CICS Transaction Server for z/OS
Version 6

Using EXCI

Note

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

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

The beta version of IBM CICS Transaction Server for z/OS might be referred to in the product and documentation as CICS TS for z/OS, beta or 6.3.

Contents

About this PDF

This PDF describes how you can use the EXternal CICS Interface (EXCI) to make the services of CICS Transaction Server for z/OS available to external programs. Before CICS TS 5.4, this information was in the *External Interfaces Guide*.

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

Date of this PDF

This PDF was created on 2026-01-15 (Year-Month-Date).

Chapter 1. How external programs access CICS through EXCI

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

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

The provision of this programming interface means that, for example, z/OS 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 z/OS job. For example, you can:
  - Open and close CICS files.
  - Enable and disable transactions in CICS (and so eliminate the need for a main 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.

Chapter 2. The external CICS interface

The external CICS interface (EXCI) is an application programming interface that enables a non-CICS program (a client program) running in z/OS to call a program (a server program) running in a CICS region and to pass and receive data by using a communications area or by using a channel and a set of containers. The CICS application program is started as if linked to by another CICS application program.

You can use the external CICS interface to allocate and open sessions, or *pipes* (a one-way communication path between a sending process and a receiving process) to a CICS region, and to pass distributed program link (DPL) requests over them. The multiregion operation (MRO) facility of CICS interregion communication (IRC) facility supports these requests, and each pipe maps onto one MRO session, where the client program represents the sending process and the CICS server region represents the receiving process.

There is a default limit of 100 pipes per EXCI address space; the limit can be changed when z/OS is IPLed. This limit prevents EXCI clients monopolizing MRO resources, which could prevent CICS regions from using MRO. The limit is applied in both MRO and cross-system MRO (XCF/MRO) environments. An **allocate_pipe** request results in an **MRO LOGON** request being issued, and there is a limit on the total number of **MRO LOGON** requests allowed from all address spaces. This is critical when you are using XCF/MRO, where the limit on the number of members in an XCF group also limits the total number of **MRO LOGON** requests.

Therefore, when a pipe is no longer needed, you must ensure that the pipe is closed and deallocated. It is the user's responsibility to ensure that the client program closes and deallocates the pipe that it has opened, before the program terminates. Failure to deallocate pipes can lead to errors resulting from the logon limit being reached.

Note: After a successful **Open_Pipe** request, when your client program finishes using the pipe, you must first issue a **Close_Pipe** command and then a **Deallocate_Pipe** command to free the pipe. If you issue a **Deallocate_Pipe** command without first closing an open pipe with **Close_Pipe**, your request fails with USER_ERROR (RC12) with reason code 405 PIPE_NOT_CLOSED.

If your Open_Pipe request fails, it is recommended to deallocate the pipe to avoid the logon limit being reached.

The external CICS interface identifies the CICS region to communicate with by the CICS region APPLID, as defined in the **APPLID** system initialization parameter. You can specify the APPLID either on an EXCI API call or by using the DFHXCURM user-replaceable program. You can also use DFHXCURM to change the value of XCFGROUP to be used on an **allocate_pipe** request. For more information about DFHXCURM, see The EXCI user-replaceable module.

Note: Do not confuse the term *generic APPLID* with *generic resource name*. Generic resource names apply only to z/OS Communications Server generic resource groups, which are not supported by EXCI.

The client program and the CICS server region (the region where the server program runs or is defined) must be in the same z/OS image unless under the following conditions:

- The CICS region is running in a sysplex that supports cross-system MRO.
- All DPL requests issued by the client program specify the SYNCONRETURN option.

Alternatively, if there is no local CICS region in the z/OS image, you must specify the SVC parameter that the external CICS interface is to use, by coding a CICSSVC parameter in the DFHXCOPT table. Although the external CICS interface does not support the cross-memory access method, it can use the XCF access method provided by the CICS XCF/MRO facility.

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 z/OS address space without knowledge of, or interference from, each other.

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

API restrictions for server programs: A CICS server program invoked by an external CICS interface request is restricted to the DPL subset of the CICS application programming interface. This subset (the DPL subset) of the API commands is the same as for a CICS-to-CICS server program. See Distributed Program Link (DPL) for details of the DPL subset for server programs.

The EXCI programming interfaces

The external CICS interface provides two forms of programming interface:

- The EXCI CALL interface
- The EXCI EXEC CICS interface

The EXCI programming interfaces

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

EXCI CALL interface

The EXCI CALL interface consists of six commands that you can use for the following actions:

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

Here is the list of the six EXCI commands:

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

For detailed descriptions of these commands and an EXCI CALL example, see "The EXCI CALL interface" on page 11.

EXEC CICS interface

The EXEC CICS interface provides several commands.

For example, the interface provides a single, composite command, **EXEC CICS LINK PROGRAM** , that performs all six commands of the EXCI CALL interface in one invocation. Each time you issue an **EXEC CICS LINK PROGRAM** command in a client application program, the external CICS interface invokes each of the six EXCI calls on your behalf.

The **EXEC CICS LINK PROGRAM** command is similar but not identical to the distributed program link command of the CICS command-level application programming interface.

