client program.

The equated value for this parameter in the CICS-supplied copybook DFHXCPL*x* (where *x* indicates the language) is VERSION_1. See page 139 for copybook details.

return_area
A 5-word output area to receive response and reason codes, and a message pointer field. For more details, see “Return area for the EXCI CALL interface” on page 138.

user_token
The 1-word token returned on the Initialize_User command.

call_type
A 1-word input area indicating the function of the command. This must be set to 3 in the client program to indicate that this is an Open_pipe command.

The equated value for this call in the CICS-supplied copybook DFHXCPL*x* (where *x* indicates the language) is OPEN_PIPE. See page 139 for copybook details.

pipe_token
A 1-word output area containing the token passed by CICS on the Allocate_Pipe command. It represents the pipe being opened on this command.

Responses and reason codes

For all non-zero response codes, a unique reason code value identifies the reason for the response.

Note: All numeric response and reason code values are in decimal.

The following is a summary of the response and reason codes that the external CICS interface can return on the Open_Pipe call:

Response OK
Command executed successfully (RC 0). Reason code:

0	NORMAL

Response WARNING
The command executed successfully, but with an error (RC 4). Reason codes:

1	PIPE_ALREADY_OPEN

Response RETRYABLE
The command failed because of setup errors but can be reissued (RC 8). Reason codes:

202	NO_PIPE
203	NO_CICS

Response USER_ERROR
The command failed because of an error in either the client or the server (RC 12). Reason codes:

401	INVALID_CALL_TYPE
402	INVALID_VERSION_NUMBER
404	INVALID_USER_TOKEN
418	INVALID_PIPE_TOKEN

Response SYSTEM_ERROR
The command failed (RC 16). Reason codes:

609	IRC_CONNECT_FAILURE
621	PIPE_RECOVERY_FAILURE
622	ESTAE_SETUP_FAILURE
623	ESTAE_INVOKED

For information about response codes, see “Response code values” on page 138.

For information about the reason codes, see “Chapter 17. Response and reason codes returned on EXCI calls” on page 189.

DPL_Request

Function

Issue a distributed program link request across an open pipe connected to the CICS region on which the server (or target) application program resides. The command is synchronous, and the VSE task waits for a response from CICS. Once a pipe is opened, any number of DPL requests can be issued before the pipe is closed. To the server program, the link request appears just like a standard EXEC CICS LINK request from another CICS region, and it is not aware that it is sent from a non-CICS client program using EXCI.

Syntax

CALL DFHXCIS

```
►►──CALL DFHXCIS,(version_number,──return_area,──user_token,──call_type,──────────►

►─pipe_token,──pgmname,──┬─COMMAREA,─┬──COMMAREA_len,──data_len,──────────────────►
                         └─null_ptr,─┘

►──┬─transid,──┬──┬─uowid,────┬──┬─userid,───┬──dpl_retarea,──┬─DPL_opts─┬─)──────►
   └─null_ptr,─┘  └─null_ptr,─┘  └─null_ptr,─┘                └─null_ptr─┘

►─VL,MF=(E,(1))──────────────────────────────────────────────────────────────────►◄
```

Parameters

version_number
A fullword binary input area indicating the version of the external CICS interface parameter list being used. It must be set to 1 in the client program.

The equated value for this parameter in the CICS-supplied copybook DFHXCPL*x* (where *x* indicates the language) is VERSION_1. See page 139 for copybook details.

return_area
A 5-word output area to receive response and reason codes, and a message pointer field. For more details, see "Return area for the EXCI CALL interface" on page 138.

user_token
A 1-word input area specifying the user token returned to the client program on the Initialize_User command.

call_type
A 1-word input area indicating the function of the command. This must be set to 6 in the client program to indicate that the pipe is now being used for the DPL_Request call.

The equated value for this call in the CICS-supplied copybook DFHXCPL*x* (where *x* indicates the language) is DPL_REQUEST. See page 139 for copybook details.

pipe_token

A 1-word input area specifying the token returned by EXCI on the Allocate_Pipe command. It represents the pipe being used for the DPL_Request call.

pgmname

The 8-character name of the CICS application program being called as the server program.

This is either the name as specified on a predefined PROGRAM resource definition installed in the CICS server region, or as it is known to a user-written autoinstall program if the program is to be autoinstalled. The program can be defined in the CICS server region as a local program, or it can be defined as remote. Programs defined as remote enable "daisy-chaining", where EXCI-CICS DPL calls become EXCI-CICS-CICS DPL calls.

COMMAREA (or null_ptr)

A variable length input area for the communications area (COMMAREA) between the client and server programs. The length is defined by *COMMAREA_len*.

This is the storage area that contains the data to be sent to the CICS application program. This area is also used to receive the updated COMMAREA from the CICS application program (the server program).

This parameter is optional. If it is not required, you must ensure that the CALL parameter list contains a null address for this parameter. How you do this depends on the language you are using for the non-CICS client program. For an example of a call that omits an optional parameter, see "Example of EXCI CALLs with null parameters" on page 140.

COMMAREA_len

A fullword binary input area. This parameter specifies the length of the COMMAREA. It is also the length of the server program's COMMAREA (EIBCALEN).

If you specify a COMMAREA, you must also specify this parameter to define the length.

If you don't specify a COMMAREA, this parameter is ignored.

data_len

A fullword binary input area. This parameter specifies the length of contiguous storage, from the start of the COMMAREA, to be sent to the server program.

This parameter restricts the amount of data sent to the server program, and should be used to optimize performance if, for example, the COMMAREA is large but the amount of data being passed is small.

Note that on return from the server program, the EXCI data transformer program ensures that the COMMAREA in the non-CICS client program is the same as that of the server program. This caters for the following conditions:

- The data returned is more than the data passed in the original COMMAREA.
- The data returned is less than the data passed in the original COMMAREA.
- There is no data returned because it is unchanged.
- The server is returning null data.

The value of *data_len* must not be greater than the value of *COMMAREA_len*. A value of zero is valid and results in no data being sent to the server program.

If you don't specify a COMMAREA, this parameter is ignored.

transid (or null_ptr)

A 4-character input area containing the id of the CICS mirror transaction under which the server program is to run. This transaction must be defined to the CICS server region, and its definition *must observe the following rules*:

- It must *not* specify the server program as the initial program of the transaction
- It *must* specify the mirror program DFHMIRS, and the profile DFHCICSA.

Failure to specify DFHMIRS as the initial program means that a COMMAREA passed from the client application program is not passed to the CICS server program. Furthermore, the DPL request fails and the client application program receives a response of SYSTEM_ERROR and reason SERVER_PROTOCOL_ERROR.

When the CICS server region receives a DPL request, it attaches the mirror transaction and invokes DFHMIRS. The mirror program then passes control to the requested server program, passing the COMMAREA supplied by the client program. The COMMAREA passed to the server program is primed with the data only, the remainder of the COMMAREA being set to nulls.

The purpose of the *transid* parameter is to distinguish between different invocations of the server program. This enables you to run different invocations of the server program under transactions that specify different attributes. For example, you can vary the transaction priorities, or the security requirements.

A transid is optional. By default, the CICS server region uses the CICS-supplied mirror transaction, CSMI. If you don't want to specify *transid*, you must ensure that the CALL parameter list contains a null address for this parameter. How you do this depends on the language you are using for the non-CICS client program. For an example of a call that omits an optional parameter, see "Example of EXCI CALLs with null parameters" on page 140.

uowid (or null_ptr)

An input area containing a unit-of-work identifier, using the APPC architected format, that is passed on the DPL_Request for correlation purposes.

This parameter is optional. If you don't want to specify *uowid*, you must ensure that the CALL parameter list contains a null address for this parameter. How you do this depends on the language you are using for the non-CICS client program. For an example of a call that omits an optional peremeter, see "Example of EXCI CALLs with null parameters" on page 140.

If specified, the *uowid* parameter is passed to the CICS server region, which uses it as the APPC UOWID for the first unit of work executed by the CICS server program. If the server program issues intermediate syncpoints before returning to the client program, CICS uses the supplied *uowid* for the subsequent units of work, but with the two byte sequence number incremented for each new logical unit of work. If the CICS server program updates remote resources, the client-supplied APPC UOWID is distributed to the remote systems that own the resources.

The *uowid* parameter is supplied on the EXCI CALL interface for correlation purposes only, to allow units of work that originated from a particular client program to be identified in CICS. The *uowid* is not provided for recovery purposes between CICS and the client program. No syncpoint coordination occurs between the client program and CICS, because all CICS server programs called from a client program run with SYNCONRETURN specified.

The *uowid* can be a maximum of 27 bytes long and has the following format:

- A 1-byte length field containing the overall length of the UOWID (excluding this field).
- A 1-byte length field containing the length of the logical unit name (excluding this field).
- A logical unit name field of variable length up to a maximum of 17 bytes.

  To conform to APPC architecture rules, the LUNAME must be of the form *AAAAAAAA.BBBBBBBB*, where *AAAAAAAA* is optional and:

  – AAAAAAAA and BBBBBBBBare 18–byte names separated by a period

  – If AAAAAAAA is omitted, the period must also be omitted

  – AAAAAAAA and BBBBBBBB must be type–1134 symbol strings (that is, character strings consisting of one or more EBCDIC uppercase letters A–Z and 0–9, the first character of which must be an uppercase letter).
- The clock value—the middle 6 bytes of an 8-byte store clock (STCK) value.
- A 2-byte sequence number.

If you omit a unit-of-work identifier (by specifying a null pointer), and the DPL request is not part of an RRMS unit-of-recovery, the external CICS interface generates one for you, consisting of the following:

- A 1-byte length field set to X'1A'.
- A 1-byte LU length field set to X'11'.
- A 17-byte LU name consisting of:

  – An 8-byte eyecatcher set to 'DFHEXCIU'.

  – A 1-byte field containing a period (.)

  – A 4-byte field containing the VSE, in characters, under which the client is running.

  – A 4-byte field containing the address space id (ASID) in which the VSE client program is running. The field contains the four character EBCDIC representation of the two–byte hex address space id.
- The clock value—the middle 6 bytes of an 8-byte store clock (STCK) value
- A two—byte sequence number set to X'0001'.

userid (or null_ptr)

An 8-character input area containing the external security manager (ESM) userid for user security checking in the CICS region. The external CICS interface passes this userid to the CICS server region for user resource and command security checking in the server application program.

A userid is required only if the MRO connection specifies the ATTACHSEC(IDENTIFY) attribute. If the connection specifies ATTACHSEC(LOCAL), the CICS server region applies link security checking. See the *CICS Security Guide* for information about link security on MRO connections.

See also “Chapter 15. Security” on page 169 for information about external CICS interface security considerations.

This parameter is optional. However, if you don’t specify a userid, the external CICS interface passes the security userid under which the client program is running. For example, if the client program is running as a VSE batch job, the external CICS interface obtains and passes the userid specified on the USER parameter of the ID statement in the batch job JCL.

If you want to let *userid* default, you must ensure that the CALL parameter list contains a null address for this parameter. How you do this depends on the language you are using for the non-CICS client program. For an example of a call that omits an optional parameter, see “Example of EXCI CALLs with null parameters” on page 140.

dpl_retarea

A 12-byte output area into which the DPL_Request processor places responses to the DPL request. Generally, these responses are from CICS, but in some cases the error detection occurs in the external CICS interface, which returns exception conditions that are the equivalent of those returned by an EXEC CICS LINK command.

This field is only meaningful in the following circumstances:

- The response field of the EXCI *return-area* has a zero value, or
- The EXCI *return-area* indicates that the server program has abended (response=USER_ERROR and reason=SERVER_ABENDED).

The 12 bytes form three fields, providing the following information:

Field 1 (*fullword value***)**

This field is a fullword containing a RESP value from the DPL_Request call. See “Error codes” on page 146 for the RESP values that can be returned on a DPL_Request call.

If the DPL_Request call reaches CICS, this field contains the EIBRESP value, otherwise it contains an equivalent response set by the external CICS interface. If this field is set by the external CICS interface, RESP is further qualified by a RESP2 value in the second field.

A zero value is the normal response, which equates to EXEC_NORMAL in the return codes copybooks.

Field 2 (*fullword value***)**

This field is a fullword that may contain a RESP2 value from the link request, further qualifying the RESP value in field 1.

If the DPL_Request call reaches CICS, the RESP2 field is generally null (CICS does not return RESP2 values across MRO links). However, if the RESP field indicates SYSIDERR (value 53), this field provides a reason code. See“Dpl_retarea return codes” on page 139 for more information.

If the RESP field is set by the external CICS interface, it is further qualified by a RESP2 value in this second field. For example, if the *data_len* parameter specifies a value greater than the *COMMAREA_len* parameter, the external CICS interface returns the RESP value 22 (which equates to EXEC_LENGERR in the return codes copybooks), and a RESP2 value of 13.

See the LINK conditions in the *CICS Application Programming Reference* for full details of the possible RESP and RESP2 values.

Note: The data transformer program makes special use of the RESP2 field. If any error occurs in the transformer, the error is returned in RESP2.

Field 3 (*fullword value***)**
The third field, a 4-character field, contains:
- The abend code if the server program abended
- Four blanks if the server program did not abend.

If a server program abends, it is backed out to its last syncpoint which may be the start of the task, or an intermediate syncpoint. The server program can issue intermediate syncpoints because SYNCONRETURN is forced.

DPL_opts (or null_ptr)
A 1-byte input area indicating options to be used on the DPL_Request call.

For CICS Transaction Server for VSE/ESA Release 1, X'80' is the only valid option, and it indicates that SYNCONRETURN is specified. SYNCONRETURN is mandatory.

The equated value for this parameter in the CICS-supplied copybook DFHXCPL*x* (where *x* indicates the language) is SYNCONRETURN. See page 139 for copybook details.

SYNCONRETURN specifies that the server region is tot ake a syncpoint on successful completion of the server program. Note that although SYNCONRETURN is mandatory, this does not prevent a server program from taking its own explicit syncpoints.

Responses and reason codes

For all non-zero response codes, a unique reason code value identifies the reason for the response.

Note: All numeric response and reason code values are in decimal.

The following is a summary of the response and reason codes that the external CICS interface can return on the DPL call:

Response OK
Command executed successfully (RC 0). Reason code:

0 NORMAL

Response WARNING
The command executed successfully, but with an error (RC 4). Reason codes:

6 IRP_IOAREA_FM_FAILURE
7 SERVER_TERMINATED

Response RETRYABLE
The command failed because of setup errors but can be reissued (RC 8). Reason codes:

203 NO_CICS

Response USER_ERROR
The command failed because of an error in either the client or the server (RC 12). Reason codes:

401	INVALID_CALL_TYPE
402	INVALID_VERSION_NUMBER
404	INVALID_USER_TOKEN
406	PIPE_NOT_OPEN
407	INVALID_USERID
408	INVALID_UOWID
409	INVALID_TRANSID
414	IRP_ABORT_RECEIVED
415	INVALID_CONNECTION_DEFN
416	INVALID_CICS_RELEASE
417	PIPE_MUST_CLOSE
418	INVALID_PIPE_TOKEN
422	SERVER_ABENDED

Response SYSTEM_ERROR

The command failed (RC 16). Reason codes:

612	TRANSFORM_1_ERROR
613	TRANSFORM_4_ERROR
614	IRP_NULL_DATA_RECEIVED
615	IRP_NEGATIVE_RESPONSE
616	IRP_SWITCH_PULL_FAILURE
617	IRP_IOAREA_GM_FAILURE
619	IRP_BAD_IOAREA
620	IRP_PROTOCOL_ERROR
622	ESTAE_SETUP_FAILURE
623	ESTAE_INVOKED
624	SERVER_TIMEDOUT
625	STIMER_SETUP_FAILURE
626	STIMER_CANCEL_FAILURE
629	SERVER_PROTOCOL_ERROR

For information about response codes, see “Response code values” on page 138.

For information about the reason codes, see “Chapter 17. Response and reason codes returned on EXCI calls” on page 189.

Close_PIPE

Function

Disconnect an open pipe from CICS. The pipe remains in an allocated state, and its tokens remain valid for use by the same user. To reuse a closed pipe, the client program must first reissue an Open_Pipe command using the pipe token returned on the Allocate_Pipe command for the pipe. Pipes should not be left open when not in use because this prevents CICS from shutting down its IRC facility in an orderly manner. Therefore, the Close_Pipe command should be issued as soon as possible after all DPL_Request calls have completed.

Syntax

CALL DFHXCIS

```
►►──CALL DFHXCIS,(version_number,──return_area,──user_token,──call_type,─────────►

►──pipe_token),──VL,MF=(E,(1))──────────────────────────────────────────────────►◄
```

Parameters

version_number

A fullword binary input area indicating the version of the external CICS interface parameter list being used. It must be set to 1 in the client program.

The equated value for this parameter in the CICS-supplied copybook DFHXCPL*x* (where *x* indicates the language) is VERSION_1. See page 136 for copybook details.

return_area

A 5-word output area to receive response and reason codes, and a message pointer field. For more details, see “Return area for the EXCI CALL interface” on page 138.

user_token

The 1-word input area specifying the token, returned to the client program by EXCI on the Initialize_User command, that represents the user of the pipe being closed.

call_type

A 1-word input area indicating the function of the command. This must be set to 4 in the client program to indicate that this is a Close_Pipe command.

The equated value for this call in the CICS-supplied copybook DFHXCPL*x* (where *x* indicates the language) is CLOSE_PIPE. See page 139 for copybook details.

pipe_token

A 1-word input area specifying the token, returned to the client program by EXCI on the original Allocate_Pipe command, that represents the pipe being closed.

Responses and reason codes

For all non-zero response codes, a unique reason code value identifies the reason for the response.

Note: All numeric response and reason code values are in decimal.

The following is a summary of the response and reason codes that the external CICS interface can return on the Close_Pipe call:

Response OK

Command executed successfully (RC 0). Reason code:

0	NORMAL

Response WARNING

The command executed successfully, but with an error (RC 4). Reason codes:

2	PIPE_ALREADY_CLOSED

Response USER_ERROR

The command failed because of an error in either the client or the server (RC 12). Reason codes:

401	INVALID_CALL_TYPE
402	INVALID_VERSION_NUMBER
404	INVALID_USER_TOKEN
418	INVALID_PIPE_TOKEN

Response SYSTEM_ERROR

The command failed (RC 16). Reason codes:

610	IRC_DISCONNECT_FAILURE
622	ESTAE_SETUP_FAILURE
623	ESTAE_INVOKED

For information about response codes, see “Response code values” on page 138.

For information about the reason codes, see “Chapter 17. Response and reason codes returned on EXCI calls” on page 189.

Deallocate_Pipe

Function

Deallocate a pipe from CICS. On completion of this command, the pipe can no longer be used, and its associated tokens are invalid. This command should be issued for pipes that are no longer required. This command frees storage associated with the pipe.

Syntax

CALL DFHXCIS

```
►►──CALL DFHXCIS,(version_number,──return_area,──user_token,──call_type,──────────►

►─pipe_token),──VL,MF=(E,(1))────────────────────────────────────────────────────►◄
```

Parameters

version_number
: A fullword binary input area indicating the version of the external CICS interface parameter list being used. It must be set to 1 in the client program.

The equated value for this parameter in the CICS-supplied copybook DFHXCPL*x* (where *x* indicates the language) is VERSION_1. See page 139 for copybook details.

return_area
: A 5-word output area to receive response and reason codes, and a message pointer field. For more details, see “Return area for the EXCI CALL interface” on page 138.

user_token
: A 1-word input area containing the token returned on the Initialize_User command.

call_type
: A 1-word input area indicating the function of the command. This must be set to 5 in the client program to indicate that this is a Deallocate_Pipe command.

The equated value for this call in the CICS-supplied copybook DFHXCPL*x* (where *x* indicates the language) is DEALLOCATE_PIPE. See page 139 for copybook details.

pipe_token
: A 1-word input area containing the token passed back on the original Allocate_Pipe command, that represents the pipe now being deallocated.

Responses and reason codes

For all non-zero response codes, a unique reason code value identifies the reason for the response.

Note: All numeric response and reason code values are in decimal.

The following is a summary of the response and reason codes that the external CICS interface can return on the Deallocate_Pipe call:

Response OK

Command executed successfully (RC 0). Reason code:

0	NORMAL

Response WARNING

The command succeeded successfully, but with an error (RC 4). Reason codes:

5	XCPIPE_FREEMAIN_ERROR
6	IRP_IOAREA_FM_FAILURE

Response USER_ERROR

The command failed because of an error in either the client or the server (RC 12). Reason codes:

401	INVALID_CALL_TYPE
402	INVALID_VERSION_NUMBER
404	INVALID_USER_TOKEN
405	PIPE_NOT_CLOSED
418	INVALID_PIPE_TOKEN

Response SYSTEM_ERROR

The command failed (RC 16). Reason codes:

611	IRC_LOGOFF_FAILURE
622	ESTAE_SETUP_FAILURE
623	ESTAE_INVOKED

For information about response codes, see “Response code values” on page 138.

For information about the reason codes, see “Chapter 17. Response and reason codes returned on EXCI calls” on page 189.

Response code values

The values that can be returned in the response field are shown in Table 11 (all values are in decimal).