EXCI also provides the ability to process data using channel and container commands. A channel together with its set of containers can then be passed on the **EXEC CICS LINK PROGRAM** command or on a call API **DPL_REQUEST** , as an alternative to using a communications area to transfer data or information from one program to another.

For detailed descriptions of the commands that are available with the EXEC CICS interface, see "The EXCI EXEC CICS interface" on page 37.

API restrictions for server programs: A CICS server program invoked by an external CICS interface request is restricted to the DPL subset of the CICS application programming interface. This subset (the

DPL subset) of the API commands is the same as for a CICS-to-CICS server program. See Distributed Program Link (DPL) for details of the DPL subset for server programs.

Choosing between the EXEC CICS and the CALL interface

You can use both the CALL interface (all six commands) and the **EXEC CICS LINK** command in the same program, to perform separate requests. As a general rule, it is unlikely that you would want to do this in a production program.

EXCI sample programs illustrates the various language versions of the CICS-supplied sample client program.

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

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

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

  Note that each invocation of an **EXEC CICS LINK** command causes the external CICS interface to perform all the functions of the CALL interface, which results in unnecessary overhead.

  Note also that this overhead is greatly increased if you use the **EXEC CICS LINK** command to communicate with a CICS server region in a different LPAR. In this case each invocation of the **EXEC CICS LINK** command generates a great deal of XCF activity because of the IRP logon, connect, disconnect and logoff which is required. You might find that you experience severe degradation of elapsed time between **EXEC CICS LINK** commands issued to a CICS server region in a separate LPAR, compared to the elapsed time of the same commands issued to a CICS server region in the same LPAR.
- For multiple or frequent DPL requests from the same client program, you are recommended to use the EXCI CALL interface.

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

Illustrations of the external CICS CALL interface

These four diagrams illustrate the external CICS interface using the EXCI CALL interface.

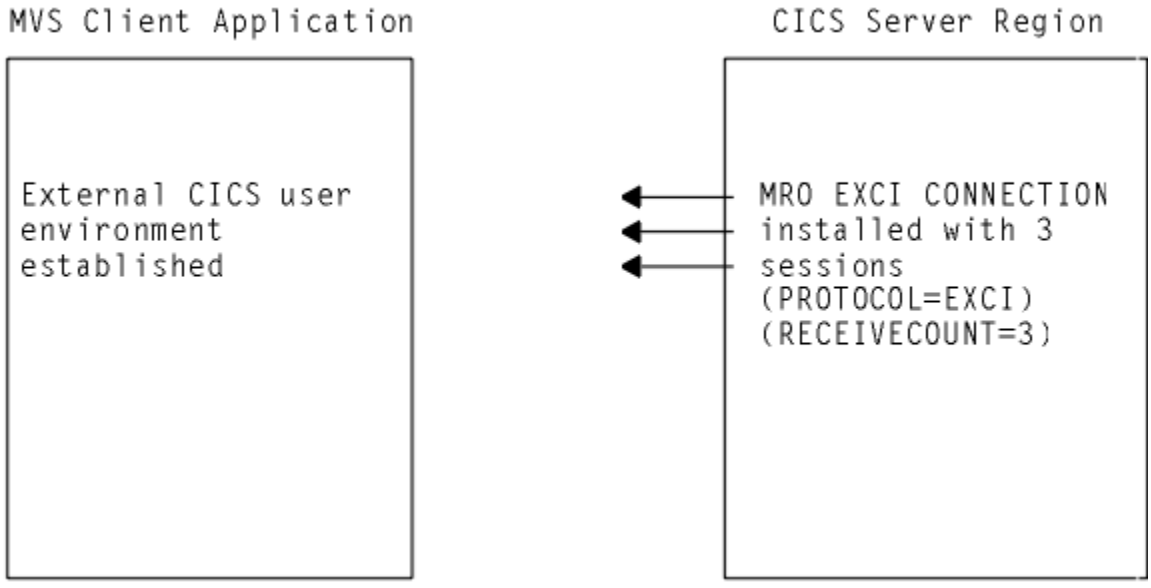


Figure 1. Stage 1: Status after an INITIALIZE_USER call

Note:

1. In Figure 1 on page 5, 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.

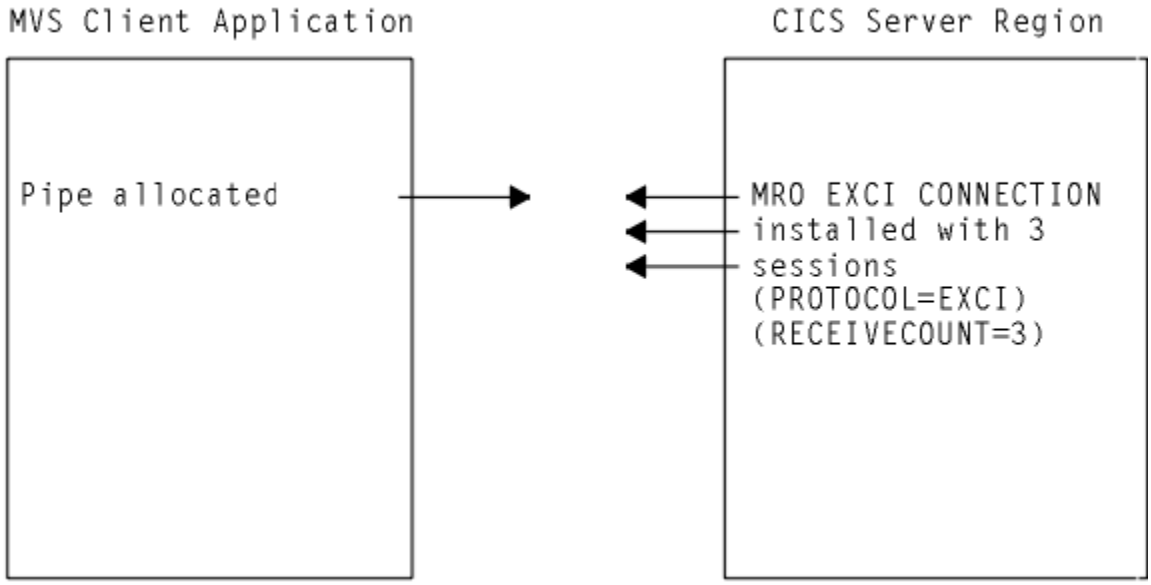


Figure 2. Stage 2: Status after the first ALLOCATE_PIPE call

Note: In Figure 2 on page 6, the external CICS interface logs on to MRO, identifying the target CICS server region.

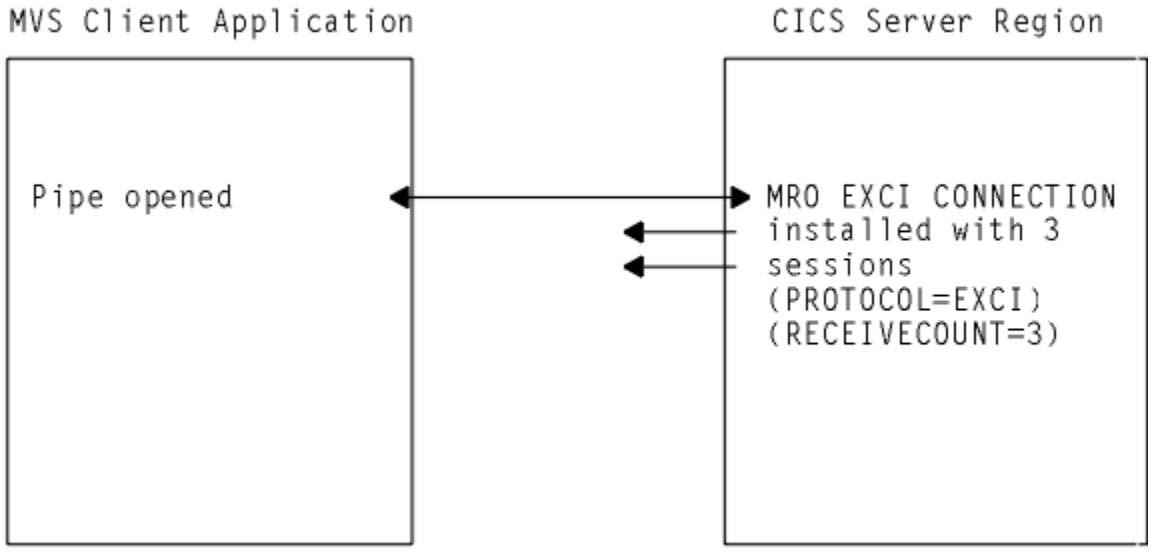


Figure 3. Stage 3: Status after the OPEN_PIPE call

Note:

1. In Figure 3 on page 6, 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).

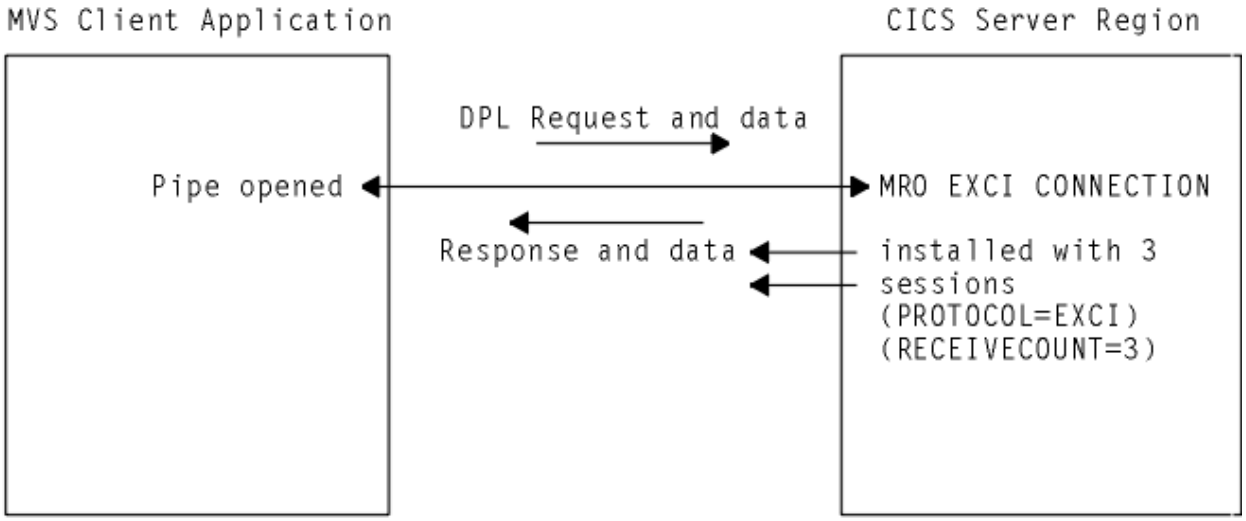


Figure 4. Stage 4: Status with one open pipe, processing a DPL call

Note: In Figure 4 on page 7, 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 2 on page 6. 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 1 on page 5.

Illustration of the EXCI EXEC CICS interface

This diagram illustrates the EXEC CICS interface, and how it resolves to the six EXCI CALLs.

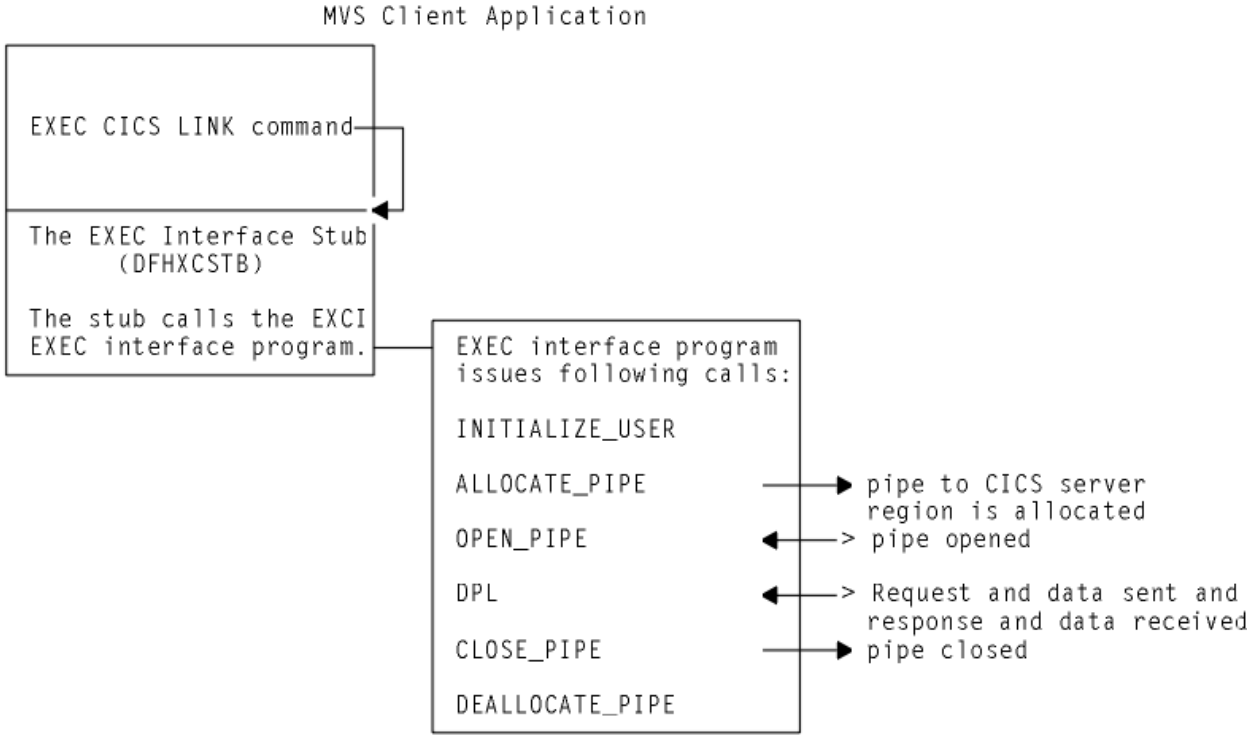


Figure 5. Illustration of the external CICS interface using the EXEC CICS command

Resource recovery

Resource recovery consists of the protocols and program interfaces that allow an application program to make consistent changes to multiple protected resources. The external CICS interface supports resource recovery.

A CICS server program that is invoked by an external CICS interface request can update recoverable resources; the changes are committed when the mirror transaction in the CICS server region takes a sync point. The client program can determine when sync pointing should occur. There are two options:

- Resource recovery controlled by the CICS server regions. In this case, changes to recoverable resources are committed at the completion of each DPL request, independently of the client program. Also, in addition to the sync point taken when the server program returns control to CICS (the SYNCONRETURN option), the server program can take explicit sync points during execution.
- Resource recovery controlled by the EXCI client program with the support of recoverable resource management services (RRMS). When the client program requests it, updates made by the server program in successive DPL requests are committed together.