Table 11. EXCI response codes (returned in response field of return_area)

Value	Meaning	Explanation
0	OK	For all EXCI CALL commands other than the DPL_request, the call was successful. If an OK response is received for a DPL_request, you must also check *dpl_retarea* to ensure CICS did not return a condition code. If the EIBRESP field of *Dpl_retarea* is zero, the DPL call was successful.
4	WARNING	The external CICS interface detected an error, but this did not stop the CALL command completing successfully. The reason code field describes the error detected.
8	RETRYABLE	The EXCI CALL command failed. This class of failure relates to errors in the setup of the system environment, and not errors in the external CICS interface or client program. The reason code documents the specific error in the environment setup. The external CICS interface command can be reissued without changing the client program once the environment error has been corrected. The environmental errors concerned are ones that do not require a VSE re-IPL. Each reason code value for a RETRYABLE response documents whether the CALL can be reissued directly, or whether the pipe being used has to be closed and reopened first.
12	USER_ERROR	The EXCI CALL command failed. This class of error means there is an error either in the client program, or in the CICS server program, or in the CICS server region. An example of an error in the CICS server system would be a failed security check, or an abend of the CICS server program, in which case the abend code is set in the abend code field of *dpl_retarea*. Each reason code value for a response of USER_ERROR explains whether the command can be reissued directly, or whether the pipe being used has to be closed and reopened first.
16	SYSTEM_ ERROR	The EXCI CALL command failed. This class of error means that the external CICS interface has detected an error. The reason code value identifies the specific error. If the error can be corrected, then the command can be reissued. Each reason code value for a SYSTEM_ERROR response explains whether the command can be reissued directly, or whether the pipe being used has to be closed and reopened first.

Return area for the EXCI CALL interface

The format of the 5-word return area for the EXCI CALL interface is as follows:

1. 1–word response field.
2. 1–word reason field.
3. Two 1–word subreason fields—subreason field-1 and subreason field-2.
4. 1–word CICS message pointer field. This is zero if there is no message present. If a message is present, this field contains the address of the storage area containing the message, which is formatted as follows:
   - A 2-byte LL field. LL is the length of the message plus the length of the LLBB field.
   - A 2-byte BB field, set to binary zero.

- A variable length field containing the text of the message.

Return area and function call EQUATE copybooks

CICS provides four language-specific copybooks that map the storage areas for the *return_area* and *dpl_retarea* parameters of the EXCI CALL commands. The copybooks also provide EQUATE statements for each type of EXCI CALL.

These copybooks, and the libraries they are supplied in for the supported languages, are shown in Table 12.

Table 12. Supplied copybooks of return areas and equated names

Copybook name	Language	Library
DFHXCPLD	Assembler	PRD1BASE
DFHXCPLH	C	PRD1BASE
DFHXCPLO	COBOL	PRD1BASE
DFHXCPLL	PL/I	PRD1BASE

Return codes

CICS provides four language-specific copybooks that map the storage areas for the *return_area* and *dpl_retarea* parameters of the EXCI CALL commands. The copybooks also provide EQUATE statements for each type of EXCI CALL. The names of the copybooks for the supported languages, and the libraries they are supplied in, are shown in Table 13.

Table 13. Supplied copybooks of RESPONSE and REASON codes

Copybook name	Language	Library
DFHXCRCD	Assembler	PRD1BASE
DFHXCRCH	C	PRD1BASE
DFHXCRCO	COBOL	PRD1BASE
DFHXCRCL	PL/I	PRD1BASE

Dpl_retarea return codes

These are the same as for CICS-to-CICS EXEC CICS DPL commands but with the following additions for the EXCI call interface:

Table 14. Exceptional conditions. RESP and RESP2 values returned to dpl_retarea

Condition	RESP2	Meaning
INVREQ	21	SYNCONRETURN_NOT_SPECIFIED
LENGERR	22	COMMAREA_LEN_TOO_BIG
LENGERR	23	COMMAREA_BUT_NO_COMMAREA_LEN

SYSIDERR also can be returned on an EXCI DPL_Request, if the DPL_Request specifies a program defined in the CICS server region as a remote program, and the link between the server and the remote CICS region is not open. In this situation, SYSIDERR is returned in the first word of the DPL_Retarea (code 53). The reason code qualifying SYSIDERR is placed in the second word of this area (the equivalent of a RESP2 value).For SYSIDERR, the information stored in this field is derived from bytes 1 and 2 of the CICS EIBRCODE field. For example, if a remote link is not available, the EIBRCODE value stored in bytes 2 and 3 of the DPL_Retarea RESP2 field is X'0800'. For a list of the SYSIDERR reason codes that can be returned in the RESP2 field, see the SYSIDERR section of the notes on EIBRCODE in the *CICS Application Programming Reference* manual.

TERMERR also may be returned on an EXCI DPL request if the DPL request was to a program defined as remote, and an unrecoverable error occurs during conversation with the mirror on the remote CICS system. For example, suppose client program BATCH1 issues an EXCI DPL request to CICSA for program PROG1, which is defined as remote, and the request is function-shipped to CICSB where the program resides. If the session between CICSA and CICSB fails, or CICSB itself fails whilst executing the program PROG1, then TERMERR is returned to CICSA, and in turn to BATCH1.

No unique EXCI_DPL_RESP2 values are returned for TERMERR, PGMIDERR, NOTAUTH, and ROLLBACK.

Example of EXCI CALLs with null parameters

If you omit an optional parameter, such as *userid* on a DPL_Request, you must ensure that the parameter list is built with a null address for the missing parameter. The example that follows illustrates how to issue an EXCI DPL Call with the *userid* and *uowid* parameters omitted in a COBOL program.

DPL CALL without userid and uowid (COBOL): In this example, the DPL parameters used on the call are defined in the WORKING-STORAGE SECTION, as follows:

DPL parameter	COBOL variable	Field definition
version_number	01 VERSION-1	PIC S9(8) COMP VALUE 1.
return_area	01 EXCI-RETURN-CODE.	(structure)
user_token	01 USER-TOKEN	PIC S9(8) COMP VALUE ZERO.
call_type	03 DPL-REQUEST	PIC S9(8) COMP VALUE 6.
pipe_token	01 PIPE-TOKEN	PIC S9(8) COMP VALUE ZERO.
pgmname	01 TARGET-PROGRAM	PIC X(8) VALUE ″DFHœAXCS″.
commarea	01 COMMAREA.	(structure)
commarea_len	01 COMM-LENGTH	PIC S9(8) COMP VALUE 98.
data_len	01 DATA-LENGTH	PIC S9(8) COMP VALUE 18.
transid	01 TARGET-TRANSID	PIC X(4) VALUE ″EXCI″.
dpl_retarea	01 EXCI-DPL-RETAREA.	(structure)
dpl_opts	01 SYNCONRETURN	PIC X VALUE X'80'.

The variable used for the null address is defined in the LINKAGE SECTION, as follows:

```
LINKAGE  SECTION.
   01   NULL-PTR             USAGE  IS  POINTER.
```

Using the data names specified in the WORKING-STORAGE SECTION as described above, and the NULL-PTR name as described in the LINKAGE SECTION, the following invocation of the DPL function omits the *uowid* and the *userid* parameters, and replaces them in the parameter list with the NULL-PTR variable:

```
 DPL-SECTION.
*
     SET ADDRESS OF NULL-PTR TO NULLS.
*
     CALL 'DFHXCIS' USING  VERSION-1   EXCI-RETURN-CODE  USER-TOKEN
                          DPL-REQUEST    PIPE-TOKEN   TARGET-PROGRAM
                          COMMAREA       COMM-LENGTH  DATA-LENGTH
                          TARGET-TRANSID NULL-PTR     NULL-PTR
                          EXCI-DPL-RETAREA    SYNCONRETURN.
```

This example is taken from the CICS-supplied sample external CICS interface program, DFH0CXCC, which is supplied in PRD1BASE. For an example of how to omit the same parameters from the DPL call in the other supported languages, see the following sample programs:

DFH$AXCC
The assembler sample

DFH$PXCC
The PL/I sample

DFH$DXCC
The C/370™ sample.

Chapter 10. The EXEC CICS interface

This chapter describes the EXEC CICS LINK PROGRAM command for the external CICS interface. It covers the following topics:

- “EXEC CICS LINK command” on page 144
- “Retries on an EXEC CICS LINK command” on page 148
- “Translation required for EXEC CICS LINK command” on page 150

The external CICS interface provides this as a single, composite command, to invoke all the calls of the EXCI CALL interface. Each time you issue an EXEC CICS LINK PROGRAM command in a client application program, the external CICS interface invokes on your behalf each of the six EXCI calls.

EXEC CICS LINK command

Purpose

Link from a VSE client program to the specified server program in a server CICS region.

Format

LINK

```
►►──LINK──PROGRAM(name)──RETCODE(data-area)──┬───────────────┬──────────────────►
                                              └─SYNCONRETURN─┘

►──┬──────────────────────────────────────────┬──┬───────────────┬─────────────►
   └─COMMAREA(data-area)───LENGTH(data-value)─┘  └─APPLID(name)─┘

►──┬───────────────┬──┬───────────────────────┬───────────────────────────────►◄
   └─TRANSID(name)─┘  └─DATALENGTH(data-value)─┘
```

Error conditions:LENGERR, LINKERR, NOTAUTH, PGMIDERR, ROLLEDBACK, SYSIDERR, TERMERR, WARNING

Comments

With the exception of the APPLID and RETCODE parameters, the external CICS interface parameters for an EXEC CICS LINK command are the same as for a CICS-CICS DPL command.

This book describes only those parameters that you can use with the external CICS interface. For programming information about the EXEC CICS LINK PROGRAM command, see the *CICS Application Programming Reference* manual.

Note that the LENGTH and DATALENGTH parameters specify halfword binary values, unlike the corresponding *COMMAREA_len* and *data_len* parameters of the EXCI CALL interface, which specify fullword values.

An external CICS interface EXEC CICS LINK command always uses a generic connection.

Parameters

The parameters that you can use on the external CICS interface form of the LINK command, are as follows:

APPLID(*name***)**
Specifies the generic APPLID of the target CICS server region.

Although an applid is required for an external CICS interface command, this parameter is optional on the LINK command itself because you can also specify it in the user-replaceable module, DFHXCURM. If you omit the generic APPLID from the LINK command, you must ensure it is specified by the user-replaceable module, DFHXCURM, on the URMAPPL parameter. You can also use the URMAPPL parameter in DFHXCURM to override an applid specified on the LINK command. See “Chapter 12. The EXCI user-replaceable module” on page 157 for information about the URMAPPL parameter.

COMMAREA(*data-area***)**
Specifies a communication area that is to be made available to the invoked program. In this option, a pointer to the data area is passed.

See the *CICS Application Programming Guide* for more information about passing data to CICS application programs.

DATALENGTH(*data-value***)**
Specifies a halfword binary value that is the length of a contiguous area of storage, from the start of the COMMAREA, to be passed to the invoked program. If the amount of data being passed in a COMMAREA is small, but the COMMAREA itself is large so that the linked-to program can return the requested data, you should specify DATALENGTH in the interest of performance.

LENGTH(*data-value***)**
Specifies a halfword binary value that is the length in bytes of the communication area.

PROGRAM(*name***)**
Specifies the program name (1-8 characters) of the CICS server application program to which control is to be passed unconditionally. The specified name must either have been defined as a program to CICS, or the CICS server region must be capable of autoinstalling a definition for the named program.

Note the use of quotes:

```
EXEC CICS LINK PROGRAM('PROGX')
```

PROGX is in quotes because it is the program name.

```
EXEC CICS LINK PROGRAM(DAREA)
```

DAREA is not in quotes because it is the name of a data area that contains the 8-character program name.

RETCODE(*data-area***)**
Specifies a 20-byte area into which the external CICS interface places return code information. This area is formatted into five one–word fields as follows:

RESP The primary response code indicating whether the external CICS interface LINK command caused an exception condition during its execution.

RESP2
The secondary response code that further qualifies, where necessary, some of the conditions raised in the RESP parameter.

ABCODE
Contains a valid CICS abend code if the server program abended in the server region.

MSGLEN
Indicates the length of the message (if any) issued by the CICS server region during the execution of the server program. Note that the length is the actual length of the message text only, and does not include this one—word length field.

MSGPTR
This is the address of the message text returned by the CICS server region.

Note: MSGLEN and MSGPTR are only valid on a LINKERR condition, with the RESP2 value 414.

SYNCONRETURN
Specifies that the CICS server region, named on the APPLID parameter, is to take a syncpoint on successful completion of the server program.

SYNONRETURN is mandatory for an external CICS interface LINK command.

TRANSID(*name*)
Specifies the name of the mirror transaction that the remote region is to attach, and under which it is to run the server program. If you omit the TRANSID option, the CICS server region attaches CSMI.

Note: The TRANSID option specified on the LINK command overrides any TRANSID option specified on the program resource definition installed in the CICS server region.

While you can specify your own name for the mirror transaction initiated by DPL requests, the transaction *must* be defined in the server region, and the transaction definition must specify the mirror program, DFHMIRS. Defining your own transaction to invoke the mirror program gives you the freedom to specify appropriate values for some other options on the transaction resource definition.

See also the important rules about specifying transid with a DPL_Request on page 129.

Error codes

Most of the exception conditions that are returned on the external CICS interface LINK command are the same as for the CICS-to-CICS distributed program link command. Those that are the same, and their corresponding numeric values are as follows:

INVREQ
16

LENGERR
22

PGMIDERR
27

SYSIDERR
53

NOTAUTH
70

TERMERR
81

ROLLEDBACK
82

These exception condition codes are returned in the RESP field.

RESP and RESP2

References to the RESP and RESP2 fields in this section are to the first two fields of the RETCODE parameter.

The exception conditions that are specific to the external CICS interface are as follows:

- The RESP2 values on the error conditions INVREQ and LENGERR are specific to the external CICS interface.
- The exception conditions WARNING and LINKERR are specific to the external CICS interface.

The WARNING and LINKERR exceptions are a result of responses to individual EXCI calls issued by the external CICS interface in response to an EXEC CICS LINK command. These WARNING and LINKERR exceptions correspond to EXCI call responses as follows:

WARNING (RESP value 4)

This is returned when the EXCI module handling the EXEC CICS LINK request receives a USER_ERROR or SYSTEM_ERROR response to a Close_Pipe or Deallocate_Pipe request issued on behalf of an EXEC CICS LINK command. The RESP value is set to WARNING because the DPL request to CICS completed successfully, but an error occurred in subsequent processing.

The RESP2 field is set to the EXCI reason code, which gives more information about the error.

LINKERR (RESP value 88)

This is returned when the EXCI module handling the EXEC CICS LINK request receives a RETRYABLE, USER_ERROR, or SYSTEM_ERROR response to an EXCI call issued on behalf of the EXEC CICS LINK command. The DPL request has failed. The RESP2 field is set to the EXCI reason code, which gives more information about the error.

See “Chapter 17. Response and reason codes returned on EXCI calls” on page 189 for descriptions of EXCI reason codes.

Note: The external CICS interface ignores any WARNING conditions that occur in response to EXCI calls it issues on behalf of an EXEC CICS LINK command. It treats the WARNING on an EXCI call as a good response and continues normally. If no other errors occur, the EXEC CICS command completes with a zero response in the EXEC_RESP field.

Retries on an EXEC CICS LINK command

If the external CICS interface receives a RETRYABLE response on an EXCI call that it makes on behalf of an EXEC CICS LINK command, it automatically retries the EXEC CICS LINK command up to five times, providing more serious errors do not occur. If the RETRYABLE response is still received after the fifth retry, the RESP field is set to LINKERR, and the reason returned on the EXCI CALL request that causes the exception is returned in the RESP2 field.

The external CICS interface retries the EXEC CICS LINK command by first closing and deallocating the pipe, then reissuing the six EXCI CALL commands. During Allocate_Pipe processing, the EXCI CALL interface calls the user-replaceable module, DFHXCURM, to give you the opportunity to change the APPLID of the CICS system to which the request has been sent. See “Chapter 12. The EXCI user-replaceable module” on page 157 for details of DFHXCURM.

Table 15 lists all the exception conditions and RESP2 values that are specific to the EXEC CICS LINK command for the external CICS interface.

Table 15. Exceptional conditions. RESP and RESP2 values returned from the EXEC API.

Condition (RESP)	RESP2	Meaning
INVREQ (16)	21	SYNCONRETURN has not been specified
LENGERR (22)	22	COMMAREA length greater than 32763 bytes specified
	23	COMMAREA specified but no LENGTH parameter specified
WARNING (4)	401	Invalid *call_type* parameter value specified on Close_Pipe or Deallocate_Pipe call
	402	Invalid *version_number* parameter specified on Close_Pipe or Deallocate_Pipe call
	404	Invalid *user_token* specified on Close_Pipe or Deallocate_Pipe call
	405	A Deallocate_Pipe call has been issued against a pipe that is not yet closed
	418	An invalid pipe token has been issued on a Close_Pipe or Deallocate_Pipe call
	610	There has been a CICS IRP logoff failure on a Deallocate_Pipe call
	611	There has been a CICS IRC disconnect failure on a Close_Pipe call
	622	There has been a VSE ESTAEX setup failure on a Close_Pipe or Deallocate_Pipe call
	623	A program check on a Close_Pipe or Deallocate_Pipe call has caused the ESTAEX to be invoked
LINKERR (88)	201	Command has been issued on a VSE image which has had no IRC activity since the previous IPL
	202	There are no available sessions
	203	CICS has not yet been brought up, or (2) has not yet opened IRC, or (3) no generic connection is installed, or (4) no specific connection is installed with the required netname.
	401	Invalid parameter
	402	Invalid version number
	403	User name is all blanks
	404	Invalid address in user token
	405	Command has been issued against a pipe that is not closed
	406	Command has been issued against a pipe that is not open
	407	Userid of all blanks has been passed

Table 15. Exceptional conditions (continued). RESP and RESP2 values returned from the EXEC API.

Condition (RESP)	RESP2	Meaning
	408	Error in UOWID parameter
	409	Transid consisting of all blanks or zero has been passed
	410	Load of message module, DFHMEBM, failed
	411	Load of message module, DFHMET4E, failed
	412	Load of DFHXCURM failed
	413	Load of DFHXCTRA failed
	414	If run as a CICS-to-CICS linked-to program, this server program would have resulted in an error with an appropriate message sent to the terminal. Running the program as an EXCI server program returns the message addressed by the MSGPTR field of the RETCODE area
	415	Target connection is an MRO connection, not an EXCI connection
	416	Command has been issued against a pre-CICS Transaction Server for VSE/ESA Release 1 system
	417	Command has been issued against a pipe in the MUST CLOSE state. Further EXCI EXEC CICS LINK commands will have unpredictable results and are, therefore, not permitted
	418	Pipe_token does not address an XCPIPE control block, or there is a mismatch between user_token and pipe_token
	419	CICS runs, or did run, under the VSE subtask that this command is attempting to use. This is not permitted asnd the command fails
	420	Load of DFHXCOPT failed
	422	The server has abended
	601	A VSE GETMAIN of working storage failed. This error leads to user abend 408
	602	A VSE GETMAIN failed. This error leads to user abend 403
	603	A VSE GETMAIN failed. This error leads to user abend 410
	604	A VSE GETMAIN failed
	605	A VSE GETMAIN for the VERIFY block failed. This error leads to user abend 409
	607	An SVC call failed. This error leads to user abend 406
	608	Logon to IRP failed
	609	Connect to IRP failed
	610	Disconnect from IRP failedl
	611	Logoff from IRP failed
	612	Invalid data input to transformer_1
	613	Invalid data input to transformer_4
	614	CICS has responded but has not sent any data
	615	CICS cannot satisfy the request
	616	IRP_SWITCH_PULL request (to read data sent from CICS into a larger input/output area) has failed
	617	A GETMAIN for a larger input/output area failed
	619	IRP has had a problem with the input/output area passed from the client program
	620	IRP has disconnected from EXCI
	621	A DISCONNECT command is issued in an error situation following an IRP CONNECT. The DISCONNECT has failed, indicating a serious error.
	622	XCPRH ESTAEX setup command failed This error leads to user abend 402.

Table 15. Exceptional conditions (continued). RESP and RESP2 values returned from the EXEC API.

Condition (RESP)	RESP2	Meaning
	623	XCPRH ESTAEX invoked due to program check during the processing of this command. ESTAEX attempts backout and takes a SYSDUMP. Further requests are permitted although the pipe is now in a MUST CLOSE state.
	624	The DPL request has been passed to CICS but the time specified in DFHXCOPT has been exceeded. The request is aborted.
	625	A VSE STIMERM macro call failed
	626	A VSE STIMERM CANCEL request failed
	627	The CICS SVC is at the incorrect level. This error leads to user abend 407.
	628	DFHIRP is at the incorrect level
	903	AN XCEIP ESTAEX setup command failed
	904	The server program abended with the abend code in the ABCODE field of the RETCODE area
	905	An XCEIP ESTAEX invoked due to program check during the processing of this command. ESTAEX attempts backout and takes a SYSDUMP

See “Return codes” on page 139 for details of the various copybooks that contain full details of all response and reason codes, including equated values.

Note: All numeric response and reason code values are shown in decimal.

Translation required for EXEC CICS LINK command

Application programs that use the EXEC CICS LINK form of the external CICS interface command must translate their programs before assembly or compilation. You do this using the version of the CICS translator that is appropriate for the language of your client program, specifying the translator option EXCI.

The translator option EXCI is mutually exclusive with the CICS and DLI options.

For more information about translating programs that contain EXEC CICS commands, see the *CICS Application Programming Guide*.

For information about compiling and link-editing external CICS interface client programs, see page 161.

Chapter 11. Defining connections to CICS

Connections between an EXCI client program and a CICS region require connection definitions in the CICS region. You define these using the CONNECTION and the SESSIONS resource definition facilities provided by CICS.

The following options are provided specifically for the external CICS interface:

- CONNTYPE on the CONNECTION resource definition
- EXCI on the PROTOCOL attribute of the CONNECTION and SESSIONS resource definitions.

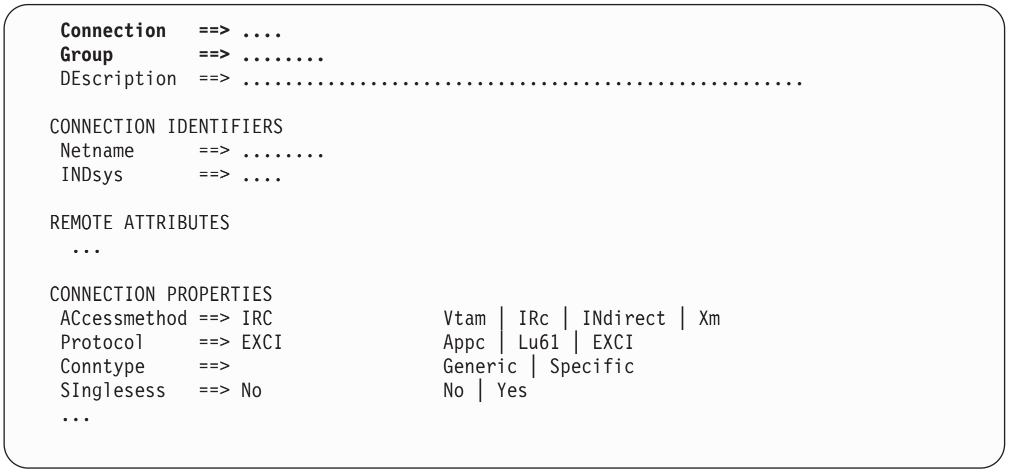
The following topics are covered in this chapter:

- "CONNECTION resource definition"
- "SESSIONS resource definitions for EXCI connections" on page 153
- "Inquiring on the state of EXCI connections" on page 155

CONNECTION resource definition

The EXCI option is provided on the PROTOCOL attribute of the CONNECTION resource definition to indicate that the connection is for use by a VSE program using the external CICS interface.

The CONNTYPE attribute is provided on the CONNECTION resource definition. For EXCI connections, this indicates whether the connection is generic or specific. It is not to be used for any protocol other than the external CICS interface.



```
 Connection   ==> ....
 Group        ==> ........
 DEscription  ==> ..................................................

CONNECTION IDENTIFIERS
 Netname      ==> ........
 INDsys       ==> ....

REMOTE ATTRIBUTES
  ...

CONNECTION PROPERTIES
 ACcessmethod ==> IRC               Vtam | IRc | INdirect | Xm
 Protocol     ==> EXCI              Appc | Lu61 | EXCI
 Conntype     ==>                   Generic | Specific
 SInglesess   ==> No                No | Yes
 ...
```

Figure 30. The DEFINE panel for CONNECTION

CONNTYPE({SPECIFIC|GENERIC})
For external CICS interface connections, indicates the nature of the connection.

SPECIFIC
The connection is for communication from a non-CICS client program to the CICS region, and is specific. A specific connection is an MRO link with one or more sessions dedicated to a single user in a client program.

Note: A *user* is a program that has issued an Initialize_User request (or for which an Initialize_User request has been issued), with a unique name per VSE task. For example:

- A simple client program running under VSE can be a single user of the external CICS interface.
- A client program running under VSE can open several pipes and issue external CICS interface calls over them sequentially, on behalf of different vendor packages. In this case, from the viewpoint of the client program, each of the packages is a user, identified by a unique user name. Thus a single client program can operate on behalf of multiple users.
- A program running under VSE can attach several VSE subtasks, under each of which a vendor package issues external CICS interface calls on its own behalf. Each package is a client program in its own right, and runs under its own VSE subtask. Each is also a user, with a unique user name.

For a specific connection, NETNAME is mandatory.

GENERIC
The connection is for communication from a non-CICS client program to the CICS system, and is generic. A generic connection is an MRO link with a number of sessions to be shared by multiple EXCI users. For a generic connection you cannot specify the NETNAME attribute.

Note: You must install only one generic EXCI connection in a CICS region.

NETNAME
For an external CICS interface connection, NETNAME corresponds to the name of the user of a specific pipe, as specified on the *user_name* parameter of an INITIALISE_USER call.

For an external CICS interface specific pipe, you must specify a NETNAME.

For external CICS interface generic pipes, you must leave NETNAME blank.

PROTOCOL({APPC|LU61|EXCI|blank})
The type of protocol that is to be used for the link.

blank
For MRO between CICS regions. You must leave the PROTOCOL blank for MRO, and on the SESSIONS definition you must specify LU6.1 as the PROTOCOL.

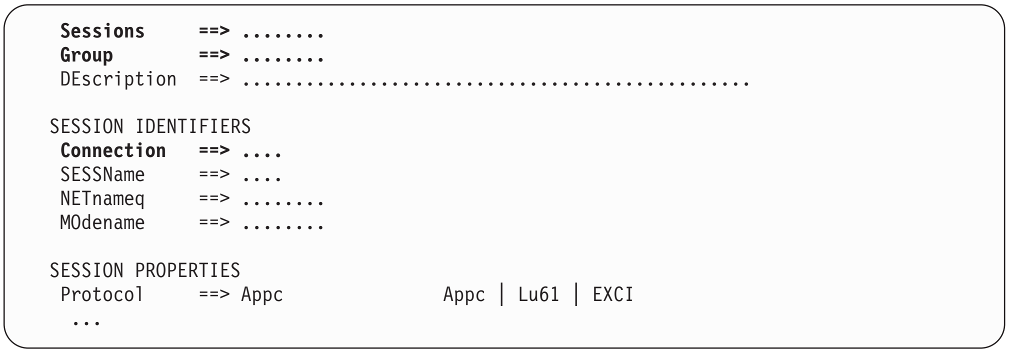
APPC (LUTYPE6.2 protocol)
Advanced program-to-program communication, or APPC protocol. This is the default value for ACCESSMETHOD(VTAM). Specify this for CICS-CICS ISC.

LU61
LUTYPE6.1 protocol. Specify this for CICS-CICS ISC or CICS-IMS ISC, but not for MRO.

EXCI
The external CICS interface. Specify this to indicate that this connection is for use by a non-CICS client program using the external CICS interface.

SESSIONS resource definitions for EXCI connections

You indicate on the PROTOCOL attribute of the SESSIONS resource definition whether the sessions allocated on the MRO connection are for use by the external CICS interface.



```
 Sessions     ==> ........
 Group        ==> ........
 DEscription  ==> ................................................

SESSION IDENTIFIERS
 Connection   ==> ....
 SESSName     ==> ....
 NETnameq     ==> ........
 MOdename     ==> ........

SESSION PROPERTIES
 Protocol     ==> Appc               Appc | Lu61 | EXCI
  ...
```

Figure 31. The DEFINE panel for SESSIONS

PROTOCOL({APPC|LU61|EXCI})

Indicates the type of protocol that is to be used for an intercommunication link (ISC or MRO).

APPC (LUTYPE6.2)

Advanced program-to-program communication (APPC) protocol. Specify this for CICS-CICS ISC.

LU61

LUTYPE6.1 protocol. Specify this for CICS-CICS ISC, for CICS-IMS, or for MRO.

EXCI

The external CICS interface. Specify this to indicate that the sessions are for use by a non-CICS client program using the external CICS interface. If you specify EXCI, you must leave SENDCOUNT blank.

RECEIVECOUNT({blank|*number*})

The number of MRO, LUTYPE6.1, or EXCI sessions that usually receive before sending.

For MRO, receive sessions can only receive before sending.

blank

These sessions can send only; there are no receive sessions.

number

Specifies the number of receive sessions on connections that specify blank, LU61, or EXCI on the protocol parameter of the CONNECTION definition. CICS uses the number to generate the last two or three characters of the session names (see RECEIVEPFX for details).

If you are using the default receive prefix (<), or your own 1-character prefix, specify a number in the range 1 through 999.

If you specify a 2-character prefix, the number is restricted to the range 1 through 99.

Except for external CICS interface (EXCI) connections, the RECEIVECOUNT in this system should equal SENDCOUNT in the other system.

RECEIVEPFX(<|*prefix*)

Specifies a 1- or 2-character prefix that CICS is to use as the first 1 or 2 characters of the receive session names (the names of the terminal control table terminal entries (TCTTEs) for the sessions).

Prefixes must not cause a conflict with an existing connection or terminal name.

< (MRO and EXCI sessions)

For MRO sessions, if you do not specify your own receive prefix, CICS enforces the default prefix—the less-than symbol (<), which is used in conjunction with the receive count to generate receive session names.

CICS creates the last three characters of the session names from the alphanumeric characters A through Z, and 1 through 9. These 3-character identifiers begin with the letters AAA, and continue in ascending sequence until the number of session entries reaches the limit set by the RECEIVECOUNT value. Note that receive session names are generated *after* the send sessions, and they follow in the same sequence.

For example, if the last session name generated for the send sessions is <AAJ, using the default prefix (<) CICS generates the receive session names as <AAK, <AAL, <AAM, and so on. (This method of generation of session identifiers is the same as for APPC sessions, except for the initial prefix symbol.)

Note: If you specify your own prefix, CICS generates the session names as in earlier releases, which is the same as for LUTYPE6.1 sessions.

prefix **(LUTYPE6.1 sessions)**

If the sessions are on LUTYPE6.1 ISC connections, you must specify a 1- or 2-character prefix. Do not use the default < symbol for LUTYPE6.1 sessions.

For LUTYPE6.1 sessions (and MRO if you specify your own 1- or 2-character prefix) CICS generates session names by appending a number to the prefix, either in the range 1 through 99, or 1 through 999. The number begins with 1 and is incremented by 1 until the specified RECEIVECOUNT is reached.

SENDCOUNT(blank|*number*)

The number of MRO or LUTYPE6.1 sessions that usually send before receiving.

For MRO, send sessions must send before they can receive.

blank

These sessions can receive only; there are no send sessions.

You must leave this field blank when the sessions are on an external CICS interface (EXCI) connection.

number

Specifies the number of send sessions on connections that specify blank or

LU61 on the protocol parameter of the CONNECTION definition. CICS uses the number to generate the last two or three characters of the session names (see SENDPFX for details).

If you are using the default send prefix (>), or your own 1-character prefix, specify a number in the range 1 through 999.

If you specify a 2-character prefix, the number is restricted to the range 1 through 99.

Except for external CICS interface (EXCI) connections the SENDCOUNT in the sending system should equal RECEIVECOUNT in the receiving system.

SENDPFX(>|*prefix*)

Specifies a 1- or 2-character prefix that CICS is to use as the first 1 or 2 characters of the send session names (the names of the terminal control table terminal entries (TCTTEs) for the sessions).

Prefixes must not cause a conflict with an existing connection or terminal name.

> (MRO sessions)

For MRO sessions, if you do not specify your own send prefix, CICS enforces the default prefix—the greater-than symbol (>), which is used in conjunction with the send count to generate send session names.

CICS creates the last three characters of the session names from the alphanumeric characters A through Z, and 1 through 9. These 3-character identifiers begin with the letters AAA, and continue in ascending sequence until the number of session entries reaches the limit set by the SENDCOUNT value.

For example, using the default prefix (>) CICS generates session names as >AAA, >AAB, >AAC, and so on. (This method of generation of session identifiers is the same as for APPC sessions, except for the initial symbol.)

Note: If you specify your own prefix, CICS generates the session names as in earlier releases, which is the same as for LUTYPE6.1 sessions.

prefix **(for LUTYPE6.1 sessions)**

If the sessions are on LUTYPE6.1 ISC connections, you must specify a 1- or 2-character prefix. Do not use the default > symbol for LUTYPE6.1 sessions.

For LUTYPE6.1 sessions (and MRO if you specify your own 1- or 2-character prefix) CICS generates session names by appending a number to the prefix, either in the range 1 through 99, or 1 through 999. The number begins with 1 and is incremented by 1 until the specified SENDCOUNT is reached.

Inquiring on the state of EXCI connections

If you have access, through a CICS terminal, to the CICS server region, you can inquire about batch jobs that are running a client application program, and which are using the external CICS interface to link to a server program in CICS.

To obtain this information about batch jobs linked to CICS through MRO, you use the CEMT INQUIRE EXCI command. This command enables you to identify the names of external CICS interface batch jobs currently connected to CICS through the interregion communication (IRC) facility.

CICS returns job identifications in the form:

```
jobname.execprogramname.partitionid.procname - vseid
```

The `vseid` identifies the VSE system on which the job is running.

Information about jobs using the external CICS interface is available only when the job has issued at least one DPL request. A non-zero task number indicates that a DPL request is currently active. A zero task number indicates an external CICS interface session is still open (connected) for that job, although no DPL request is currently active.

See the *CICS Supplied Transactions* manual for more information about the CEMT command.

Chapter 12. The EXCI user-replaceable module

This chapter contains Product-sensitive Programming Interface information.

The external CICS interface provides a user-replaceable module, DFHXCURM. The load module is supplied in PRD1.SDFHEXCI, and the source in PRD1.BASE.

DFHXCURM is invoked in the non-CICS region during the processing of Allocate_Pipe commands, and after the occurrence of any retryable error. The retryable responses are:

- The target CICS region is not available
- There are no pipes available on the target CICS region
- There has been no IRC activity since the VSE/ESA IPL.

As supplied, DFHXCURM is effectively a dummy program because of a branch instruction that bypasses the sample logic and returns control to the external CICS interface caller. To use the sample logic, remove the branch instruction and assemble and link-edit the module. Customizing DFHXCURM allows you to do the following:

- When invoked during Allocate_Pipe processing, you can change the specified CICS APPLID, in order to route the request to another CICS system.
- When invoked after a retryable error you can store information regarding CICS availability. You can then use this information on the next invocation of DFHXCURM for Allocate_Pipe processing, so that you can decide to which CICS system to route the request.

DFHXCURM is called using standard VSE/ESA register conventions, with register 1 containing the address of the parameter list, and register 14 the return address of the caller. The parameters addressed by register 1 are mapped in the EXCI_URM_PARMS DSECT, which is contained within the DFHXCPLD copybook. The parameters passed to DFHXCURM are as follows:

URMINV

The address of a fullword that contains the reason for the invocation of DFHXCURM, defined by the following equates:

```
URM_ALLOCATE     EQU 1  This invocation is for an Allocate_Pipe
URM_NO_CICS      EQU 2  The target CICS region is not available
URM_NO_PIPE      EQU 3  There are no pipes available
URM_NO_CICS_IRC  EQU 4  There has been no IRC activity since the VSE/ESA IPL
```

URMCICS

The address of an 8-byte area that contains the generic APPLID of the target CICS system, as specified on the *CICS_applid* parameter of the Allocate_Pipe command, or on the APPLID parameter of the EXEC CICS LINK command.

When specified by one of these commands, you can change the APPLID to that of a different target CICS region.

If the *CICS_applid* parameter is omitted from the Allocate_Pipe call, or APPLID is omitted from the EXEC CICS LINK command, the field addressed by this parameter contains 8 blanks. In this case, you must specify an APPLID in DFHXCURM before returning control to the caller.

URMAPPL

The address of an 8-byte area that contains the client program's user name as specified on the *my_name* parameter of the Initialize_User command. Note that if DFHXCURM is invoked for an EXEC CICS LINK command, this name is always set to DFHXCEIP.

URMPROG
The address of an 8-byte area that contains the name of the target program (if available). This name is available only if DFHXCURM is invoked for an EXEC CICS LINK command. For an external CICS interface Allocate_Pipe command, the program name is not known until the DPL call is issued.

URMOPTS
The address of a 1-byte area that contains the allocate options, which can be X'00' or X'80', as specified on the *allocate_opts* parameter. This address is valid for an Allocate_Pipe request only.

URMANCH
The address of a 4-byte area that is provided for use by DFHXCURM only. A typical use for this is to store a global anchor address of an area used to save information across a number of invocations of DFHXCURM. For example, you can GETMAIN the necessary storage and save the address in the 4-byte area addressed by this parameter. The initial value of the 4-byte area is set to zero.

Chapter 13. External CICS interface options table, DFHXCOPT

The EXCI options table, generated by the DFHXCOPT macro, enables you to specify a number of parameters that are required by the external CICS interface.

CICS provides the default DFHXCOPT table in source form, which you can tailor to your own requirements. The source of the default table and the load module are supplied in PRD1.BASE.

You assemble and link-edit the modified DFHXCOPT table into a suitable library in the LIBDEF PHASE, SEARCH library chain of the job that runs the VSE/ESA client program. You can use your own version of the CICS DFHAUPLE procedure to assemble and link-edit your customized options table. The DFHAUPLE procedure is supplied in PRD1.SDFHINST. Unlike the tables you specify for CICS regions, the DFHXCOPT table cannot be suffixed, and the external CICS interface component loads the first table of this name that it finds in the STEPLIB concatenation. Unlike the tables you specify for CICS regions, the DFHXCOPT table cannot be suffixed, and the external CICS interface component loads the first table of this name that it finds in the LIBDEF PHASE, SEARCH library chain.

Table 16 shows the format of the DFHXCOPT macro and its parameters.

Table 16. The DFHXCOPT macro parameters

	DFHXCO	TYPE={**CSECT**|DSECT} [,MSGCASE={**MIXED**|UPPER}] [,TIMEOUT={**0**|*number*}] [,TRACE={**OFF**|1|2}] [,TRACESZE={**16**|*number-of-kilobytes*}] [,TRAP={**OFF**|ON}] You must terminate your parameters with the following END statement.
	END	DFHXCOPT

TYPE={CSECT|DSECT}
Indicates the type of table to be generated.

CSECT
A regular control section that is normally used.

DSECT
A dummy control section.

MSGCASE={MIXED|UPPER}
Specifies whether the DFHEX*xxxx*messages are to be issued in mixed or uppercase.

MIXED
Code this if messages are to be issued in mixed case.

UPPER
Code this if messages are to be issued in uppercase.

TIMEOUT={0|*number*}
Specifies the time interval, in hundredths of a second, during which the external CICS interface waits for a DPL command to complete.

0 Specifies that you do not want any time limit applied, and that the external CICS interface is to wait indefinitely for a DPL command to complete.

number
Specifies the time interval, in hundredths of a second, that the external CICS interface is to wait for a DPL command to complete. The number represents hundredths of a second, from 1 up to a maximum of 2 147 483 647. For example:

6000	Represents a timeout value of one minute
30000	Represents a timeout value of five minutes
60000	Represents a timeout value of ten minutes.

TRACE={OFF|1|2}
Specifies whether you want external CICS interface internal tracing, and at what level.

OFF
External CICS interface internal tracing is not required. However, even with normal tracing switched off, exception trace entries are always written to the internal trace table.

1 Exception and level-1 trace entries are written to the internal trace table.

2 Exception, level-1, and level-2 trace entries are written to the internal trace table.

TRACESZE={16|*number-of-kilobytes*}
Specifies the size in kilobytes of the internal trace table for use by the external CICS interface. This table is allocated in virtual storage above the 16MB line, if available.You should ensure that there is enough virtual storage for the trace table by specifying a large enough partition size on the VSE/ESA ALLOC parameter.

16 16KB is the default size of the trace table, and also the minimum size.

number-of-kilobytes
The number of kilobytes of storage to be allocated for the internal trace table, in the range 16KB through 1 048 576KB. Subpool 1 is used for the trace table storage, which exists for the duration of the jobstep VSE subtask. The table is page-aligned and occupies a whole number of pages. If the value specified is not a multiple of the page size (4KB), it is rounded up to the next multiple of 4KB.

TRAP={OFF|ON}
Specifies whether the service trap module, DFHXCTRA, is to be used. DFHXCTRA is supplied as a user-replaceable module, in which IBM service personnel can add code to trap errors.

OFF
Code this if you do not want to use DFHXCTRA.

ON
Code this if you require DFHXCTRA.

Chapter 14. Compiling and link-editing external CICS interface client programs

This chapter discusses the following topics:

- The external CICS interface stub, DFHXCSTB
- The CICS-supplied procedures for the external CICS interface
- Language considerations
- Sample application programs
- Job control language to run an EXCI client program.

The external CICS interface stub, DFHXCSTB

All programs that use the external CICS interface to pass DPL requests to a CICS server region must include the CICS-supplied program stub, DFHXCSTB.

The stub intercepts all external CICS interface commands, whether they are EXCI CALL interface commands, or EXEC CICS LINK commands, and ensures they are passed to the appropriate external CICS interface routine for processing.

DFHXCSTB is a common stub, designed for inclusion in programs written in all the supported languages. It is supplied in the PRD1.BASE library.

Note: The PRD1.BASE also contains entries for DFHXCIE and DFHXCIS, which are aliases for DFHXCSTB.

The required linkage editor modes

You must specify AMODE(31) for your EXCI client program.

Language considerations

There are some language requirements that apply to writing a VSE/ESA client program that uses the external CICS interface. These affect programs written in PL/I and C.

PL/I considerations

PL/I programs written to the external CICS interface must provide their parameters on the CALL to DFHXCIS in the form of an assembler-style parameter list.

The EXCI copybook for PL/I, DFHXCPLL, contains the necessary definition of the DFHXCIS entry point, as follows:

```
 DCL DFHXCIS  ENTRY        OPTIONS(INTER ASSEMBLER);
```

The same rule applies for the EXCI LINK command, and in this case the CICS translator ensures that the correct parameter list is built.

For an example of an EXCI client program written in PL/I, see the source of the sample program, DFH$PXCC.

C considerations

C programs written to the external CICS interface must provide their parameters on the CALL to DFHXCIS in the form of an assembler-style parameter list. You ensure this by declaring the entry point to DFHXCIS with OS LINKAGE.

The EXCI copybook for C, DFHXCPLH, contains the necessary definition of the DFHXCIS entry point, as follows:

```
#pragma linkage(dfhxcis,OS)
```

The same rule applies for the EXCI LINK command, and in this case the CICS translator ensures that the correct parameter list is built.

For an example of an EXCI client program written in C, see the source of the sample program, DFH$DXCC.

Sample application programs

CICS provides a number of sample programs that are designed to help you in writing your own application programs. To help with writing programs that use the external CICS interface, CICS provides a sample VSE/ESA client program and a sample CICS server program.

The samples show you how to code client applications that use both the EXCI CALL interface and EXEC CICS LINK command.

Description of the sample applications

The sample external CICS interface programs are included on the CICS Transaction Server for VSE/ESA Release 1 base tape.