  To support this option, CICS and the external CICS interface both use resource recovery services (RRS), the z/OS sync point manager [1] , which is a z/OS component of recoverable resource management services (RRMS). In the context of RRMS, CICS is a **resource manager** ; the client program can issue requests to other resource managers, and have resources owned by those resource managers committed in the same **unit-of-recovery (UR).** [2]

These options are controlled as follows:

- By the DPL_opts parameter of the DPL_request.
- By the SYNCONRETURN option, either specified or omitted, on the **EXEC CICS LINK PROGRAM** command.

If you specify SYNCONRETURN, a sync point is taken on completion of each DPL request. If SYNCONRETURN is omitted, a sync point is taken when the client program requests it using the interfaces described in “Use of sync points in the client program” on page 11.

Use of RRMS with the external CICS interface

You can use z/OS recoverable resource management services (RRMS) to coordinate distributed program link (DPL) requests.

To use RRMS to coordinate DPL requests, ensure that the following conditions are met:

- The EXCI client and the CICS region to which it sends DPL requests run in the same z/OS image. This is an RRMS restriction, and does not apply to DPL requests that specify SYNCONRETURN.
- The CICS region that receives the DPL requests is started with **RRMS=YES** specified as a system initialization parameter (the default is **RRMS=NO**).
- Resource recovery services (RRS) run in the z/OS image where CICS and the client program execute. See z/OS MVS Programming: Resource Recovery.

The following figure shows how the external CICS interface and CICS use RRMS. It shows the flow between the z/OS batch region that contains the external CICS interface and the EXCI client program, and the CICS server region that contains the CICS mirror and a CICS application program. The numbers on the diagram refer to the principal steps in a unit of recovery (UR), as listed after the figure.

[1] RRMS comprises three z/OS components: registration services, context services, and resource recovery services (RRS)