The sample VSE/ESA client program is provided in Assembler language, COBOL for VSE, Cfor VSE, and PL/I for VSE. The sample CICS server program is provided in assembler only. Assembler language programs are in source and executable form. COBOL, PL/I, and C programs are provided in source form only. Each version of the client program has basically the same function, but programming methods vary somewhat according to the language used.

The sample programs, shown in Table 17, are supplied in source form in PRD1.BASE. The sample assembler server program is also supplied in executable form in PRD1.BASE. The assembler client program is supplied in PRD1.BASE.

Note: The assembler versions of the client program use SAM, which requires the programs to be link-edited in RMODE(24). The assembler source code includes the required RMODE(24) statement. Normally, EXCI client programs run AMODE(31),RMODE(ANY).

Table 17. The external CICS interface sample programs

Language	Name	Type of program
Assembler	DFH$AXCC	Client program
Assembler	DFH$AXCS	Server program
COBOL	DFH0CXCC	Client program
PL/I	DFH$PXCC	Client program
C	DFH$DXCC	Client program

Each version of the client is divided into three separate sections as follows:

1. The first section issues a single EXEC CICS LINK command to inquire on the state of the sample VSAM file, FILEA, in the target CICS system.

If the file is in a suitable state, processing continues to sections two and three, which together provide complete examples of the use of the EXCI CALL interface.

2. The second section initiates a specific MRO connection to the target CICS system and, once the pipe is open, performs a series of calls that each retrieve a single sequential record from the sample VSAM file, until no more records are available.
3. The third section is a simple routine to close the target sample file once processing of the data is complete. It also terminates the MRO connection now that the link is no longer required.

Some of the parameters used on the EXCI CALL and EXEC CICS LINK commands in the client program need to be tailored for your own target CICS server region. Change these as required, then retranslate, compile (or assemble), and link-edit the program.

The variables and their values specified in the sample programs are given in Table 18.

Table 18. Parameters used in the sample client programs

Variable name in sample program	Default value
TARGET_FILE	FILEA
TARGET_TRANSID	EXCI
TARGET_SYSTEM	DBDCCICS (applid)
TARGET_PROGRAM	DFH$AXCS
APPLICATION	BATCHCLI

The assembler versions of the client programs are supplied pregenerated in an executable form. All versions of the program accept a run-time parameter to specify the target server region APPLID. For the pregenerated assembler version this avoids you having to reassemble the program to specify the applid of your own CICS server region. You can also use the sample client program with different CICS regions without needing to modify the program each time.

Installing the EXCI sample definitions

Resource definitions that support the EXCI sample programs are included in the CICS system definition file (CSD) in groups DFH$EXCI and DFH$FILA.

Note that the sample definitions, while included in the CSD, are not included in the IBM-defined group list DFHLIST. Thus, if CICS is initialized with GRPLIST=DFHLIST, you must install the EXCI resource definition groups before using the samples. Alternatively, you can add the sample groups to your startup group list, so that they are installed automatically at system initialization.

The resource definition groups that must be installed are as follows:

DFH$EXCI

This contains definitions for the sample server transaction, server program, EXCI connections, and sessions.

Only one server program is included—in assembler language, called DFH$AXCS.

The sample application is designed to run the transaction EXCI, which is defined to invoke the DFHMIRS mirror program and references profile DFHCICSA. The required transaction definition for EXCI is included in the group.

Sample CONNECTION and SESSIONS definitions for specific and generic connections are included.

Note: Both the generic and specific connection definitions supplied in the sample group DFH$EXCI specify ATTACHSEC(IDENTIFY). This security option causes the server program DFH$EXCS to fail with an ATCY abend if you run the sample programs in an environment that does not have an external security manager (ESM) installed and active.

If you want to run the external CICS interface sample programs without any security active, you must alter the connection resource definitions to specify ATTACHSEC(LOCAL).

DFH$FILA

This contains the definition for the supplied sample VSAM file, FILEA, which is referenced by the EXCI sample programs.

Once these are installed, you must ensure that interregion communication (IRC) is open. If IRC is not opened during CICS initialization, set it open using the CEMT SET IRC OPEN command.

Running the EXCI sample applications

If you want to use the COBOL, PL/I, or C version of the EXCI client program, you must translate, compile, and link-edit the program into a suitable library.

You can use the sample JCL shown in Figure 36 on page 167 as a basis for creating your own batch job to run the client program. Note the use of the OS390 parameter on the EXEC statement: this is mandatory.

If you use the pregenerated assembler version, specify the APPLID of your target CICS server region as a parameter on the EXEC statement for the client program, as follows:

```
//*==============================================================*
// EXEC  PGM=DFH$AXCC,PARM='applid',OS390
```

Results of running the EXCI sample applications

An example of the output produced by successful execution of the pregenerated assembler version of the client program, DFH$AXCC, is shown in Figure 32 on page 165.

If an error occurs while running the application, then, assuming the error is not severe, messages are written to the SYSLST output log displaying the reasons and/or return codes that cause processing to be terminated. Several examples of error-invoked output are shown in Figure 33, Figure 34, and Figure 35 on page 166.

```
*===================== EXCI Sample Client Program =============================*
*                                                                              *
*  EXEC Level Processor.                                                       *
*    Setting up the EXEC level call.                                           *
*    The Link Request has successfully completed.                              *
*    Server Response:                                                          *
*      The file is set to a browsable state.                                   *
*                                                                              *
*  CALL Level Processor.                                                       *
*    Initialize_User call complete.                                            *
*    Allocate_Pipe call complete.                                              *
*    Open_Pipe call complete.                                                  *
*    The connection has been successful.                                       *
*      The target file follows:                                                *
*                                                                              *
*=========================== Top of File ======================================*
000102F. ALDSON            WARWICK, ENGLAND    9835618326 11 81$1111.11Y00007300
000104S. BOWLER            LONDON,ENGLAND      1284629326 11 81$0999.99Y00007400
000106B. ADAMS             CROYDON, ENGLAND    1948567326 11 81$0087.71Y00007500
000111GENE BARLOWE         SARATOGA,CALIFORNIA 4612075301 02 74$0111.11Y00007600
000762GEORGE BURROW        SAN JOSE,CALIFORNIA 2231212101 06 74$0000.00Y00007700
000983H. L. L. CALL        WASHINGTON, DC      3451212021 04 75$9999.99Y00007800
003210B.CREPIN             NICE, FRANCE        1234567026 11 81$3349.99Y00008100
003214HUBERT C HERBERT     SUNNYVALE, CAL.     3411212000 06 73$0009.99N00008200
003890PHILIPPE SMITH, JR   NICE, FRANCE        0000000028 05 74$0009.99N00008300
004004STAN SMITH           DUBLIN, IRELAND     7111212102 11 73$1259.99N00008400
004445S. GALSON            SOUTH BEND, S.DAK.  6121212026 11 81$0009.99N00008500
004878D.C. CURRENT         SUNNYVALE, CALIF.   3221212010 06 73$5399.99N00008600
005005J. S. LAVERENCE      SAN FRANCISCO, CA.  0000000101 08 73$0009.99N00008700
005444JEAN LAWRENCE        SARATOGA, CALIF.    6771212020 10 74$0809.99N00008800
005581JOHN ALDEN III       BOSTON, MASS.       4131212011 04 74$0259.99N00008900
006016DR W. T. KAR         NEW DELHI, INDIA    7033121121 05 74$0009.88Y00009000
006670WILLIAM KAPP         NEW YORK, N.Y.      2121212031 01 75$3509.88N00009100
06968D. CONRAD             WARWICK, ENGLAND    5671382126 11 81$0009.88Y00009200
007248B. C. WILLIAMSON     REDWOOD CITY, CALF. 3331212111 10 75$0009.88N00009400
007779MRS. W. WELCH        SAN JOSE, CALIF.    4151212003 01 75$0009.88Y00009500
100000G. NEADS             TORONTO, ONTARIO    0341512126 11 81$0010.00Y00009600
111111C. MEARS             OTTAWA, ONTARIO     5121200326 11 81$0011.00Y00009700
200000A. BONFIELD          GLASCOW, SCOTLAND   6373829026 11 81$0020.00Y00009900
300000K. TRENCHARD         NEW YORK, U.S.      6473980126 11 81$0030.00Y00010000
333333D. MYRING            CARDIFF, WALES      7849302026 11 81$0033.00Y00010100
400000W. TANNER            MILAN, ITALY        2536373826 11 81$0040.00Y00010200
444444A. FISHER            CALGARY, ALBERTA    7788982026 11 81$0044.00Y00010300
500000J. DENFORD           MADRID, SPAIN       4445464026 11 81$0000.00Y00010400
555555C. JARDINE           KINGSTON, N.Y.      3994442026 11 81$0005.00Y00010500
600000F. HUGHES            DUBLIN, IRELAND     1239878026 11 81$0010.00Y00010600
666666A. BROOKMAN          LA HULPE, BRUSSELS  4298384026 11 81$0016.00Y00010700
700000A. MACALLA           DALLAS, TEXAS       5798432026 11 81$0002.00Y00010800
777777D. PRYKE             WILLIAMSBURG, VIRG. 9187613126 11 81$0027.00Y00010900
800000H. BRISTOW           WESTEND, LONDON     2423338926 11 81$0030.00Y00011000
888888B. HOWARD            NORTHAMPTON, ENG.   2369163926 11 81$0038.00Y00011100
900000D. WOODSON           TAMPA, FLA.         3566812026 11 81$0040.00Y00011200
999999R. JACKSON           RALEIGH, N.Y.       8459163926 11 81$0049.00Y00011300
 *=========================== End of File ======================================*
 *                                                                              *
 *    Closing Dpl Request has been attempted.                                   *
 *    Close_Pipe call complete.                                                 *
 *    Deallocate_Pipe call complete.                                            *
 *                                                                              *
 *=================== End of EXCI Sample Client Program ========================*
```

Figure 32. Successful execution

```
*===================== EXCI Sample Client Program =============================*
*                                                                              *
*  EXEC Level Processor.                                                       *
*    Setting up the EXEC level call.                                           *
*    The Link Request has failed.  Return codes are;                           *
*        Resp = 00000088  Resp2 = 00000203  Abend Code:                        *
*    >>>> Aborting further processing <<<<                                     *
*                                                                              *
*=================== End of EXCI Sample Client Program ========================*
```

Figure 33. No CICS return code. The target CICS region specified by the client program is not found, or IRC was not opened.

```
*===================== EXCI Sample Client Program =============================*
*                                                                              *
*  EXEC Level Processor.                                                       *
*    Setting up the EXEC level call.                                           *
*    The Link Request has successfully completed.                              *
*    Server Response:                                                          *
*      The file could not be found.                                            *
*    >>>> Aborting further processing <<<<                                     *
*                                                                              *
*=================== End of EXCI Sample Client Program ========================*
```

Figure 34. No file found. The target file name to the server program was not found on the target CICS system.

```
*===================== EXCI Sample Client Program =============================*
*                                                                              *
*  EXEC Level Processor.                                                       *
*    Setting up the EXEC level call.                                           *
*    The Link Request has failed.  Return codes are;                           *
*        Resp = 00000088  Resp2 = 00000414  Abend Code:                        *
*    A message was received from the target CICS system:                       *
*                                                                              *
 DFHAC2001 04/29/93 16:43:03 IYAHZCAZ Transaction 'BAD_' is unrecognized.  Check
 that the transaction name is correct.
*                                                                              *
*    >>>> Aborting further processing <<<<                                     *
*                                                                              *
*=================== End of EXCI Sample Client Program ========================*
```

Figure 35. Incorrect transaction identifier. The target transid passed in the external CICS interface call is not defined on the target CICS system. Note the message received from the target CICS system.

Job control language to run an EXCI client program

An EXCI client program runs in a VSE partition, for example, as a batch job. Note the following requirements when writing the JCL for your client program:

- Include in the LIBDEF PHASE, SEARCH statement those libraries that contain the CICS-supplied external CICS interface modules and also the client program. The external CICS interface modules are supplied in PRD1.BASE. These are:

  DFH$AXCC
  DFHMEBM
  DFHMET4E
  DFHXCEIX
  DFHXCOPT
  DFHXCPRX

DFHXCSTB
DFHXCTRA
DFHXCURM

- You are recommended to include a LIBDEF DUMP,CATALOG=*library.sublibrary*. The external CICS interface uses SYSDUMP for some error conditions.
- The EXCI job must run in a partition large enough to allow for the size of the internal trace table specified by the TRACESZE parameter in the DFHXCOPT options table.

Figure 36 shows a sample job that you can use or modify to start a client program.

```
// JOB EXCI    accounting_information
//*===============================================================*
//*    JCL to execute an external CICS interface client program   *
//*===============================================================*
// ID USER=userid,PWD=password
// LIBDEF   PHASE,SEARCH=(lib.sublib,lib2,lib3,...)
// LIBDEF   DUMP,CATALOG=(lib.sublib)
//   EXEC  PGM=pgmname,OS390
```

Figure 36. Sample job for starting an EXCI client program

Notes:

1. The job userid,, specified on the USER parameter of the ID statement in the batch job JCL, must be defined to an external security manager (ESM) if batch security is used.
2. In addition to being used for job step initiation security, the job userid is also used for MRO logon and bind-time security checking.

   See “Chapter 15. Security” on page 169 for information about security when using the external CICS interface.
3. See “Installing the EXCI sample definitions” on page 163 for information about modifying the sample connection definitions before you run the sample application programs in an environment that does not have an external security manager (ESM) installed and active.

Chapter 15. Security

CICS applies security checks in a number of ways against requests received from a VSE client program. These security checks use the security authorization facility (SAF) interface, and require the services of an external security manager (ESM). The security checks are described in the following topics:

- MRO logon and bind-time security, performed by DFHIRP
- Link security, performed by the CICS server region
- User security in the server application program
- Surrogate user checking performed by the external CICS interface in the client program address space.

MRO logon and bind-time security

DFHIRP, the CICS interregion communication program, performs two security checks against users that want to:

1. Log on to IRP (**specific connections only**)
2. Connect to a CICS region (also referred to as bind-time security).

Generic EXCI connections

The discussion about logon security checking in this chapter applies only to EXCI connections that are defined as SPECIFIC. The MRO logon security check is not performed for generic connections.

The VSE client program is treated just the same as another CICS region as far as MRO logon and connect (bind-time) security checking is concerned. This means that when the client program logs on to the interregion communication program, IRP performs logon and bind-time security checks against the userid under which the client program is running. In the remainder of this chapter, we refer to this as the batch region's userid.

To enable your client program to log on successfully to IRP, and to connect to the target server region, first ensure that you define the batch region's user ID in a user profile to the ESM. When you have defined the batch region's userid to the ESM, you can then give the batch region the appropriate logon and bind-time authorizations.

1. Logon authorization

Authorize the batch region's userid to the DFHAPPL.*user_name* defined on the INITIALIZE_USER command. Generally, depending on the ESM, the batch region requires UPDATE authoirty to the relevant IRP logon security profile.

Failure to authorize the batch region's userid to logon to IRP causes Allocate_Pipe processing to fail with RESPONSE(SYSTEM_ERROR) and REASON(IRC_LOGON_FAILURE). The subreason field-1 for a logon security check failure returns decimal 204.

2. Bind-time authorization

Ensure the batch region's userid has the appropriate authority to connect to the target CICS server region. Generally, depending on the ESM, the batch region requires READ authority to the relevant IRP bind security profile.

Failure to authorize the batch region's userid to connect to the CICS server region causes Open_Pipe processing to fail with

RESPONSE(SYSTEM_ERROR) and REASON(IRC_CONNECT_FAILURE). The subreason field-1 for a bind-time security check failure returns decimal 176.

See the ESM's documentation for information about how to specify MRO logon and bind-time security checks.

Link security

The target CICS server region performs link security checking against requests from the client program. These security checks cover transaction attach security (when attaching the mirror transaction), and resource and command security checking within the server application program. The link userid that CICS uses for these security checks is the batch region's userid.

To ensure these link security checks do not cause security failures, you must ensure that the link user ID is authorized to the following resource profiles, as appropriate:

- The profile for the mirror transaction, either CSMI for the default, or the mirror transaction specified on the *transid* parameter. This is required for transaction attach security checking.
- The profiles for all the resources accessed by the CICS server application program—files, queues (transient data and temporary storage), programs, and so on. This is required for resource security checking.
- The CICS command profiles for the SPI commands issued by the CICS server application program—INQUIRE, SET, DISCARD and so on. This is required for command security checking.

User security

The target CICS server region performs user security checking against the userid passed on a DPL CALL request. User security checking is performed only when connections specify ATTACHCSEC(IDENTIFY).

User security is performed in addition to any link security.

For user security, in addition to any authorizations you make for link security, you must also authorize the userid specified on the DPL_Request call.

Note that there is no provision for specifying a userid on the EXEC CICS LINK command. In this case, the external CICS interface passes the batch region's userid. User security checking is therefore performed against the batch region's userid if the connection definition specifies ATTACHSEC(IDENTIFY).

Note: If your connection resource definitions for the external CICS interface specify ATTACHSEC(IDENTIFY), your server programs will fail with an ATCY abend if you run them in an environment that does not have an external security manager (ESM) installed and active.

If you want to run external CICS interface server programs without any security active, you must specify ATTACHSEC(LOCAL).

Surrogate user checking

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

EXCI client jobs are subject to surrogate user checking if SURROGCHK=YES (the default) is specified in the EXCI options table, DFHXCOPT. If you specify SURROGCHK=YES (or allow it to default) authorize the batch region's userid as a surrogate of the userid specified on all DPL_Request calls.

If surrogate user checking is enabled (SURROGCHK=YES), but no userids specified on the DPL call, no surrogate user check is performed, because the userid on the DPL call defaults to the batch region's userid. For this bypass of surrogate user checking to be successful, ensure that you have correctly omitted the userid on the DPL call. See "Example of EXCI CALLs with null parameters" on page 140 for information about the correct way to specify a null pointer when omitting an EXCI call parameter.

If you don't want surrogate user security checking, specify SURROGCHK=NO in the DFHXCOPT options table (note that SURROGCHK=YES is the default).

Surrogate user checking is useful when the batch region's userid is the same as the CICS server region userid, in which case the link security check (see "Link security" on page 170) is bypassed. In this case, a surrogate user check is recommended, because the USERID specified on the DPL call is not an authenticated userid (no password is passed).

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

Chapter 16. Problem determination

This chapter contains Diagnosis, Modification or Tuning information.

This chapter describes some of the aids to problem determination provided by the external CICS interface. It covers:
- Trace
- System dumps
- VSE/ESA abends
- The EXCI service trap, DFHXCTRA
- EXCI trace entry points

Details of the external CICS interface messages and abend codes are given in “Chapter 18. Messages and Codes” on page 199.

Trace

The external CICS interface writes trace data to an internal trace table. The internal trace table resides in the partition GETVIS in 31–bit storage. Trace data is formatted and included in any dumps produced by the external CICS interface.

Trace entries are issued by the external CICS interface destined for the internal trace table. They are listed in “EXCI trace entry points” on page 177.

Note: The external CICS interface maintains a separate trace table for each user VSE task in an external CICS interface application program.

The external CICS interface does not support any form of auxiliary trace.

To format external CICS interface trace entries, you use the same FID and ID as for CICS (that is, FID=X'EF', and ID=X'F6C').

System dumps

The external CICS interface produces VSE/ESA SDUMPs for error conditions. These dumps contain all the external CICS interface control blocks, as well as trace entries.

Formatting system dumps

You can use the CICS INFOANA exit, DFHPD410, to format the system dumps. The following keywords are available for use when formatting an external CICS interface dump using DFHPD410:

KE
Formats PSW and registers, and all external CICS interface control blocks.

LD
Formats a load map of where the external CICS interface modules are loaded in the address space, and gives their PTF level.

MRO
Formats the MRO control blocks for the external CICS interface address space, including common control blocks that reside in the VSE/ESA common service area (CSA). This option also formats some MRO blocks that reside in the CICS address space for pipes connected to CICS.

TR

Formats the external CICS interface trace table. You can format the trace table in abbreviated and full forms (TR=1 gives you the abbreviated trace).

SU

Produces a dump summary.

Multiple VSE subtasks

If the external CICS interface takes a system dump when there is more than one VSE task in use, it dumps only the control blocks and trace table for the VSE task that requested the dump.

If you take a dump of the external CICS address space using a console command, the CICS verb exit routine, DFHPD410 formats the control blocks and trace tables for every VSE task using EXCI that it finds in the dump.

Abends from related CICS programs

Certain abends can occur in other CICS programs during the implementation of an external CICS interface client program. These include:

- The CICS translator
- The system dump formatter, DFHPD410
- The resource definition offline utility program, DFHCSDUP.

The following VSE 04xx abends can occur when you are running an external CICS interface job:

0401

Explanation: An external CICS interface (EXCI) request was issued using the CALL API or the EXEC API, and the EXCI stub DFHXCSTB link-edited with the application detected that it was running in AMODE 24. The external CICS interface only supports calls made in AMODE 31.

System Action: The application terminates abnormally.

User Response: Change the application so that EXCI calls are made in AMODE 31, or relink-edit the application AMODE 31.

Module: DFHXCSTB

0402

Explanation: The external CICS interface module DFHXCPRH issued an VSE/ESA ESTAEX macro to establish a recovery environment, but a nonzero return code was returned from VSE/ESA.

System Action: The application terminates abnormally with a dump.

User Response: Examine the dump and any associated VSE/ESA messages produced to determine why the VSE/ESA ESTAEX request failed.

If the error occurred while processing an INITIALIZE_USER request on behalf of the application, an attempt to format the dump using the CICS INFOANA dump formatter does not produce any formatted output. This is because the error occurred too early in EXCI initialization for there to be any control blocks.

Module: DFHXCPRH

0403

Explanation: The external CICS interface module DFHXCPRH issued an VSE/ESA GETMAIN request to obtain storage for its XCGLOBAL block, but a nonzero return code was returned from VSE/ESA.