[2] A unit of recovery is analogous to a CICS unit of work

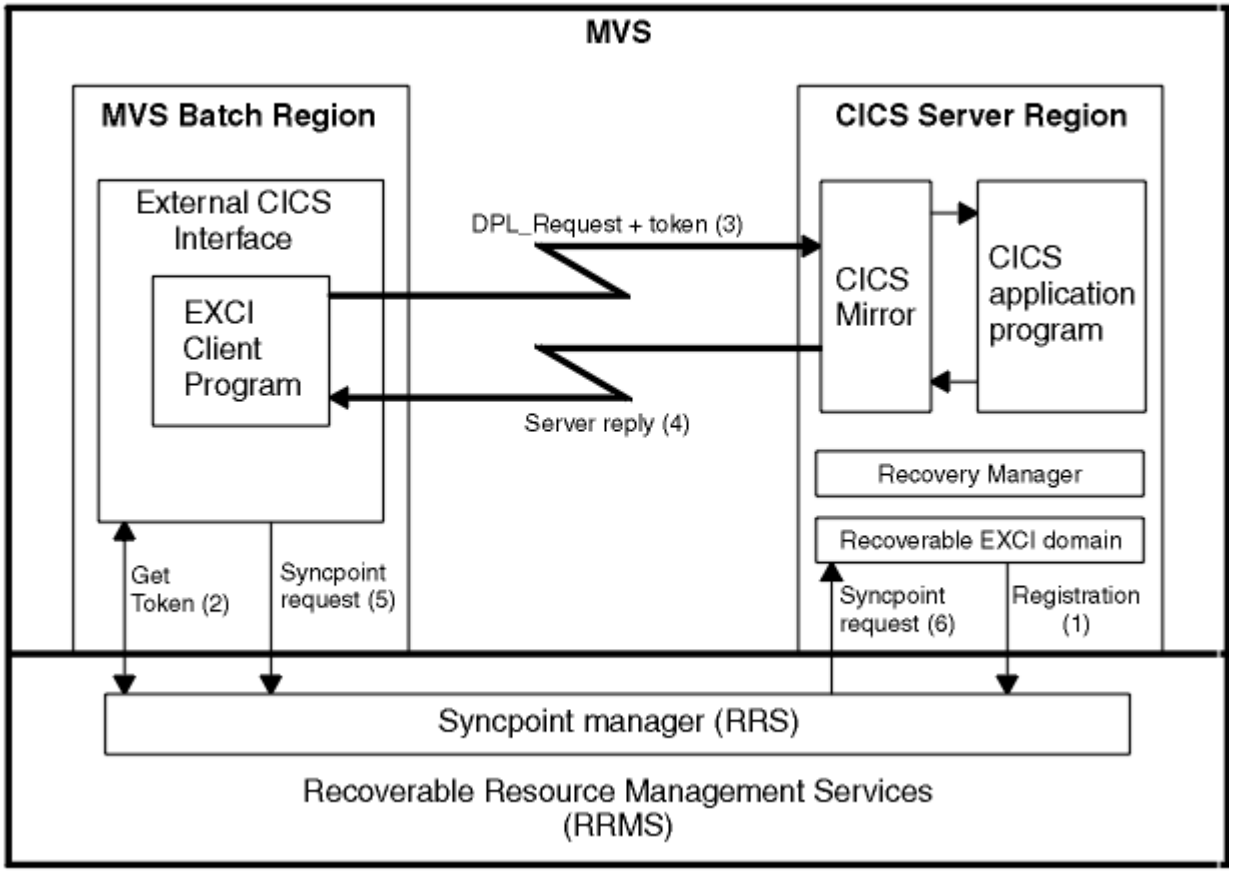


Figure 6. Conceptual view of EXCI client and CICS server region using RRMS

1. If the CICS system initialization parameter **RRMS=YES** is specified, CICS registers with RRMS as a resource manager. This registration occurs during CICS initialization.
2. When the EXCI client program issues a DPL_Request call in 2-phase commit mode (a call that omits the SYNCONRETURN option), it receives the following from RRMS:
   - A unit-of-recovery identifier (URID)
   - A context token
   - A pass token
3. The URID and the tokens obtained by EXCI on behalf of the client program are included on the DPL request that is passed to the CICS server region. If the DPL request is the first one in the UR, CICS calls RRS to express an interest in the UR, attaches a new mirror transaction, and validates the tokens. If the request is valid, the mirror program links to the specified server application program. The server program completes its work, which is all performed in the UR. This work can include updating recoverable resources in the local server region, or daisy chaining to other CICS regions.
4. When the server program completes, it returns the communications area (COMMAREA) or channels and containers , with return codes, to the client program.

   Note: Steps 3 and 4 can repeated many times for the same UR.
5. When the EXCI client program is ready to commit or back out its changes, the program invokes RRS to begin the 2-phase commit protocol.
6. RRS acts as coordinator and completes one of the following actions:
   - RRS asks the resource managers to prepare to commit all updates in the UR. Resource managers other than the CICS server region might also express an interest in the UR. If all vote yes, RRS tells them to go ahead and commit the changes. If any vote no, RRS tells all the resource managers to back out all the changes made in the UR.

- RRS tells all the resource managers that express an interest in the UR to back out all the changes made in the UR.

The UR is now complete and CICS detaches the mirror task. If the EXCI client sends any new DPL requests after this point, EXCI starts a new UR and CICS attaches a new mirror transaction.

Each DPL request that specifies the SYNCONRETURN option attaches a new mirror task in the target CICS region. The first DPL request that does not specify SYNCONRETURN also attaches a new mirror task, but subsequent requests are directed to the same mirror task. When a sync point takes place, the mirror task ends, and the next non-SYNCONRETURN request attaches a new mirror. See Figure 7 on page 10. In this figure, a z/OS client application issues DPL requests with and without SYNCONRETURN. The numbers on the figure refer to the principal flow, as listed after the figure.

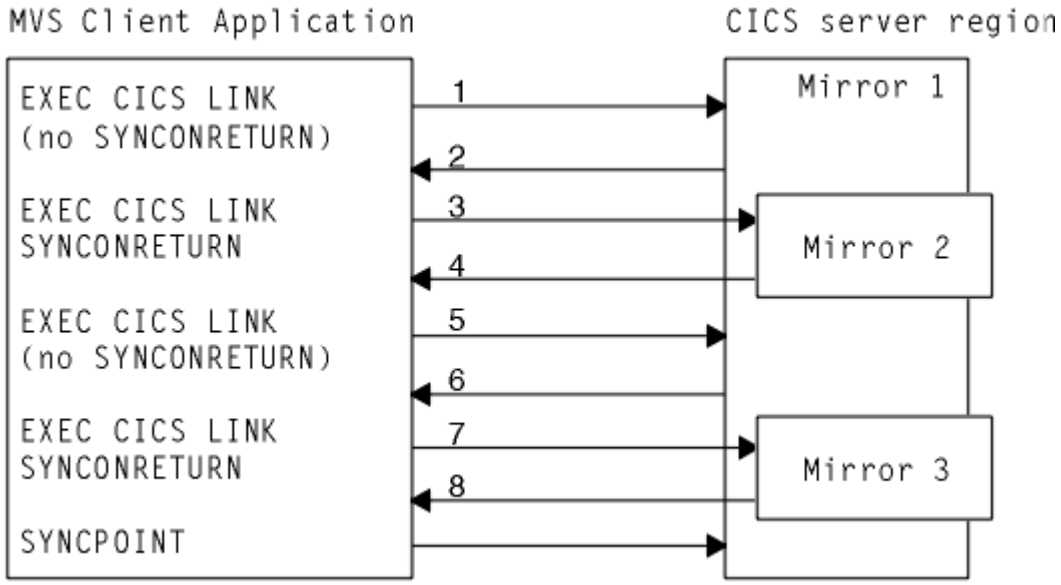


Figure 7. Effect of mixing DPL requests with and without the SYNCONRETURN option

1. Client issues a DPL request without the SYNCONRETURN option.

   Because no mirror transaction is running, a new mirror (Mirror 1) is attached.
2. The DPL request completes, and because it was issued without the SYNCONRETURN option, the mirror transaction waits for another request.
3. Client issues DPL request with the SYNCONRETURN option.

   A new mirror transaction (Mirror 2) is attached.
4. On completion of the DPL request, resources updated by the mirror transaction are committed, and the mirror transaction ends.
5. Client issues another DPL request without the SYNCONRETURN option. Mirror 1 receives and executes the DPL request.
6. The DPL request completes, and once again, the mirror transaction waits for another request.
7. Client issues DPL request with the SYNCONRETURN option.

   A new mirror transaction (Mirror 3) is attached.
8. On completion of the DPL request, resources updated by the mirror transaction are committed, and the mirror transaction ends.
9. The client program requests a syncpoint. Resources updated by mirror 1 are committed, and the transaction ends.

Use of sync points in the client program

A client program can request that a sync point is taken by using a z/OS callable service to commit or back out changes.

To commit changes instigated by the client program, use one of the following z/OS callable services:

Application_Commit_UR (SRRCMIT)
For a description of Application_Commit_UR, see z/OS MVS Programming: Callable Services for High-Level Languages.

Commit_UR (ATRCMIT)
For a description of Commit_UR, see z/OS MVS Programming: Resource Recovery.

To back out changes in the client program, use one of the following z/OS callable services:

Application_Backout_UR (SRRBACK)
For a description of Application_Backout_UR, see z/OS MVS Programming: Callable Services for High-Level Languages.

Backout_UR (ATRBACK)
For a description of Backout_UR, see z/OS MVS Programming: Resource Recovery.

If none of these interfaces are used, changes are committed or backed out explicitly when the client program either ends normally or abends. It is not advisable to use implicit commit or backout for the following reasons:

- The client program cannot tell whether updates were committed or backed out. Even if the program ends normally, a resource manager might back out any changes.
- The runtime environment for high level languages might intercept errors that would otherwise result in an operating system abend. If such an error is intercepted and the client program does not take any explicit action, the program might terminate normally and updates might be committed. Code your client program to ensure that resources are committed or backed out correctly in these situations. For example, a PL/I program can include an ON unit that issues an **SRRBACK** command when errors are encountered. Similarly, a COBOL program can use the ON phrase on statements that might encounter errors.

The EXCI CALL interface

The EXCI CALL interface consists of six commands that you can use for the following actions:

- Allocate and open sessions to a CICS system from non-CICS programs running under z/OS.
- Issue distributed program link (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 as follows:

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

Illustration of the EXCI CALL interface

These four diagrams illustrate the EXCI interface using the EXCI CALL interface.

Stage 1: INITIALIZE_USER (initializing the user environment)
This diagram shows the z/OS Client Application with the External CICS user environment established and the CICS server region with MRO EXCI CONNECTION installed with 3 sessions.

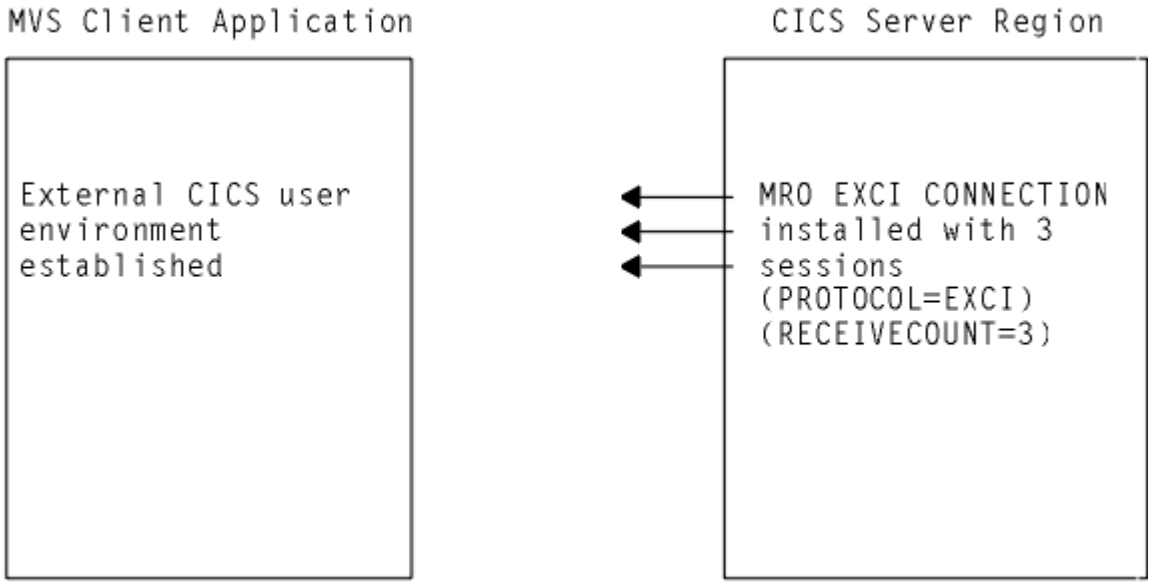


Figure 8. Stage 1: Status after an INITIALIZE_USER call

1. In Figure 8 on page 12, 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.

Stage 2: ALLOCATE_PIPE (allocating a pipe to CICS)

This diagram shows the z/OS Client Application with a pipe allocated and the CICS server region with an MRO EXCI CONNECTION installed with 3 sessions.