System Action: Module DFHXCPRH issues a VSE/ESA abend with abend code 0403 which invokes its ESTAEX routine to clear up its environment. A dump is taken before returning control to the application. An application using the EXCI CALL API receives RESPONSE(SYSTEM_ERROR) REASON(XCGLOBAL_GETMAIN_ERROR) in its return area. The subreason1 field of the return area contains the R15 return code from VSE/ESA indicating why the GETMAIN failed. An application using the EXCI EXEC API receives RESP(LINKERR) RESP2(602).

User Response: Use the VSE/ESA R15 return code obtained from the application or from the dump to determine why the VSE/ESA GETMAIN request failed. If the reason is insufficient storage, increase the region size of the batch application.

An attempt to format the dump produced with the CICS IPCS dump formatter does not produce any formatted output for the job because the error occurred too early in EXCI initialization for there to be any control blocks.

Module: DFHXCPRH

0404

Explanation: The external CICS interface module DFHXCPRH needed to take an VSE/ESA SDUMP for an earlier reported problem. However the error has occurred too early in EXCI initialization for EXCI dump services to be available.

System Action: Module DFHXCPRH issues a VSE/ESA abend with abend code 0404 which invokes its ESTAEX routine from which a SYSDUMPX is taken to capture the earlier reported problem.

User Response: Examine the dump to determine the cause of the earlier reported problem.

An attempt to format the SYSMDUMP produced with the CICS IPCS dump formatter does not produce any formatted output for the job because the error occurred too early in EXCI initialization for there to be any control blocks.

Module: DFHXCPRH

0406

Explanation: The external CICS interface module DFHXCPRH called the CICS SVC to initialize the EXCI environment. The CICS SVC call failed.

System Action: Module DFHXCPRH issues a VSE/ESA abend with abend code 0406 which invokes its ESTAEX routine to clear up its environment. A system dump is taken before returning control to the application. An application using the EXCI CALL API receives RESPONSE(SYSTEM_ERROR) REASON(CICS_SVC_CALL_FAILURE) in its return area. The subreason1 field of the return area contains the R15 return code from the CICS SVC indicating why it failed. An application using the EXCI EXEC API receives RESP(LINKERR) RESP2(607).

User Response: Use the VSE/ESA R15 return code obtained from the application or from the dump to determine why the CICS SVC call failed.

An attempt to format the system dump produced with the CICS INFOANA dump formatter does not produce any formatted output for the job because the error occurred too early in EXCI initialization for there to be any control blocks.

Module: DFHXCPRH

0407

Explanation: The external CICS interface module DFHXCPRH issued a call to the CICS SVC to check whether the SVC in use is at the correct level to be used with the external CICS interface. The check failed indicating that the CICS SVC is not at the correct level.

System Action: Message DFHEX0100 is output, and module DFHXCPRH issues a VSE/ESA abend with abend code 0407 which invokes its ESTAEX routine to clear up its environment. A system dump is taken before returning control to the application. An application using the EXCI CALL API receives RESPONSE(SYSTEM_ERROR) REASON(INCORRECT_SVC_LEVEL) in its return area. An application using the EXCI EXEC API receives RESP(LINKERR) RESP2(627).

User Response: See the explanation of message DFHEX0100 for guidance.

An attempt to format the system dump produced with the CICS INFOANA dump formatter does not produce any formatted output for the job because the error occurred too early in EXCI initialization for there to be any control blocks.

Module: DFHXCPRH

0408

Explanation: The external CICS interface module DFHXCPRH issued a VSE/ESA GETMAIN request for its working storage but a nonzero return code was returned from VSE/ESA.

System Action: Module DFHXCPRH issues a VSE/ESA abend with abend code 0408 which invokes its ESTAEX routine to clear up its environment. A system dump is taken before returning control to the application. An application using the EXCI CALL API receives RESPONSE(SYSTEM_ERROR) REASON(WS_GETMAIN_ERROR) in its return area. The subreason1 field of the return area contains the R15 return code from VSE/ESA indicating why the GETMAIN failed. An application using the EXCI EXEC API receives RESP(LINKERR) RESP2(601).

User Response: Use the VSE/ESA R15 return code obtained from the application or from the dump to determine why the VSE/ESA GETMAIN request failed. If the reason is insufficient storage, increase the region size of the batch application.

An attempt to format the system dump produced with the CICS INFOANA dump formatter does not produce any formatted output for the job because the error occurred too early in EXCI initialization for there to be any control blocks.

Module: DFHXCPRH

0410

Explanation: The external CICS interface module DFHXCPRH issued a VSE/ESA GETMAIN request for an XCUSER block but a nonzero return code was returned from VSE/ESA.

System Action: Module DFHXCPRH issues a VSE/ESA abend with abend code 0410 which invokes its ESTAEX routine to clear up its environment. A system dump is taken before returning control to the application. An application using the EXCI CALL API receives RESPONSE(SYSTEM_ERROR) REASON(XCUSER_GETMAIN_ERROR) in its return area. The subreason1 field of the return area contains the R15 return code from VSE/ESA indicating why the GETMAIN failed. An application using the EXCI EXEC API receives RESP(LINKERR) RESP2(603).

User Response: Use the VSE/ESA R15 return code obtained from the application or from the dump to determine why the VSE/ESA GETMAIN request failed. If the reason is insufficient storage, increase the region size of the batch application.

Module: DFHXCPRH

0412

Explanation: The external CICS interface dump module DFHXCEIP was processing an EXCI EXEC API request and detected that the EXEC parameter list passed to it contained a function that is not supported by the external CICS interface.

System Action: The application is abnormally terminated with a dump.

User Response: This error indicates that the parameter list being passed to the EXCI has not been generated by the CICS translator. The translator should always be used. Correct the application to specify the correct EXCI EXEC API command.

An attempt to format the system dump produced with the CICS INFOANA dump formatter may not produce any formatted output for the job if this was the first EXCI request for this TCB.

Module: DFHXCEIP

0413

Explanation: The external CICS interface dump module DFHXCEIP was processing an EXCI EXEC API request and detected that the EXEC parameter list passed to it did not require the mandatory RETCODE parameter in which return codes are returned to the application.

An attempt to format the system dump produced with the CICS INFOANA dump formatter may not produce any formatted output for the job if this was the first EXCI request for this VSE task.

System Action: The application is abnormally terminated with a dump.

User Response: This error indicates that the parameter list being passed to the EXCI has not been generated by the CICS translator. The translator should always be used. Correct the application to specify RETCODE.

Module: DFHXCEIP

0414

Explanation: The external CICS interface module DFHXCEIP issued an VSE/ESA ESTAEX macro to establish a recovery environment but a nonzero return code was returned from VSE/ESA.

System Action: The application terminates abnormally with a dump.

User Response: Examine the dump and any associated VSE/ESA messages to determine why the VSE/ESA ESTAEX request failed.

An attempt to format the system dump produced with the CICS INFOANA dump formatter may not produce any formatted output for the job if this was the first EXCI request for this VSE task.

Module: DFHXCEIP

0415

Explanation: The external CICS interface module DFHXCEIP detected an error early in EXCI initialization before EXCI dump services were available. DFHXCEIP issues abend 0415 so that its ESTAEX routine is invoked from where a system dump is taken instead to capture the error.

System Action: The application terminates abnormally with a dump.

User Response: Examine the system dump to determine the cause of the earlier reported error.

An attempt to format the system dump produced with the CICS INFOANA dump formatter does not produce any formatted output for the job because the error occurred too early in EXCI initialization for there to be any control blocks.

Module: DFHXCEIP

The EXCI service trap, DFHXCTRA

A user-replaceable program, DFHXCTRA, is available for use under the guidance of IBM service personnel. It is the equivalent of DFHTRAP used in CICS. It is invoked every time the external CICS interface writes a trace entry.

DFHXCTRA can perform one or all of the following actions:

1. Request the external CICS interface to write a trace entry on its behalf
2. Instruct the external CICS interface to take an SDUMP
3. Instruct the external CICS interface to disable DFHXCTRA

The CICS-supplied sample version of DFHXCTRA performs all three of the above functions if it detects a trace entry that indicates that a FREEMAIN error occurred while trying to free an EXCI pipe control block.

The source for DFHXCTRA is supplied in PRD1.BASE. The parameter list passed to DFHXCTRA is defined in the copybook DFHXCTRD, which is supplied in PRD1.BASE.DFHXCTRD also defines all the external CICS interface trace points for use by DFHXCTRA.

EXCI trace entry points

Reviewer

Additional trace points.

surrogate user checking - EX2009 and EX200A

Table 19. External CICS interface trace entries

Point ID	Module	Lvl	Type	Data
EX 0001	DFHXCPRH	Exc	PIPE_ALREADY_OPEN	1. Caller’s parameter list 2. Call type 3. Caller’s user name 4. Return codes and message pointer 5. Pipe token
EX 0002	DFHXCPRH	Exc	PIPE_ALREADY_CLOSED	1. Caller’s parameter list 2. Call type 3. Caller’s user name 4. Return codes and message pointer 5. Pipe token
EX 0003	DFHXCPRH	Exc	VERIFY_BLOCK_FM_ERROR	1. Caller’s parameter list 2. Call type 3. Caller’s user name 4. Return codes and message pointer
EX 0005	DFHXCPRH	Exc	XCPIP_ FM_ERR	1. Caller’s parameter list 2. Call type 3. Caller’s user name 4. Return codes and message pointer 5. Pipe token

Table 19. External CICS interface trace entries (continued)

Point ID	Module	Lvl	Type	Data
EX 0006	DFHXCPRH	Exc	IRP_IOAREA_FM_ERR	1. Caller’s parameter list 2. Call type 3. Caller’s user name 4. Return codes and message pointer
EX 0201	DFHXCPRH	Exc	NO_CICS_IRC_STARTED	1. Caller’s parameter list 2. Call type 3. Caller’s user name 4. Return codes and message pointer
EX 0202	DFHXCPRH	Exc	NO_PIPE	1. Caller’s parameter list 2. Call type 3. Caller’s user name 4. Return codes and message pointer 5. Pipe token 6. Target CICS applid
EX 0203	DFHXCPRH	Exc	NO_CICS_ON_OPEN	1. Caller’s parameter list 2. Call type 3. Caller’s user name 4. Return codes and message pointer 5. Pipe token 6. Target CICS applid
EX 0204	DFHXCPRH	Exc	NO_CICS_ON_DPL_1	1. Caller’s parameter list 2. Call type 3. Caller’s user name 4. Return codes and message pointer 5. Pipe token 6. Target CICS applid
EX 0205	DFHXCPRH	Exc	NO_CICS_ON_DPL_2	1. Caller’s parameter list 2. Call type 3. Caller’s user name 4. Return codes and message pointer 5. Pipe token 6. Target CICS applid
EX 0206	DFHXCPRH	Exc	NO_CICS_ON_DPL_3	1. Caller’s parameter list 2. Call type 3. Caller’s user name 4. Return codes and message pointer 5. Pipe token 6. Target CICS applid
EX 0403	DFHXCPRH	Exc	INVALID_APPL_NAME	1. Caller’s parameter list 2. Call type 3. Caller’s user name 4. Return codes and message pointer

Table 19. External CICS interface trace entries (continued)

Point ID	Module	Lvl	Type	Data
EX 0405	DFHXCPRH	Exc	PIPE_NOT_CLOSED	1. Caller's parameter list 2. Call type 3. Caller's user name 4. Return codes and message pointer 5. Pipe token
EX 0406	DFHXCPRH	Exc	PIPE_NOT_OPEN	1. Caller's parameter list 2. Call type 3. Caller's user name 4. Return codes and message pointer 5. Pipe token
EX 0407	DFHXCPRH	Exc	INVALID_USERID	1. Caller's parameter list 2. Call type 3. Caller's user name
EX 0408	DFHXCPRH	Exc	INVALID_UOWID	1. Caller's parameter list 2. Call type 3. Caller's user name 4. UOWID
EX 0409	DFHXCPRH	Exc	INVALID_TRANSID	1. Caller's parameter list 2. Call type 3. Caller's user name
EX 0414	DFHXCPRH	Exc	ABORT_RECEIVED	1. Caller's parameter list 2. Call type 3. Caller's user name 4. Target CICS applid 5. Message to be returned
EX 0415	DFHXCPRH	Exc	INVALID_CONNECTION	1. Caller's parameter list 2. Call type 3. Caller's user name 4. Connection name 5. Target CICS applid
EX 0416	DFHXCPRH	Exc	INVALID_CICS_RELEASE	1. Caller's parameter list 2. Call type 3. Caller's user name 4. Target CICS applid
EX 0417	DFHXCPRH	Exc	PIPE_MUST_CLOSE	1. Caller's parameter list 2. Call type 3. Caller's user name 4. Pipe token
EX 0418	DFHXCPRH	Exc	INVALID_PIPE_TOKEN	1. Caller's parameter list 2. Call type 3. Caller's user name 4. Pipe token
EX 0422	DFHXCPRH	Exc	SERVER_ABENDED	1. Caller's parameter list 2. Call type 3. Caller's user name 4. Return codes and message pointer 5. DPL return area

Table 19. External CICS interface trace entries (continued)

Point ID	Module	Lvl	Type	Data
EX 0603	DFHXCPRH	Exc	XCUSER_GM_ERROR	1. Caller's parameter list 2. Call type 3. Caller's user name 4. Return codes and message pointer
EX 0604	DFHXCPRH	Exc	XCPIPE_GM_ERROR	1. Caller's parameter list 2. Call type 3. Caller's user name 4. Return codes and message pointer
EX 0607	DFHXCPRH	Exc	SVC_CALL_FAILED	1. Caller's parameter list 2. Call type 3. Caller's user name 4. Return codes and message pointer
EX 0608	DFHXCPRH	Exc	IRP_LOGON_FAILURE	1. Caller's parameter list 2. Call type 3. Caller's user name 4. Return codes and message pointer 5. Target CICS applid 6. Logon name
EX 0609	DFHXCPRH	Exc	IRP_CONNECT_FAIL	1. Caller's parameter list 2. Call type 3. Caller's user name 4. Return codes and message pointer 5. Pipe token 6. Target CICS applid
EX 0610	DFHXCPRH	Exc	IRP_DISC_FAIL	1. Caller's parameter list 2. Call type 3. Caller's user name 4. Return codes and message pointer 5. Target CICS applid 6. Pipe token
EX 0611	DFHXCPRH	Exc	IRP_LOGOFF_FAILED	1. Caller's parameter list 2. Call type 3. Caller's user name 4. Return codes and message pointer 5. Target CICS applid 6. Pipe token
EX 0612	DFHXCPRH	Exc	TRANSFORM_1_ERROR	1. Caller's parameter list 2. Call type 3. Caller's user name 4. Return codes and message pointer
EX 0613	DFHXCPRH	Exc	TRANSFORM_4_ERROR	1. Caller's parameter list 2. Call type 3. Caller's user name 4. Return codes and message pointer

Table 19. External CICS interface trace entries (continued)

Point ID	Module	Lvl	Type	Data
EX 0614	DFHXCPRH	Exc	IRP_NULL_DATA	1. Caller's parameter list 2. Call type 3. Caller's user name 4. Return codes and message pointer 5. Target CICS applid
EX 0615	DFHXCPRH	Exc	IRP_NEG_RESPONSE	1. Caller's parameter list 2. Call type 3. Caller's user name 4. Return codes and message pointer 5. Target CICS applid
EX 0616	DFHXCPRH	Exc	IRP_SWITCH_PULL_ERR	1. Caller's parameter list 2. Call type 3. Caller's user name 4. Return codes and message pointer 5. Target CICS applid 6. Pipe token
EX 0617	DFHXCPRH	Exc	IRP_IOAREA_GM_ERR	1. Caller's parameter list 2. Call type 3. Caller's user name 4. Return codes and message pointer
EX 0619	DFHXCPRH	Exc	IRP_BAD_IOAREA	1. Caller's parameter list 2. Call type 3. Caller's user name 4. IOAREA address
EX 0620	DFHXCPRH	Exc	IRP_PROTOCOL_ERR	1. Caller's parameter list 2. Call type 3. Caller's user name 4. Target CICS applid 5. Pipe token
EX 0621	DFHXCPRH	Exc	PIPE_RECOVERY_FAILURE	1. Caller's parameter list 2. Call type 3. Caller's user name 4. Target CICS applid 5. Pipe token
EX 0622	DFHXCPRH	Exc	ESTAEX_SETUP_FAIL	1. Caller's parameter list 2. Call type 3. Caller's user name 4. Return codes and message pointer
EX 0623	DFHXCPRH	Exc	ESTAEX_INVOKED	1. Caller's parameter list 2. Call type 3. Caller's user name 4. Return codes and message pointer 5. OS/390 abend code (see ″OS/390 API Abend Codes″ in section ″VSE/Advanced Functions & SVC Errors″ of VSE/ESA messages and Codes — Volume 1)

Table 19. External CICS interface trace entries (continued)

Point ID	Module	Lvl	Type	Data
EX 0624	DFHXCPRH	Exc	TIMEDOUT	1. Caller's parameter list 2. Call type 3. Caller's user name 4. Server program name 5. Target CICS applid
EX 0625	DFHXCPRH	Exc	STIMER_SETUP_FAIL	1. Caller's parameter list 2. Call type 3. Caller's user name 4. Return codes and message pointer
EX 0626	DFHXCPRH	Exc	STIMER_CANCEL_FAIL	1. Caller's parameter list 2. Call type 3. Caller's user name 4. Return codes and message pointer
EX 0627	DFHXCPRH	Exc	INCORRECT_SVC_LEVEL	1. Caller's parameter list 2. Call type 3. Caller's user name 4. SVC instruction
EX 0800	DFHXCPRH	Exc	RESP shows LENGERR	1. Caller's parameter list 2. Call type 3. Caller's user name 4. Return codes and message pointer 5. COMMAREA length 6. Data length
EX 0801	DFHXCPRH	Exc	RESP shows INVREQ	1. Caller's parameter list 2. Call type 3. Caller's user name 4. Return codes and message pointer 5. DPL options specified
EX 0802	DFHXCPRH	Exc	RESP shows PGMIDERR	1. Caller's parameter list 2. Call type 3. Caller's user name 4. Program name 5. Target CICS applid
EX 0803	DFHXCPRH	Exc	RESP shows ROLLEDBACK	1. Caller's parameter list 2. Call type 3. Caller's user name 4. Program name 5. Target CICS applid
EX 0804	DFHXCPRH	Exc	RESP shows NOTAUTH	1. Caller's parameter list 2. Call type 3. Caller's user name 4. Program name 5. Target CICS applid
EX 0805	DFHXCPRH	Exc	RESP shows SYSIDERR	1. Caller's parameter list 2. Call type 3. Caller's user name 4. Program name 5. Target CICS applid 6. DPL_Retarea

Table 19. External CICS interface trace entries (continued)

Point ID	Module	Lvl	Type	Data
EX 0806	DFHXCPRH	Exc	RESP shows TERMERR	1. Caller's parameter list 2. Call type 3. Caller's user name 4. Program name 5. Target CICS applid
EX 0904	DFHXCTRP	Exc	Overlength trace data field	1. XCTRP parameter list
EX 0905	DFHXCTRA	Exc	DFHXCTRA trace entry	1. User specified data
EX 1000	DFHXCPRH	EX 1	Entry	For INIT_USER commands: 1. Caller's parameter list 2. Call type 3. Caller's user name 4. Caller's register 14 For Allocate_Pipe requests: 1. Caller's parameter list 2. Call type 3. Caller's user name 4. CICS name 5. Allocate options 6. Caller's register 14 For Open, Close, and Deallocate requests: 1. Caller's parameter list 2. Call type 3. Caller's user name 4. CICS name 5. Pipe token 6. Caller's register 14 For DPL requests: 1. Caller's parameter list 2. Call type 3. Caller's user name 4. CICS name 5. Pipe token 6. Program name 7. Caller's register 14

Table 19. External CICS interface trace entries (continued)

Point ID	Module	Lvl	Type	Data
EX 1001	DFHXCPRH	EX 1	Exit	For INIT_USER, OPEN, CLOSE, and DEALLOCATE requests: 1. Caller's parameter list 2. Call type 3. Caller's user name 4. Return codes and message pointer 5. Caller's register14 For Allocate requests: 1. Caller's parameter list 2. Call type 3. Caller's user name 4. Return codes and message pointer 5. Pipe token 6. Caller's register 14 For DPL requests: 1. Caller's parameter list 2. Call type 3. Caller's user name 4. Return codes and message pointer 5. DPL return area 6. Caller's register 14
EX 1010	DFHXCEIP	EX 1	Entry	1. Program name 2. Target CICS applid 3. Transaction ID 4. Caller's register 14 5. Up to the first 100 bytes of COMMAREA (if passed) 6. COMMAREA length, if COMMAREA passed 7. Data length, if COMMAREA passed
EX 1011	DFHXCEIP	EX 1	Exit	1. EXEC retarea 2. Program name 3. Target CICS applid 4. Transaction ID 5. Caller's register 14 6. Up to the first 100 bytes of COMMAREA (if passed) 7. COMMAREA length, if COMMAREA passed
EX 2000	DFHXCPRH	EX 2	IRP_LOGON	1. Caller's parameter list 2. Call type 3. Caller's user name 4. Target CICS applid 5. IRP userid 6. SLCB address 7. Connection name

Table 19. External CICS interface trace entries (continued)

Point ID	Module	Lvl	Type	Data
EX 2001	DFHXCPRH	EX 2	IRP_CONN	1. Caller’s parameter list 2. Call type 3. Caller’s user name 4. Target CICS applid 5. IRP userid 6. IRP thread ID 7. SCCB address
EX 2002	DFHXCPRH	EX 2	IRP_DISC	1. Caller’s parameter list 2. Call type 3. Caller’s user name 4. Target CICS applid 5. Pipe token
EX 2003	DFHXCPRH	EX 2	IRP_LOGOFF	1. Caller’s parameter list 2. Call type 3. Caller’s user name 4. Pipe token 5. IRP userid
EX 2004	DFHXCPRH	EX 2	IRP_SWITCH	1. Caller’s parameter list 2. Call type 3. Caller’s user name 4. Target CICS applid 5. IRP userid 6. IRP threadid
EX 2005	DFHXCPRH	EX 2	IRP_SWITCH_DATA	1. User’s appl name 2. Pipe token 3. Request header 4. Bind data 5. UOWID/USERID FMH 6. Transformed DPL request to CICS (up to 1000 bytes) 7. Final 1000 bytes of transformed DPL request
EX 2006	DFHXCPRH	EX 2	IRP_DATA	1. Caller’s parameter list 2. Call type 3. Caller’s user name 4. Target CICS applid 5. Length of data returned 6. Data (first 1000 bytes) 7. Data (final 1000 bytes)
EX 2007	DFHXCPRH	EX 2	PRE_URM	1. Caller’s parameter list 2. Call type 3. Caller’s user name 4. Parameters passed to DFHXCURM 5. URMINV, reason for calling URM 6. URMCICS, target CICS applid 7. URMANCH, URM anchor point address

Table 19. External CICS interface trace entries (continued)

Point ID	Module	Lvl	Type	Data
EX 2008	DFHXCPRH	EX 2	POST_URM	1. Caller's parameter list 2. Call type 3. Caller's user name 4. Parameters passed to DFHXCURM 5. URMINV, reason for calling URM 6. URMCICS, target CICS applid 7. URMANCH, URM anchor point address
EX 3000	DFHXCEIP	Exc	ESTAEX_SETUP_ERROR	1. Return area (20 bytes) 2. VSE/ESA return code
EX 3001	DFHXCEIP	Exc	ESTAEX_INVOKED	1. Return area (20 bytes)
EX 3002	DFHXCEIP	Exc	INV_CTYPE_ON_INIT	1. Return area (20 bytes) 2. Call type
EX 3003	DFHXCEIP	Exc	INV_VNUM_ON_INIT	1. Return area (20 bytes) 2. Version number
EX 3004	DFHXCEIP	Exc	INV_APPL_NAME_ON_INIT	1. Return area (20 bytes) 2. User name
EX 3005	DFHXCEIP	Exc	INV_CTYPE_ON_ALLOC	1. Return area (20 bytes) 2. Call type
EX 3006	DFHXCEIP	Exc	INV_VNUM_ON_ALLOC	1. Return area (20 bytes) 2. Version number
EX 3007	DFHXCEIP	Exc	INV_UTOKEN_ON_ALLOC	1. Return area (20 bytes) 2. User token
EX 3008	DFHXCEIP	Exc	INV_CTYPE_ON_OPEN	1. Return area (20 bytes) 2. Call type
EX 3009	DFHXCEIP	Exc	INV_VNUM_ON_OPEN	1. Return area (20 bytes) 2. Version number
EX 3010	DFHXCEIP	Exc	INV_UTOKEN_ON_OPEN	1. Return area (20 bytes) 2. User token
EX 3011	DFHXCEIP	Exc	INV_PTOKEN_ON_OPEN	1. Return area (20 bytes) 2. Pipe token
EX 3012	DFHXCEIP	Exc	INV_CTYPE_ON_DPL	1. Return area (20 bytes) 2. Call type
EX 3013	DFHXCEIP	Exc	INV_VNUM_ON_DPL	1. Return area (20 bytes) 2. Version number
EX 3014	DFHXCEIP	Exc	INV_UTOKEN_ON_DPL	1. Return area (20 bytes) 2. User token
EX 3015	DFHXCEIP	Exc	INV_PTOKEN_ON_DPL	1. Return area (20 bytes) 2. Pipe token
EX 3017	DFHXCEIP	Exc	INV_USERID_ON_DPL	1. Return area (20 bytes) 2. Userid
EX 3018	DFHXCEIP	Exc	PIPE_NOT_OPEN_ON_DPL	1. Return area (20 bytes) 2. Pipe token
EX 3019	DFHXCEIP	Exc	PIPE_MUST_CLOSE_ON_DPL	1. Return area (20 bytes) 2. Pipe token

Table 19. External CICS interface trace entries (continued)

Point ID	Module	Lvl	Type	Data
EX 3020	DFHXCEIP	Exc	INV_CTYPE_ON_CLOSE	1. Return area (20 bytes) 2. Call type
EX 3021	DFHXCEIP	Exc	INV_VNUM_ON_CLOSE	1. Return area (20 bytes) 2. Version number
EX 3022	DFHXCEIP	Exc	INV_UTOKEN_ON_CLOSE	1. Return area (20 bytes) 2. User token
EX 3023	DFHXCEIP	Exc	INV_PTOKEN_ON_CLOSE	1. Return area (20 bytes) 2. Pipe token
EX 3024	DFHXCEIP	Exc	INV_CTYPE_ON_DEALL	1. Return area (20 bytes) 2. Call type
EX 3025	DFHXCEIP	Exc	INV_VNUM_ON_DEALL	1. Return area (20 bytes) 2. Version number
EX 3026	DFHXCEIP	Exc	INV_UTOKEN_ON_DEALL	1. Return area (20 bytes) 2. User token
EX 3027	DFHXCEIP	Exc	INV_PTOKEN_ON_DEALL	1. Return area (20 bytes) 2. Pipe token
EX 3028	DFHXCEIP	Exc	PIPE_NOT_CLOSED_ON_DEALL	1. Return area (20 bytes) 2. Pipe token
EX 3029	DFHXCEIP	Exc	XCEIP_RETRYING	1. Return area (20 bytes)

Chapter 17. Response and reason codes returned on EXCI calls

This chapter gives details of the reason codes for the responses returned on the EXCI call interface.

Note: All numeric response and reason code values shown are in decimal.

See also ″OS/390 API Return Codes″ in section ″VSE/Advanced Functions & SVC Errors″ of VSE/ESA Messages and Codes — Volume 1 for OS/390 return codes.

Reason code for response: OK

0 NORMAL

Explanation: Call completed normally.

Reason codes for response: WARNING

1 PIPE_ALREADY_OPEN

Explanation: An Open_Pipe request has been issued for a pipe that is already open.

System Action: None. The pipe remains open.

User Response: If this response is unexpected, investigate whether an incorrect pipe token has been used on the Open_Pipe call.

2 PIPE_ALREADY_CLOSED

Explanation: A Close_Pipe request has been issued for a pipe that is already closed.

System Action: The external CICS interface ignores the request and the pipe remains closed.

User Response: If the response is unexpected, check that the Close_Pipe call is specifying the correct pipe token.

4 WS_FREEMAIN_ERROR

Explanation: An attempt to FREEMAIN working storage has resulted in a OS/390 FREEMAIN error.

System Action: The return code from the FREEMAIN is returned in the EXCI subreason field-1. The Initialize_User request continues unaffected.

User Response: If the problem persists, take a dump of the batch region and use the dump, together with the return code from the OS/390 FREEMAIN to determine why the FREEMAIN is failing.

5 XCPIPE_FREEMAIN_ERROR

Explanation: An attempt to FREEMAIN pipe storage has resulted in an OS/390 FREEMAIN error.

System Action: The return code from the FREEMAIN is returned in the EXCI subreason field-1. However, the external CICS interface continues processing the Deallocate_Pipe request. If the request fails with another error, this reason code is overwritten.

User Response: If the problem persists, take a dump of the client application program address space, and use the dump, with the return code from the OS/390 FREEMAIN to determine why the FREEMAIN is failing.

6 IRP_IOAREA_FM_FAILURE

Explanation: An attempt to FREEMAIN an MRO I/O area has resulted in an OS/390 FREEMAIN error.

System Action: The return code from the FREEMAIN is returned in the EXCI subreason field-1, but the DPL request continued to completion. Reason IRP_IOAREA_FM_FAILURE is returned to your application only if the DPL request completes, otherwise it is overwritten by subsequent response and reason codes.

User Response: If the problem persists, take a dump of the batch region and use it with the return code from the OS/390 FREEMAIN to determine why the FREEMAIN is failing.

7 SERVER_TERMINATED

Explanation: The CICS session, on which the server program has been executing, has been freed by CICS.

System Action: The CICS application server program has been detached at some point in its processing, and control is returned to the external CICS interface, which writes a trace entry for this error.

User Response: The most likely reason for this error is that the server program has caused CICS to terminate, perhaps by an EXEC CICS PERFORM SHUTDOWN command. During shutdown, CICS frees EXCI sessions so that shutdown can complete.

Reason codes for response: RETRYABLE

202 NO_PIPE

Explanation: An attempt has been made to open a pipe, but the target CICS system associated with the pipe has no free receive sessions.

System Action: The Open_pipe call fails, and the external CICS interface invokes the user-replaceable module, DFHXCURM.

User Response: This situation can occur even if the client application program has allocated (using Allocate_Pipe calls) no more pipes than the number of receive sessions defined on the target connection. This is because CICS can be in the process of cleaning up a pipe from a Close_Pipe request. For this reason, you are recommended to specify a larger RECEIVECOUNT value than is theoretically necessary when defining the SESSIONS resource definition to CICS. The application program can reissue the Open_Pipe request.

203 (on Open_Pipe call) NO_CICS

Explanation: An attempt has been made to open a pipe but the target CICS system is not available, or hasn't yet opened IRC, or the target connection is out of service, or the relevant EXCI connection definition is not installed in the target CICS.

System Action: The open pipe request fails, and the external CICS interface invokes the user-replaceable module, DFHXCURM.

User Response: If subreason field-1 is non-zero (the IRP response code (R15)), subreason field-2 contains the IRP reason code. For an explanation of the IRP return codes, see the interregion control blocks in the *CICS Data Areas* manual. The IRP return codes are in the DFHIRSPS copybook, listed under the heading IRC.

When you have corrected the problem, your client application program can reissue the Open_Pipe call.

Reason codes for response: USER_ERROR

401 INVALID_CALL_TYPE

Explanation: An invalid *call-type* parameter value is specified on this EXCI request.

System Action: The request is rejected.

User Response: Check your EXCI client program and ensure the *call_type* parameter specifies the appropriate value for the EXCI call, as follows.

1	Initialize_User
2	Allocate_Pipe
3	Open_Pipe
4	Close_Pipe
5	Deallocate_Pipe
6	DPL

402 INVALID_VERSION_NUMBER

Explanation: The *version_number* parameter does not specify a value of 1.

System Action: The request is rejected.

User Response: Check the client application program and ensure that all EXCI calls specify the value of 1 for the version number.

403 INVALID_APPL_NAME

Explanation: The *user_name* parameter consists of all blank characters (X'40').

System Action: The call is rejected.

User Response: Change the application program to specify a valid, non-blank user name.

404 INVALID_USER_TOKEN

Explanation: The client application program has issued an EXCI request using a user token that is unknown to the external CICS interface.

System Action: The request is rejected.

User Response: The Initialize_User call returns a 4-byte token that must be used on *all* further requests for the that user. Check the client application program and correct the error to ensure that the correct token is passed.

405 PIPE_NOT_CLOSED

Explanation: A Deallocate_Pipe request has been issued against a pipe that has not yet been closed.

System Action: The external CICS interface ignores the request and the pipe remains open.

User Response: Check the client application program, and ensure that the Deallocate_Pipe request is intended. If so, issue a Close_Pipe request for the pipe before issuing the Deallocate_Pipe request.

406 PIPE_NOT_OPEN

Explanation: A DPL call has been issued on a pipe that is not open.

System Action: The external CICS interface rejects the DPL request.

User Response: Check the client application program, and ensure that an Open_Pipe request is issued before using the pipe on a DPL request. If an Open_Pipe has been issued by the application program, check that it has not been closed inadvertently before all the DPL requests have been made.

407 INVALID_USERID

Explanation: A DPL request has been issued with a USERID parameter that consists of all blanks.

System Action: The DPL request is rejected.

User Response: Check the EXCI client program and ensure that the DPL request passes a valid USERID parameter. If you don't want to specify a userid, code the call parameter list with a null address for *userid*. If you pass a null address, the external CICS interface passes the userid under which the client application program is running (the batch region's userid).

408 INVALID_UOWID

Explanation: A DPL request has been issued with a *uowid* parameter that has invalid length fields.

System Action: The DPL request is rejected.

User Response: Check the client application program and ensure that the DPL request passes a valid *uowid* parameter. If you don't want to specify a unit of work id, code the call parameter list with a null address for *uowid*, in which case the external CICS interface generates a unit of work id for you.

409 INVALID_TRANSID

Explanation: A DPL request has been issued with a *transid* parameter that consists of all blanks.

System Action: The DPL request is rejected.

User Response: Check the client application program and ensure that the *transid* parameter is specified correctly or has not been overwritten in some way. If you don't want to specify your own transid, code the call parameter list with a null address for *transid*, in which case the external CICS interface uses the default CICS mirror transaction, CSMI.

410 DFHMEBM_LOAD_FAILED

Explanation: During Initialize_User processing, the external CICS interface attempted to load the main message module in preparation for issuing external CICS interface messages, and the load of this module failed.

System Action: The Initialize_User call is rejected. The return code from the OS/390 load macro (R15) is returned in the subreason field-1. The external CICS interface handles the error, and returns the abend (R0) that would have occurred in the subreason field-2.

User Response: Using the OS/390 return code, determine why the load failed. The most likely reason is that the message module, DFHMEBM, is not in any library included in the LIBDEF PHASE, SEARCH library concatenation of the batch job. Ensure the PRD1.BASE library is included in the LIBDEF PHASE, SEARCH library concatenation, and restart the client application program.

411 DFHMET4E_LOAD_FAILED

Explanation: The load of message module, DFHMET4E, has failed. During Initialize_User processing, the external CICS interface attempted to load its message table in preparation for issuing messages. The load of this module failed.

System Action: The Initialize_User call is rejected. The return code from the OS/390 load macro (R15) is returned in the subreason field-1. The external CICS interface handles the error, and returns the abend (R0) that would have occurred in the subreason field-2.

User Response: Using the OS/390 reason code, determine why the load failed. The most likely reason is that the message table, DFHMET4E, is not in any library included in the LIBDEF PHASE, SEARCH library concatenation of the batch job. Ensure the PRD1.BASE library is included in the LIBDEF PHASE, SEARCH library concatenation, and restart the client application program.

412 DFHXCURM_LOAD_FAILED

Explanation: During Initialize_User processing, the external CICS interface attempted to load the user-replaceable module, DFHXCURM. The load of this module failed.

System Action: The Initialize_User call is rejected. The return code from the OS/390 load macro (R15) is returned in the subreason field-1. The external CICS interface handles the error, and returns the abend (R0) that would have occurred in the subreason field-2.

User Response: Using the OS/390 reason code, determine why the load failed. The most likely reason is that module DFHXCURM is not in any library included in the LIBDEF PHASE, SEARCH library concatenation of the batch job. Ensure the library containing the module is included in the LIBDEF PHASE, SEARCH library concatenation, and restart the client application program.

413 DFHXCTRA_LOAD_FAILED

Explanation: During Initialize_User processing, the external CICS interface attempted to load the trap module (DFHXCTRA). The load of this module has failed.

System Action: The Initialize_User call is rejected. The return code from the OS/390 load macro (R15) is returned in the subreason field-1. The external CICS interface handles the error, and returns the abend (R0) that would have occurred in the subreason field-2.

User Response: Using the OS/390 reason code, determine why the load failed. The most likely reason is that DFHXCTRA is not in any library included in the LIBDEF PHASE, SEARCH library concatenation of the batch job. Ensure the library containing the module is included in the LIBDEF PHASE, SEARCH library concatenation, and restart the client application program.

414 IRP_ABORT_RECEIVED

Explanation: Whilst processing a DPL request, an error occurred in the CICS server region, resulting in an abort FMH7 flow being returned to the external CICS interface.

System Action: A message is returned to the client application program. This is the message that would have been issued to the terminal if the server program had been initiated from a terminal. A pointer to the message is returned to the client application program in the message pointer field of the EXCI return area. See the description of the EXCI return areas for the exact definition of the message format. The pipe is put into a “must close” state.

User Response: Use the message to determine the cause of the error. A typical example is where the server transaction cannot be attached, either because is disabled, or it has not been defined, or because of a security failure. Correct the problem, close and reopen the pipe, and reissue the DPL request.

415 INVALID_CONNECTION_DEFN

Explanation: A DPL request has been rejected by CICS because the target connection is not defined for use by an external CICS client application program.

System Action: The DPL request is rejected and the pipe is put into a “must close” state.

User Response: The most likely reason for this is that the connection definition in the CICS server region has been defined incorrectly as a CICS-to-CICS MRO connection, instead of an EXCI connection. Ensure that PROTOCOL(EXCI) is specified on the appropriate CONNECTION and SESSIONS resource definitions. You must close and reopen the pipe before reissuing the DPL request.

416 INVALID_CICS_RELEASE

Explanation: A DPL request has been rejected by the target CICS server region because it doesn’t recognize the request.

System Action: The DPL call is rejected and the pipe is put into a “must close” state.

User Response: The most likely reason for this is that the client application program has specified a target CICS server region that is not a CICS Transaction Server for VSE/ESA Release 1 region. CICS regions earlier than this do not recognize EXCI call requests. Correct the problem, close and reopen the pipe and then reissue the DPL request.

417 PIPE_MUST_CLOSE

Explanation: A DPL request has been issued on a pipe that is in a “must close” state.

System Action: The DPL request is rejected.

User Response: Some EXCI errors are serious enough to require that the pipe be closed and reopened in order to restore the pipe to a point where it can be used for further DPL requests. Others, more minor errors, allow further calls without closing and reopening the pipe. A previous error on this pipe has been of the more serious variety and the pipe is now in a “must close” state. Close and reopen the pipe and reissue the DPL request.

418 INVALID_PIPE_TOKEN

Explanation: An Open_Pipe, Close_Pipe, Deallocate_Pipe, or DPL request has been issued, but the pipe token passed on the call is either not a valid pipe, or is not a valid pipe allocated for this user (that is, there is mismatch between the user token and the pipe token).

System Action: The call is rejected.

User Response: Ensure that the pipe token has not been overwritten and is being passed correctly on the call. Also ensure there is no mismatch between the user token and the pipe token.

419 CICS_AFCB_PRESENT

Explanation: An Initialize_User request has been issued on a VSE task that has already been used by CICS. The external CICS interface cannot share a VSE task with CICS, ensuring that a CICS application program cannot issue EXCI requests.

System Action: The Initialize_User request is rejected.

User Response: To use the external CICS interface, you must create a new VSE subtask (or daughter subtask), and issue the EXCI calls under that unique subtask.

420 DFHXCOPT_LOAD_FAILED

Explanation: During Initialize_User processing, the external CICS interface attempted to load its options module, DFHXCOPT. The load of this module failed.

System Action: The Initialize_User call is rejected. The return code from the OS/390 load macro (R15) is returned in the subreason field-1. The external CICS interface handles the error, and returns the abend (R0) that would have occurred in the subreason field-2.

User Response: Using the OS/390 reason code, determine why the load failed. The most likely reason is that DFHXCOPT is not in any library included in the LIBDEF PHASE, SEARCH library concatenation of the batch job. Correct the problem and restart the client application program.

422 SERVER_ABENDED

Explanation: Whilst processing a DPL request, the CICS server application program abended without handling the error.

System Action: The server application program is abended and backout out. The abend code is returned in the abend code field of the EXCI return area.

User Response: Determine why the server program abended and fix the problem.

Reason codes for response: SYSTEM_ERROR

601 WS_GETMAIN_ERROR

Explanation: During Initialize_User processing, a GETMAIN for working storage failed.

System Action: Processing cannot continue without working storage, so the request is terminated. At this point the external CICS interface trace and dump services are not available to provide diagnostic information, therefore EXCI issues an OS/390 abend (U0408) to force a SYSDUMP. The return code from the OS/390 GETMAIN request is returned in the return area.

User Response: Locate the GETMAIN return code in the dump, and use this and the rest of the dump to determine why the GETMAIN failed. Possible reasons are:

- The ALLOC size specified for the partition is too small
- The SIZE parameter is too large, restricting the amount of storage available for the OS/390 GETMAIN.

Correct the ALLOC or SIZE parameters and restart the client application.

602 XCGLOBAL_GETMAIN_ERROR

Explanation: During Initialize_User processing, a GETMAIN failed for a critical control block (XCGLOBAL).

System Action: Processing cannot continue without this control block, and the request is terminated. At this point the external CICS interface trace and dump services are not available to provide diagnostic information, therefore EXCI issues an OS/390 abend (U0403) to force a SYSDUMP. The return code from the OS/390 GETMAIN request is returned in the return area.

User Response: Locate the GETMAIN return code in the dump, and use this and the rest of the dump to determine why the GETMAIN failed. Possible reasons are:

- The ALLOC size specified for the partition is too small
- The SIZE parameter is too large, restricting the amount of storage available for the OS/390 GETMAIN.

Correct the ALLOC or SIZE parameters and restart the client application.

603 XCUSER_GETMAIN_ERROR

Explanation: During Initialize_User processing, a GETMAIN request failed for the user control block (XCUSER).

System Action: Initialize_User processing is terminated. The return code from the GETMAIN is returned in subreason field-1 of the return area. The external CICS interface issues message DFHEX0003 and issues an OS/390 abend (0410) to force a SYSDUMP.

User Response: Use the return code from the GETMAIN, with the dump, to determine why the GETMAIN failed. Possible reasons are:

- The ALLOC size specified for the partition is too small
- The SIZE parameter is too large, restricting the amount of storage available for the OS/390 GETMAIN.

Correct the ALLOC or SIZE parameters and restart the client application.

604 XCPIPE_GETMAIN_ERROR

Explanation: During Allocate_Pipe processing, a GETMAIN request for the pipe control block (XCPIPE) failed.

System Action: Allocate_Pipe processing is terminated. The return code from the GETMAIN is returned in subreason field-1 of the EXCI return area. The external CICS interface issues message DFHEX0003, and takes a system dump.

User Response: Use the return code from the GETMAIN, and the dump, to determine why the GETMAIN failed. Possible reasons are:

- The ALLOC size specified for the partition is too small
- The SIZE parameter is too large, restricting the amount of storage available for the OS/390 GETMAIN.

Correct the ALLOC or SIZE parameters and restart the client application.

605 VERIFY_BLOCK_GM_ERROR

Explanation: During Initialize_User processing, a GETMAIN failed for an EXCI internal control block.

System Action: Initialize_User processing is terminated. The return code from the GETMAIN is returned in the subreason field-1 of the EXCI return area. This error occurs before EXCI dumping services are initialized, Therefore EXCI issues an OS/390 abend (U0409) to force a SYSDUMP The return code from the OS/390 GETMAIN request is returned in the return area.

User Response: Locate the GETMAIN return code in the dump, and use this and the rest of the dump to determine why the GETMAIN failed. Possible reasons are:

- The ALLOC size specified for the partition is too small
- The SIZE parameter is too large, restricting the amount of storage available for the OS/390 GETMAIN.

Correct the ALLOC or SIZE parameters and restart the client application.

607 CICS_SVC_CALL_FAILURE

Explanation: During Initialize_User processing, a call to the currently installed CICS SVC failed.

System Action: The return code from the CICS SVC is returned in the subreason field-1 of the EXCI return area. This error occurs before the external CICS interface dump services are initialized, therefore EXCI issues an OS/390 user abend (0406) to force a SYSDUMP.

User Response: Contact your IBM support center for assistance, with the return code and the dump available.

608 IRC_LOGON_FAILURE

Explanation: During Allocate_Pipe processing, an attempt by the external CICS interface to LOGON to DFHIRP failed.

System Action: The Allocate_Pipe request fails. DFHIRP returns a R15 value to subreason field-1 and a R0 value (the reason code) to subreason field-2. The first two bytes of subreason field-1 are the return code qualifier and the last two bytes are the return code itself.

User Response: For an explanation of the IRP return codes, see the interregion control blocks in the *CICS Data Areas* manual. The IRP return codes are in the DFHIRSPS copybook, listed under the heading IRC. Use the return codes to determine why the logon failed, or contact your IBM support personal with details of the failure.

609 IRC_CONNECT_FAILURE

Explanation: During Open_Pipe processing, an attempt to connect to the target CICS system failed.

System Action: The Open_Pipe request fails. DFHIRP returns a R15 value to subreason field-1 and a R0 value (the reason code) to subreason field-2. The first two bytes of subreason field-1 are the return code qualifier and the last two bytes are the return code itself.

User Response: For an explanation of the IRP return codes, see the interregion control blocks in the *CICS Data Areas* manual. The IRP return codes are in the DFHIRSPS copybook, listed under the heading IRC.

Use the return code to determine why the logon failed, and reissue the open pipe request.

Note: This error is not caused by the target CICS being unavailable, which is returned as a RETRYABLE condition (NO_CICS).

610 IRC_DISCONNECT_FAILURE

Explanation: During Close_Pipe processing, CICS issued a DFHIRP disconnect call to terminate the connection to CICS. This request has failed.

System Action: The call fails and the pipe is left open. DFHIRP returns a R15 value to subreason field-1 and a R0 value (the reason code) to subreason field-2. The first two bytes of subreason field-1 are the return code qualifier and the last two bytes are the return code itself.. The external CICS interface takes a system dump.

Although the disconnect failed, it is possible that the pipe is still connected to CICS. However, all connections are automatically disconnected at the end of the batch program.

User Response: For an explanation of the IRP return codes, see the interregion control blocks in the *CICS*

Data Areas manual. The IRP return codes are in the DFHIRSPS copybook, listed under the heading IRC. Use the return code and the dump to determine the cause of the error.

611 IRC_LOGOFF_FAILURE

Explanation: During Deallocate_Pipe processing, CICS issued a DFHIRP logoff call. This request failed.

System Action: The Deallocate_Pipe call fails and the pipe remains allocated. DFHIRP returns a R15 value to subreason field-1 and a R0 value (the reason code) to subreason field-2. The first two bytes of subreason field-1 are the return code qualifier and the last two bytes are the return code itself. The external CICS interface takes a system dump.

Note: Because it remains allocated, the pipe is available for further calls. Any storage associated with the pipe is not freed. However, this storage is freed at the end of the client application program.

User Response: For an explanation of the IRP return codes, see the interregion control blocks in the *CICS Data Areas* manual. The IRP return codes are in the DFHIRSPS copybook, listed under the heading IRC. Use the return code and the dump to determine the cause of the error.

612 TRANSFORM_1_ERROR

Explanation: During DPL processing, whilst processing the data in preparation for sending to CICS, an internal call to program DFHXFQ resulted in an error.

System Action: The DPL request is terminated.

User Response: The return code from the call is returned in the EXCI subreason field-1, and the external CICS interface takes a system dump.

This is an external CICS interface error. Contact your IBM support center with details of the return code and the dump.

613 TRANSFORM_4_ERROR

Explanation: During DPL processing, whilst processing the data returned by the CICS server region, an internal call to module DFHXFQ resulted in an error.

System Action: The DPL request is terminated. Note that the server application program has executed. The return code from the call to DFHXFQ is returned in the EXCI subreason field-1. This return code corresponds to any EIBRCODE information that was available. The external CICS interface takes a system dump.

User Response: This is an external CICS interface error. Contact your IBM support center with details of the return code and the dump.

614 IRP_NULL_DATA_RECEIVED

Explanation: During DPL processing, a request has been sent to the target CICS and this target CICS has replied without returning any data.

System Action: The DPL processing is terminated and the external CICS interface takes a system dump.

User Response: This is an internal protocol error. Contact your IBM support center with details of the dump.

615 IRP_NEGATIVE_RESPONSE

Explanation: An internal protocol error has occurred whilst trying to communicate with the target CICS region.

System Action: The DPL request fails, the pipe is put into a “must close” state, and the external CICS interface takes a system dump.

User Response: This is an external CICS interface error. Keep the dump and contact your IBM support center.

Note: The pipe is in a “must close” state. Before attempting further calls, the pipe must first be closed and reopened.

616 IRP_SWITCH_PULL_FAILURE

Explanation: An internal protocol error has occurred whilst trying to communicate with the target CICS region.

System Action: The DPL request fails, the pipe is put into a “must close” state, and the external CICS interface takes a system dump. The IRP return code (R15) and reason code if any (R0) are returned in the EXCI subreason field-1 and subreason field-2.

User Response: This is an external CICS interface error. Keep the dump and contact your IBM support center.

Note: The pipe is in a “must close” state, and before attempting further DPL calls, the pipe must first be closed and reopened.

617 IRP_IOAREA_GM_FAILURE

Explanation: During DPL processing, an OS/390 GETMAIN request for an internal control block failed.

System Action: The DPL request is terminated. The return code from the GETMAIN is returned in the EXCI subreason field-1.

Note: This error occurs whilst processing the data returned by CICS, after the server application program has completed execution. This error results in the pipe being put into a "must close" state.

User Response: Use the return code to determine why the GETMAIN failed. Possible reasons are:

- The ALLOC size specified for the partition is too small
- The SIZE parameter is too large, restricting the amount of storage available for the OS/390 GETMAIN.

Correct the ALLOC or SIZE parameters and restart the client application.

619 IRP_BAD_IOAREA

Explanation: During a DPL request, an I/O area has been supplied to DFHIRP that could not be used.

System Action: The DPL request is terminated, the pipe is forced into a "must close" state, and the external CICS interface takes a system dump.

User Response: This is an external CICS interface error. Contact the IBM support center with details of the return code and the dump.

Note: The pipe is in a "must close" state after this error, and before attempting further calls must first be closed and reopened.

620 IRP_PROTOCOL_ERROR

Explanation: An internal protocol error has occurred whilst trying to communicate with the target CICS system.

System Action: The DPL request is terminated, the pipe is forced into a "must close" state, and the external CICS interface takes a system dump.

User Response: This is an external CICS interface error. Keep the dump and contact your IBM support center.

Note: The pipe is in a "must close" state after this error, and before attempting further calls must first be closed and reopened.

621 PIPE_RECOVERY_FAILURE

Explanation: An error has occurred during an open pipe request. The external CICS interface attempts to recover by disconnecting the pipe again. During this disconnection, further errors have occurred.

System Action: The Open_Pipe call is terminated and the pipe is placed in a "must close" state. The return code from DFHIRP is returned in the EXCI subreason field-1, and a system dump is taken.

User Response: For an explanation of the IRP return codes, see the interregion control blocks in the *CICS Data Areas* manual. The IRP return codes are in the DFHIRSPS copybook, listed under the heading IRC. Use the dump and IRP return codes to determine why the disconnect failed. You may also want to use the EXCI trace to determine the earlier error that caused the open pipe recovery routine to be invoked.

Note: The pipe is now in a "must close" state and if further calls are to be issued, the pipe must be closed and reopened again first.

622 ESTAEX_SETUP_FAILURE

Explanation: In order to protect itself from possible program checks the external CICS interface establishes an OS/390 ESTAEX. In this case, the OS/390 ESTAEX macro has failed.

System Action: The call terminated, and the return code from the OS/390 ESTAEX command is returned in the EXCI subreason field-1. This error may occur before EXCI dump services are initialized, therefore an EXCI issues an OS/390 abend (U0402) to force a SYSDUMP.

User Response: Use the return code and the dump to determine why the ESTAEX command failed. This may be an internal EXCI error and if the problem persists, contact your IBM support center.

623 ESTAEX_INVOKED

Explanation: A program check is encountered during call processing, and the ESTAEX is invoked.

System Action: The program check is handled by the EXCI ESTAEX and an attempt is made to recover to a state that can support further EXCI calls. The OS/390 abend code is returned in the EXCI subreason field-1 of the return area. To aid further diagnosis, a SYSDUMP is taken.

User Response: Use the return code and the dump to determine why a program check occurred in the external CICS interface. The most likely reason for this is that the EXCI code abended whilst trying to access the client program's parameters. Use the EXCI trace to determine if any of the parameters might have caused this error. If this is not the case, this may be an error in the external CICS interface. Keep the dump and contact your IBM support center.

624 SERVER_TIMEDOUT

Explanation: A DPL request has been issued and the target server program has executed in the CICS server region. However, the server program has been executing for longer than the time-out value specified in the DFHXCOPT table.

System Action: The external CICS interface stops waiting for the server program to complete. Because the server program might complete some time after the time-out, and try to respond to the DPL call, the pipe is forced into a “must close” state.

User Response: Determine why the server application program timed out. Either there is a problem with the server program itself (for example, it might be in a loop), or the timeout value is too low.

625 STIMER_SETUP_FAILURE

Explanation: In order to provide a TIMEOUT mechanism, the external CICS interface issues an OS/390 STIMERM macro call. This call has failed.

System Action: The return code from the call is returned in the subreason field-1 of the EXCI return area. The DPL request is terminated and the external CICS interface takes a system dump. The pipe is placed in a “must close” state.

User Response: Use the OS/390 return code and the dump to determine why the call failed. This could be an external CICS interface error. Contact your IBM support center with details of the dump.

Note: The pipe is in a “must close” state after this error, and before attempting further calls must first be closed and reopened.

626 STIMER_CANCEL_FAILURE

Explanation: On successful completion of a DPL request, the cancel of an STIMERM request issued to check the TIMEOUT value has failed with an error.

System Action: The return code from the STIMERM CANCEL is returned in the subreason field-1 of the EXCI return area. The pipe is placed in a “must close” state, and the external CICS interface takes a system dump.

User Response: Use the return code and the dump to determine why the OS/390 STIMERM CANCEL command failed. This could be an external CICS interface error. Contact your IBM support center with details of the dump.

Note: The pipe is in a “must close” state after this error, and before attempting further calls must first be closed and reopened.

628 IRP_LEVEL_CHECK_FAILURE

Explanation: The release level of the module DFHIRP is not at the same, or higher, level than the release level of the external CICS interface.

System Action: The Allocate_pipe request is terminated. The IRP return code (R15) is returned in the EXCI subreason field-1, and the function level of DFHIRP being used is returned in the EXCI subreason field-2. Subreason field-2 is only meaningful if subreason field-1 is zero. The external CICS interface takes a system dump.

User Response: Check the level of the DFHIRP module installed in the LPA. Ensure that it is at least the same as the external CICS interface. The installed level of DFHIRP must be the highest level of CICS or external CICS interface in use in the VSE/ESA image. For more details about installing DFHIRP, see the *CICS Installation Guide*.

629 SERVER_PROTOCOL_ERROR

Explanation: A response to a DPL request has been returned by CICS but the external CICS interface does not understand the response.

System Action: The DPL request is terminated and the external CICS interface takes a system dump.

User Response: Use the dump to determine why the response was in error. The most likely reason for this is that the CICS application server program was not running under the control of a CICS mirror task. This can happen if the transaction definition named by the transid parameter on the DPL call does not specify DFHMIRS as the program name. This would cause unidentified responses being sent from the CICS server region.

Chapter 18. Messages and Codes

For details of all messages and abend codes for the external CICS interface, see the following manuals:

VSE/ESA Messages and Codes — Volume 1
VSE/ESA Messages and Codes — Volume 2
VSE/ESA Messages and Codes — Volume 3

Part 4. Appendixes

Appendix. Routing program-link requests

Important

For detailed information about routing program-link requests, see the *CICS Intercommunication Guide*. This appendix is an overview of how program-link requests received from outside CICS can be routed to other regions.

“Traditional” CICS-to-CICS distributed program link (DPL) calls, instigated by EXEC CICS LINK PROGRAM commands, can be “daisy-chained” from region to region, simply by defining the program as remote in each region except the last (server) region, where it is to execute. The same applies to program-link requests received from *outside CICS*. For example, all of the following types of program-link request can be routed:

- Calls received from:
  - CICS Web support
  - The CICS Transaction Gateway
- Calls from external CICS interface (EXCI) client programs
- External Call Interface (ECI) calls from any of the CICS Client workstation products
- Distributed Computing Environment (DCE) remote procedure calls (RPCs)
- ONC/RPC calls.

Static routing

A program-link request received from outside CICS can be statically routed to a remote CICS region by specifying the name of the remote region on the REMOTESYSTEM option of the installed program definition.

Dynamic routing

A program-link request received from outside CICS can be dynamically routed by:

- Defining the program to CICS as DYNAMIC(YES)
- Coding your dynamic routing program to route the request.

Notices

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

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

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

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

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106, Japan

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

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

Licensees of this program who wish 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 Laboratory Counsel, MP151, IBM United Kingdom Limited, Hursley Park, Winchester, Hampshire, England SO21 2JN.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

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

Programming interface information

This book is intended to help you use the external interfaces provided by the CICS Transaction Server for OS/390. This book documents General-use Programming Interface and Associated Guidance Information provided by CICS.

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

This book also documents Product-sensitive Programming Interface and Associated Guidance Information and Diagnosis, Modification or Tuning Information provided by CICS.

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

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

Diagnosis, Modification, or Tuning Information is provided to help you diagnose problems in your CICS system.

Note: Do not use this Diagnosis, Modification, or Tuning Information as a programming interface.

Diagnosis, Modification, or Tuning Information is identified, where it occurs, by an introductory statement to a chapter or section.

Trademarks and service marks

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

BookManager
C/370
CICS
CICS/ESA
DB2
IBM
IBMLink
IMS
MQ
MQSeries
MVS/ESA
OS/390
OpenEdition
RACF
VTAM

UNIX is a registered trademark in the United States and other countries licensed exclusively through X/Open Company Limited.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and other countries.

Bibliography

CICS Transaction Server for VSE/ESA Release 1 Library

Evaluation and planning	
CICS Release Guide	GC33-1645
CICS Migration Guide	GC33-1646
CICS Report Controller Planning Guide	SC33-1941
CICS Enhancements Guide	GC34-5763
General	
CICS Master Index	SC33-1648
CICS Trace Entries Handbook	SC34-5556
CICS User's Handbook	SX33-6101
CICS Glossary (softcopy only)	GC33-1649
Administration	
CICS System Definition Guide	SC33-1651
CICS Customization Guide	SC33-1652
CICS Resource Definition Guide	SC33-1653
CICS Operations and Utilities Guide	SC33-1654
CICS Supplied Transactions	SC33-1655
Programming	
CICS Application Programming Guide	SC33-1657
CICS Application Programming Reference	SC33-1658
CICS 4.1 Sample Applications Guide	SC33-1713
CICS Application Migration Aid Guide	SC33-1943
CICS System Programming Reference	SC33-1659
CICS Distributed Transaction Programming Guide	SC33-1661
CICS Front End Programming Interface User's Guide	SC33-1662
REXX Guide	SC34-5764
CICS Family: Client/Server Programming	SC33-1435
Diagnosis	
CICS Problem Determination Guide	GC33-1663
CICS Messages and Codes	SC33-6799
CICS Diagnosis Reference	LY33-6085
CICS Data Areas	LY33-6086
CICS Supplementary Data Areas	LY33-6087
Communication	
CICS Internet Guide	SC34-5765
CICS Intercommunication Guide	SC33-1665
CICS Family: Interproduct Communication	SC33-0824
CICS Family: Communicating from CICS on System/390	SC33-1697
Special topics	
CICS Recovery and Restart Guide	SC33-1666
CICS Performance Guide	SC33-1667
CICS Shared Data Tables Guide	GC33-1668
CICS Security Guide	SC33-1942
CICS External Interfaces Guide	SC33-1669
CICS XRF Guide	SC33-1671
CICS Report Controller User's Guide	SC34-5688
CICS Clients	
CICS Clients: Administration	SC33-1792
CICS Universal Clients Version 3 for OS/2: Administration	SC34-5450
CICS Universal Clients Version 3 for Windows: Administration	SC34-5449

CICS Universal Clients Version 3 for AIX: Administration	SC34-5348
CICS Universal Clients Version 3 for Solaris: Administration	SC34-5451
CICS Family: OO programming in C++ for CICS Clients	SC33-1923
CICS Family: OO programming in BASIC for CICS Clients	SC33-1671
CICS Family: Client/Server Programming	SC33-1435
CICS Transaction Gateway Version 3: Administration	SC34-5448

Books from VSE/ESA 2.5 base program libraries

VSE/ESA Version 2 Release 5

Book title	Order number
Administration	SC33-6705
Diagnosis Tools	SC33-6614
Extended Addressability	SC33-6621
Guide for Solving Problems	SC33-6710
Guide to System Functions	SC33-6711
Installation	SC33-6704
Licensed Program Specification	GC33-6700
Messages and Codes Volume 1	SC33-6796
Messages and Codes Volume 2	SC33-6798
Messages and Codes Volume 3	SC33-6799
Networking Support	SC33-6708
Operation	SC33-6706
Planning	SC33-6703
Programming and Workstation Guide	SC33-6709
System Control Statements	SC33-6713
System Macro Reference	SC33-6716
System Macro User's Guide	SC33-6715
System Upgrade and Service	SC33-6702
System Utilities	SC33-6717
TCP/IP User's Guide	SC33-6601
Turbo Dispatcher Guide and Reference	SC33-6797
Unattended Node Support	SC33-6712
e-business Connectors User's Guide	SC33-6719

High-Level Assembler Language (HLASM)

Book title	Order number
General Information	GC26-8261
Installation and Customization Guide	SC26-8263
Language Reference	SC26-8265
Programmer's Guide	SC26-8264

Language Environment for VSE/ESA (LE/VSE)

Book title	Order number
C Run-Time Library Reference	SC33-6689
C Run-Time Programming Guide	SC33-6688
Concepts Guide	GC33-6680
Debug Tool for VSE/ESA Fact Sheet	GC26-8925
Debug Tool for VSE/ESA Installation and Customization Guide	SC26-8798
Debug Tool for VSE/ESA User’s Guide and Reference	SC26-8797
Debugging Guide and Run-Time Messages	SC33-6681
Diagnosis Guide	SC26-8060
Fact Sheet	GC33-6679
Installation and Customization Guide	SC33-6682
LE/VSE Enhancements	SC33-6778
Licensed Program Specification	GC33-6683
Programming Guide	SC33-6684
Programming Reference	SC33-6685
Run-Time Migration Guide	SC33-6687
Writing Interlanguage Communication Applications	SC33-6686

VSE/ICCF

Book title	Order number
Adminstration and Operations	SC33-6738
User’s Guide	SC33-6739

VSE/POWER

Book title	Order number
Administration and Operation	SC33-6733
Application Programming	SC33-6736
Networking Guide	SC33-6735
Remote Job Entry User’s Guide	SC33-6734

VSE/VSAM

Book title	Order number
Commands	SC33-6731
User’s Guide and Application Programming	SC33-6732

VTAM for VSE/ESA

Book title	Order number
Customization	LY43-0063
Diagnosis	LY43-0065
Data Areas	LY43-0104
Messages and Codes	SC31-6493
Migration Guide	GC31-8072
Network Implementation Guide	SC31-6494
Operation	SC31-6495
Overview	GC31-8114
Programming	SC31-6496
Programming for LU6.2	SC31-6497
Release Guide	GC31-8090
Resource Definition Reference	SC31-6498

Books from VSE/ESA 2.5 optional program libraries

C for VSE/ESA (C/VSE)

Book title	Order number
C Run-Time Library Reference	SC33-6689
C Run-Time Programming Guide	SC33-6688
Diagnosis Guide	GC09-2426
Installation and Customization Guide	GC09-2422
Language Reference	SC09-2425
Licensed Program Specification	GC09-2421
Migration Guide	SC09-2423
User’s Guide	SC09-2424

COBOL for VSE/ESA (COBOL/VSE)

Book title	Order number
Debug Tool for VSE/ESA Fact Sheet	GC26-8925
Debug Tool for VSE/ESA Installation and Customization Guide	SC26-8798
Debug Tool for VSE/ESA User’s Guide and Reference	SC26-8797
Diagnosis Guide	SC26-8528
General Information	GC26-8068
Installation and Customization Guide	SC26-8071
Language Reference	SC26-8073
Licensed Program Specifications	GC26-8069
Migration Guide	GC26-8070
Migrating VSE Applications To Advanced COBOL	GC26-8349
Programming Guide	SC26-8072

DB2 Server for VSE

Book title	Order number
Application Programming	SC09-2393
Database Administration	GC09-2389
Installation	GC09-2391
Interactive SQL Guide and Reference	SC09-2410
Operation	SC09-2401
Overview	GC08-2386
System Administration	GC09-2406

DL/I VSE

Book title	Order number
Application and Database Design	SH24-5022
Application Programming: CALL and RQDLI Interface	SH12-5411
Application Programming: High-Level Programming Interface	SH24-5009
Database Administration	SH24-5011
Diagnostic Guide	SH24-5002
General Information	GH20-1246
Guide for New Users	SH24-5001
Interactive Resource Definition and Utilities	SH24-5029
Library Guide and Master Index	GH24-5008
Licensed Program Specifications	GH24-5031
Low-level Code and Continuity Check Feature	SH20-9046
Library Guide and Master Index	GH24-5008
Messages and Codes	SH12-5414
Recovery and Restart Guide	SH24-5030
Reference Summary: CALL Program Interface	SX24-5103
Reference Summary: System Programming	SX24-5104
Reference Summary: HLPI Interface	SX24-5120
Release Guide	SC33-6211

PL/I for VSE/ESA (PL/I VSE)

Book title	Order number
Compile Time Messages and Codes	SC26-8059
Debug Tool For VSE/ESA User’s Guide and Reference	SC26-8797
Diagnosis Guide	SC26-8058
Installation and Customization Guide	SC26-8057
Language Reference	SC26-8054

Book title	Order number
Licensed Program Specifications	GC26-8055
Migration Guide	SC26-8056
Programming Guide	SC26-8053
Reference Summary	SX26-3836

Screen Definition Facility II (SDF II)

Book title	Order number
VSE Administrator's Guide	SH12-6311
VSE General Introduction	SH12-6315
VSE Primer for CICS/BMS Programs	SH12-6313
VSE Run-Time Services	SH12-6312

TCP/IP for VSE/ESA

Documentation for TCP/IP for VSE/ESA is available by ordering the ″TCP/IP for VSE/ESA PDF Product Library″ product kit, SK2T-1336. **Not** all the books are separately orderable as they are only available in PDF form. The following titles are available on the product kit at the time of going to press:

Book title	Order number
TCP/IP for VSE/ESA — IBM Program Setup and Supplementary Information	SC33-6601
TCP/IP for VSE/ESA Flyer	G221-9030
TCP/IP for VSE/ESA Performance Considerations	n/a
TCP/IP for VSE/ESA LPS	GC33-6594
The Native TCP/IP forVSE	GC33-6594
TCP/IP Tutorial and Technical Overview	GG24-3376
VSE/ESA as a Web Server	SG24-2040
TCP/IP for VSE 1.4 Installation Guide	n/a
TCP/IP for VSE 1.4 User's Guide	n/a
TCP/IP for VSE 1.4 Commands	n/a
TCP/IP for VSE 1.4 Programmer's Reference	n/a
TCP/IP for VSE 1.4 Messages and Codes	n/a
TCP/IP for VSE 1.4 Optional Products	n/a

Determining if a publication is current

IBM regularly updates its publications with new and changed information. When first published, both hardcopy and BookManager softcopy versions of a publication are in step, but subsequent updates will probably be available in softcopy before they are available in hardcopy.

For CICS for VSE/ESA books, these softcopy updates appear regularly on the *Transaction Processing and Data Collection Kit* CD-ROM, SK2T-0730-xx and on the

VSE/ESA Collection Kit CD-ROM, SK2T-0060-xx. Each reissue of the collection kit is indicated by an updated order number suffix (the -xx part). For example, collection kit SK2T-0730-06 is more up-to-date than SK2T-0730-05. The collection kit is also clearly dated on the cover.

Here's how to determine if you are looking at the most current copy of a publication:

- A publication with a higher suffix number is more recent than one with a lower suffix number. For example, the publication with order number SC33-0667-02 is more recent than the publication with order number SC33-0667-01. (Note that suffix numbers are updated as a product moves from release to release, as well as for hardcopy updates within a given release.)
- When the softcopy version of a publication is updated for a new collection kit the order number it shares with the hardcopy version does not change. Also, the date in the edition notice remains that of the original publication. To compare softcopy with hardcopy, and softcopy with softcopy (on two editions of the collection kit, for example), check the last two characters of the publication's filename. The higher the number, the more recent the publication. For example, DFHPF104 is more recent than DFHPF103. Next to the publication titles in the CD-ROM booklet and the readme files, asterisks indicate publications that are new or changed.
- Updates to the softcopy are clearly marked by revision codes (usually a "#" character) to the left of the changes.

Index

Numerics

3270 bridge 19
- abend processing 22
- ADS descriptor 23
- ASSIGN 45
- benefits 10
- BMS Application Data Structure 23
- BRDATA format 58
- BREXIT 44
- bridge design 75
- bridge environment 21, 39
- bridge exit 21
- bridge exit area 22, 84
- bridge exit calls 78
- bridge exit interfaces 84
- bridge facility 24
- bridge facility properties 49
- bridge mechanism 20
- bridge monitor 21
- CEMT INQUIRE TASK 47
- CEMT INQUIRE TRANSACTION 48
- client application 20
- conversational Transactions 29
- DLI 41
- Dump 109
- Example environments 30
- FACILITYLIKE 24
- formatter 22
- formatter calls 81
- formatter return codes 82
- INQUIRE_CONTEXT 48
- INQUIRE TASK 46
- INQUIRE_TRANDEF 48
- INQUIRE TRANSACTION 47
- ISSUE PRINT 41
- message data format 59
- messages 21
- Monitoring 41
- passthrough tool 25
- pseudoconversational transactions 28
- resource definition 44
- Security 41
- simple terminal transactions 27
- START 42
- STARTed transactions 43
- Supplied bridge exits 55
- supplied copybooks 25
- supplied resource definitions 25
- syncpoints 43
- TCTUA 24, 43
- Trace 108
- TRANSACTION definition 44
- transport mechanism 21
- user transaction 20
- user transaction restrictions 39
- Using DFH0CBREt 55
- Using the Web bridge exit 57

3270 bridge 19 *(continued)*
- writing a bridge exit 78
- writing a formatter 81

A

abend codes 174

addressing mode (AMODE)
- assembler sample program 162
- client program requirements 161

ADS 23

ADS descriptor 23

allocate_opts, parameter of ALLOCATE_PIPE command 124

Allocate_Pipe command 123

ALLOCATE_PIPE command
- invocation of DFHXCURM during 157
- security check failure 169

allocating a pipe 123

application programming
- commands 119
- copybooks 139
- DPL subset 114
- exception conditions returned on LINK command 146
- language considerations 161
- RESP and RESP2 fields 146
- restrictions for server programs 114
- sample programs 162
- stub 161
- translation required for EXEC CICS LINK command 150

applid, specifying on ALLOCATE_PIPE command 123

assembler
- copybook 139
- EXCI CALL interface 119
- sample program 141, 162

automatic retry of EXEC CICS LINK 148

B

basic sequential access method (BSAM)
- use of by assembler sample client program 162

batch jobs, querying the status of 156

benefits of external CICS interface 10

bind-time security 169

BMS Application Data Structure 23

BREXIT option
- CEMT INQUIRE TRANSACTION 48
- INQUIRE TRANSACTION command 47

bridge (3270) 19
- abend processing 22
- ADS descriptor 23
- ASSIGN 45
- benefits 10
- BMS Application Data Structure 23
- BRDATA format 58
- BREXIT 44

bridge (3270) 19 *(continued)*
  bridge designr 75
  bridge environment 21, 39
  bridge exit 21
  bridge exit area 22, 84
  bridge exit calls 78
  bridge exit interfaces 84
  bridge facility 24
  bridge facility properties 49
  bridge mechanism 20
  bridge monitor 21
  CEMT INQUIRE TASK 47
  CEMT INQUIRE TRANSACTION 48
  client application 20
  conversational transactions 29
  DLI 41
  Dump 109
  Example environments 30
  FACILITYLIKE 24
  formatter 22
  formatter return codes 82
  formattert calls 81
  INQUIRE_CONTEXT 48
  INQUIRE TASK 46
  INQUIRE_TRANDEF 48
  INQUIRE TRANSACTION 47
  ISSUE PRINT 41
  message data format 59
  messages 21
  Monitoring 41
  passthrough tool 25
  pseudoconversational transactions 28
  resource definition 44
  running a transaction 27
  Security 41
  simple terminal transactions 27
  START 42
  STARTed transactions 43
  Supplied bridge exits 55
  supplied copybooks 25
  supplied implementations 25
  supplied resource definitions 25
  syncpoints 43
  TCTUA 24, 43
  Trace 108
  TRANSACTION definition 44
  transport mechanism 21
  user transaction 20
  user transaction restrictions 39
  Using DFH0CBRE 55
  Using the Web bridge exit 57
  writing a bridge exit 78
  writing a formatter 81
bridge exit
  BRXA ADS descriptor area 100
  BRXA command area (BMS) 97
  BRXA command area (common) 92
  BRXA command area (interval control) 99
  BRXA command area (MSG) 99
  BRXA command area (syncpoint) 99
  BRXA command area (terminal control) 96
bridge exit *(continued)*
  BRXA header area 84
  BRXA transaction area 86
  message format 59
  message vectors 59
  MQCIH message header 59
  START data format 58
BRIDGE option
  CEMT INQUIRE TASK 47
  INQUIRE TASK command 46
BRXA
  ADS descriptor area 100
  command area (BMS) 97
  command area (common) 92
  command area (interval control) 99
  command area (syncpoint) 99
  command area (terminal control) 96
  header area 84
  transaction area 86

C

C language
  copybook 139
  EXCI CALL interface 119
  sample program 141, 162
  special considerations for client program 161
CA21 25
call_type
  parameter of ALLOCATE_PIPE command 123
  parameter of CLOSE_PIPE command 134
  parameter of DEALLOCATE_PIPE command 136
  parameter of DPL_Request command 127
  parameter of INITIALIZE_USER command 121
  parameter of OPEN_PIPE command 126
CEMT INQUIRE EXCI command 156
CICS_applid, parameter of ALLOCATE_PIPE command 123
CICS system definition file, EXCI sample definitions 163
CICS system initialization parameters
  VTAM 25, 45
client program
  addressing mode 161
  BSAM, use of by assembler sample program 162
  connection through DFHIRP 117
  definition of 113
  JCL needed
    running an EXCI client 166
  linking to server with EXEC CICS LINK 144
  MRO logon and bind-time security 169
  PL/I and C language considerations 161
  sample job for starting 167
  sample program 162
  translating 150
  use of multiple sessions 113
Close_Pipe command 134
closing a pipe 134
COBOL II
  copybook 139
  example of EXCI DPL call 140

COBOL II *(continued)*
  EXCI CALL interface 119
  sample program 162
codes, abend 174
COMMAREA, parameter of DPL_Request command 128
COMMAREA_len, parameter of DPL_Request command 128
connecting an allocated pipe 125
CONNECTION definition
  CONNTYPE attribute 151
  PROTOCOL attribute 152
connection-oriented data transmission 8
connectionless data transmission 8
CONNTYPE attribute, CONNECTION definition 151
copybooks for assembler, C language, COBOL, PL/I 139
cross-system multiregion operation (XCF/MRO) 113
CSMI
  attached by CICS server 146
CSMI (CICS-supplied mirror transaction)
  authorizing the link user ID 170
  default transid 129
  security 170

D
data_len, parameter of DPL_Request command 128
datagram 8
DCE (distributed computing environment) 10
Deallocate_Pipe command 136
deallocating a pipe 136
DFH$AXCC, assembler sample program
  example of output 164
DFH¢AXCC, assembler sample program 141
DFH¢DXCC, C sample program 141
DFH¢PXCC, PL/I sample program 141
DFH$EXCI, sample server definitions 163
DFH$FILA, sample file definitions 164
DFH0CBRD 25
DFH0CBRE 25
DFH0CBRF 25
DFH0CBRU 25
DFHAUPLE procedure 159
DFHBMSUP utility program
  introduction 76
DFHBRMHx 25
DFHBRMQx 25
DFHBRSDx 25
DFHIRP (interregion communication program)
  connection of client and server 117
  security checks performed by 169
DFHLIST, note about sample definitions 163
DFHWBLT 25
DFHXCIE, alias for DFHXCSTB stub 161
DFHXCIS, alias for DFHXCSTB stub 161
DFHXCOPT, options table 159
DFHXCPLD, return area and equate copybook for assembler 139
DFHXCPLH, return area and equate copybook for C language 139
DFHXCPLL, return area and equate copybook for PL/I 139
DFHXCPLO, return area and equate copybook for COBOL 139
DFHXCRCD, return code copybook for assembler 139
DFHXCRCH, return code copybook for C language 139
DFHXCRCL, return code copybook for PL/I 139
DFHXCRCO, return code copybook for COBOL 139
DFHXCSTB, stub for client programs 161
DFHXCTRA, EXCI service trap 177
DFHXCTRD, parameter list 177
DFHXCURM, user-replaceable module 157
disconnecting a pipe 134
distributed application design 11
distributed computing environment (DCE) 10
distributed program link (DPL)
  API subset for server programs 114
  example COBOL call without userid and uowid 140
  request program call 127
dotted decimal 9
DPL_opts, parameter of DPL_Request command 132
DPL_Request call 127
dpl_retarea, parameter of DPL_Request command 131
dumps 173
  formatting 173

E
ELPA (extended link pack area), installation of DFHIRP 118
ephemeral port numbers 9
EQUATE copybooks 139
exception conditions returned on LINK command 146
EXCI on CEMT INQUIRE command 156
EXEC CICS LINK command 144
  automatic retry 148
  choosing between EXEC CICS and CALL interface 114
  security checking 170
  translation 150
extended link pack area (ELPA), installation of DFHIRP 118
external CICS interface (EXCI) 113, 143
  benefits 10
  CALL interface
    choosing between EXEC CICS and CALL interface 114
    return area 138
    syntax 119
  CICS releases supported 117
  compiling and link-editing client programs 161
  defining connections 151
  description of 113
  inquiring on the state of connections 155
  languages supported 119
  messages 199
  options table, DFHXCOPT 159
  PL/I and C language considerations 161
  problem determination 173

external CICS interface (EXCI) 113, 143 *(continued)*
programming languages supported 119
reason codes 189
response codes 189
security 169
user-replaceable module (DFHXCURM) 157

F
FACILITYLIKE 24
FACILITYLIKE option
CEMT INQUIRE TRANSACTION 48
INQUIRE TRANSACTION command 47
File Transfer Protocol 9
freeing storage associated with a pipe 136
function call EQUATE copybooks 139

G
generic connection
definition of 152
note about lack of security checks 169

I
IDENTIFIER
bridge exit area 89
IDENTIFIER option
CEMT INQUIRE TASK 48
INQUIRE TASK command 46
Initialize_User command 120
internet address 8
Internet Protocol (IP) 8
interregion communication (IRC)
opening after installation of sample definitions 164

J
job control language (JCL)
for running an EXCI client program 166

L
LINK command 144
choosing between EXEC CICS and CALL interface 114
link-editing
DFHXCOPT options table 159
for client program 161
translation required for EXEC CICS LINK command 150
use of DFHXCSTB stub 119
using DFHAUPLE 159
logon security 169

M
messages 108, 199
mirror transaction
authorizing the link user ID 170
security 170
MSGCASE, parameter of DFHXCOPT 159
multiregion operation (MRO)
cross-system (XCF/MRO) 113
logon and bind time security 169

N
null parameters, example of EXCI CALLs with 140

O
Open_Pipe command 125
open system interface (OSI) 10
opening a pipe 125
options table, DFHXCOPT 159
OSI (open system interface) 10

P
parameters
null 140
specifying with options table 159
pgmname
parameter of DPL_Request command 128
pipe
allocating 123
closing 134
connecting 125
deallocating 136
definition of 113
disconnecting 134
freeing storage associated with 136
invocation of DFHXCURM during ALLOCATE_PIPE 157
opening 125
restriction on leaving open 125
reusing a closed pipe 134
pipe_token
parameter of ALLOCATE_PIPE command 123
parameter of CLOSE_PIPE command 134
parameter of DEALLOCATE_PIPE command 136
parameter of DPL_Request command 128
parameter of OPEN_PIPE command 126
PL/I
copybook 139
EXCI CALL interface 119
sample program 141, 162
special considerations for client program 161
port number 8
problem determination 173
dumps 173
service trap 177
trace 173
VSE abends 174
programming models 9
programming restrictions for server programs 114
PROTOCOL attribute
CONNECTION definition 152
SESSIONS definition 153

R
reason codes 189
Allocate_Pipe call 124

reason codes 189 *(continued)*
  Close_Pipe call 134
  Deallocate_Pipe call 136
  DPL call 132
  Initialize_User call 121
  Open_Pipe call 126
RECEIVECOUNT attribute, SESSIONS definition 153
RECEIVEPFX attribute, SESSIONS definition 154
residence mode (RMODE)
  assembler sample program 162
resource access control facility (RACF) 169
  specifying userid on DPL_Request command 130
resource definition
  CONNECTION definition 151
  sample programs 163
  SESSIONS definition 153
RESP and RESP2 fields 146
response codes 138, 189
  Allocate_Pipe call 124
  Close_Pipe call 134
  Deallocate_Pipe call 136
  DPL call 132
  Initialize_User call 121
  Open_Pipe call 126
retries on an EXEC CICS LINK command 148
return_area
  parameter of ALLOCATE_PIPE command 123
  parameter of CLOSE_PIPE command 134
  parameter of DEALLOCATE_PIPE command 136
  parameter of DPL_Request command 127
  parameter of INITIALIZE_USER command 120
  parameter of OPEN_PIPE command 125
reusing a closed pipe 134
running the sample applications 164

S

sample programs 141, 162
  description 162
security 169
SENDCOUNT attribute, SESSIONS definition 154
SENDPFX attribute, SESSIONS definition 155
server program
  API restrictions 114
  connection through DFHIRP 117
  definition of 113
  DPL subset 114
  linking from client with EXEC CICS LINK 144
  programming restrictions 114
  sample program 162
  security considerations 170
service trap 177
SESSIONS definition
  PROTOCOL attribute 153
  RECEIVECOUNT attribute 153
  RECEIVEPFX attribute 154
  SENDCOUNT attribute 154
  SENDPFX attribute 155
sockets interface 8
specific connection
  definition of 151
  MRO logon security checks 169
storage, freeing 136
stub for client programs
  DFHXCIE 161
  DFHXCIS 161
  DFHXCSTB 161
sysplex, use of cross-system MRO 113
system definition file (CSD), CICS 163

T

TCP/IP 7
Telnet 9
TIMEOUT, parameter of DFHXCOPT 159
trace 173
  TRACE parameter of DFHXCOPT 160
  trace points 177
  TRACESZE parameter of DFHXCOPT 160
transid, parameter of DPL_Request command 129
translation of EXEC CICS LINK command 150
Transmission Control Protocol (TCP) 8
trap, DFHXCTRA 177
  TRAP, parameter of DFHXCOPT 160
TYPE, parameter of DFHXCOPT 159

U

unit-of-work identifier, DPL_Request 129
uowid, parameter of DPL_Request 129
User Datagram Protocol (UDP) 8
user environment, initializing 120
user_name, parameter of INITIALIZE_USER command 121
user-replaceable module
  DFHXCURM 157
user security 170
user_token
  parameter of ALLOCATE_PIPE command 123
  parameter of CLOSE_PIPE command 134
  parameter of DEALLOCATE_PIPE command 136
  parameter of DPL_Request command 127
  parameter of INITIALIZE_USER command 120
  parameter of OPEN_PIPE command 125
userid, parameter of DPL_Request command 130
utility programs, offline
  recreating BMS macro statements, DFHBMSUP 76

V

version_number
  parameter of ALLOCATE_PIPE command 123
  parameter of CLOSE_PIPE command 134
  parameter of DEALLOCATE_PIPE command 136
  parameter of DPL_Request command 127
  parameter of INITIALIZE_USER command 120
  parameter of OPEN_PIPE command 125
VSE abends 174
VTAM system initialization parameter 25, 45

W

well-known ports 9

X

XFAINTU, global user exit 52

Sending your comments to IBM

If you especially like or dislike anything about this book, please use one of the methods listed below to send your comments to IBM.

Feel free to comment on what you regard as specific errors or omissions, and on the accuracy, organization, subject matter, or completeness of this book.

Please limit your comments to the information in this book and the way in which the information is presented.

To request additional publications, or to ask questions or make comments about the functions of IBM products or systems, you should talk to your IBM representative or to your IBM authorized remarketer.

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

You can send your comments to IBM in any of the following ways:

- By mail, to this address:

  Information Development Department (MP095)
  IBM United Kingdom Laboratories
  Hursley Park
  WINCHESTER,
  Hampshire
  United Kingdom

- By fax:
  - From outside the U.K., after your international access code use 44–1962–870229
  - From within the U.K., use 01962–870229
- Electronically, use the appropriate network ID:
  - IBM Mail Exchange: GBIBM2Q9 at IBMMAIL
  - IBMLink™: HURSLEY(IDRCF)
  - Internet: idrcf@hursley.ibm.com

Whichever you use, ensure that you include:

- The publication number and title
- The topic to which your comment applies
- Your name and address/telephone number/fax number/network ID.

Program Number: 5648-054

Printed in the United States of America
on recycled paper containing 10%
recovered post-consumer fiber.

SC33-1669-01

Spine information:

IBM CICS TS for VSE/ESA CICS External Interfaces Guide Release 1