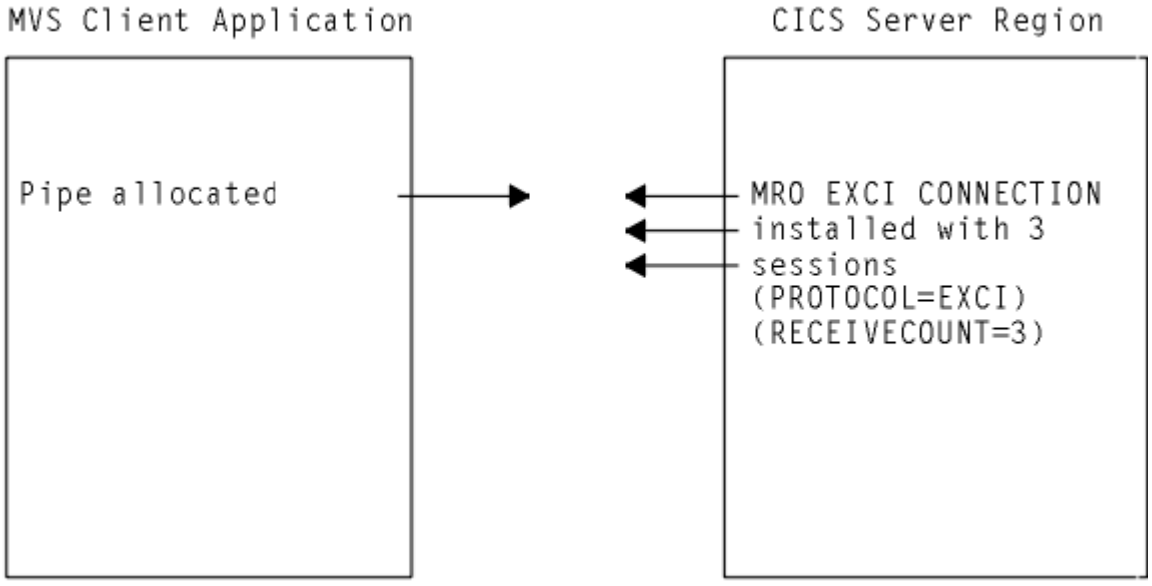


Figure 9. Stage 2: Status after the first ALLOCATE_PIPE call

In Figure 9 on page 12, the external CICS interface logs on to MRO, identifying the target CICS server region.

Stage 3: OPEN_PIPE (connecting an allocated pipe to a receive session)

This diagram shows the z/OS Client Application with a pipe opened and the CICS server region with an MRO EXCI CONNECTION installed with 3 sessions.

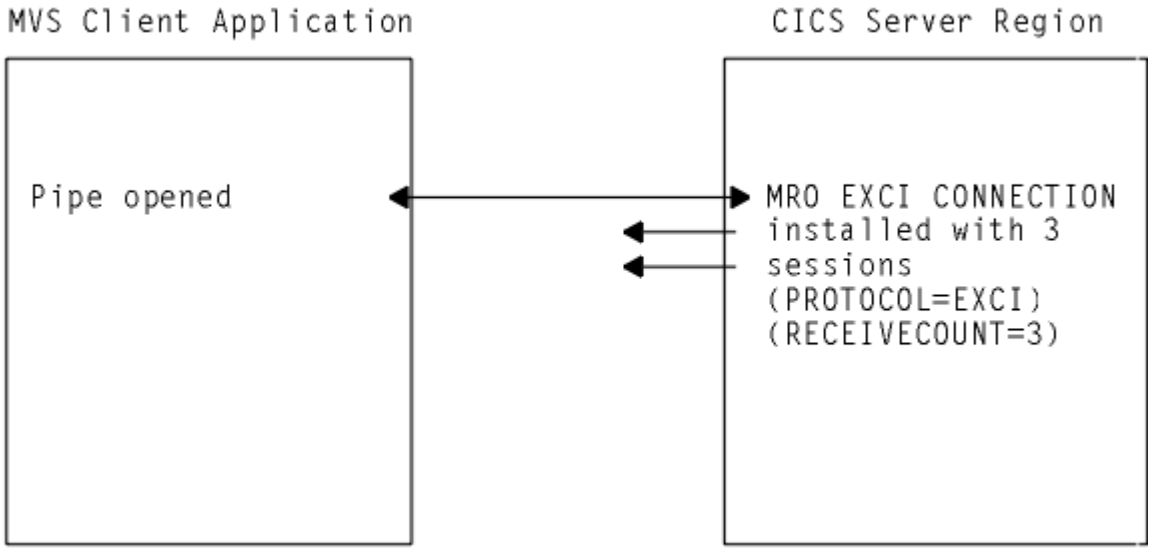


Figure 10. Stage 3: Status after the OPEN_PIPE call

1. In Figure 10 on page 13, 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).

Stage 4: DPL_Request (issuing a DPL request across an open pipe)

This diagram shows that a DPL request and data flows from the client to the server and response and data flows back. The z/OS Client Application has a pipe opened and the CICS server region has an MRO EXCI CONNECTION installed with 3 sessions.

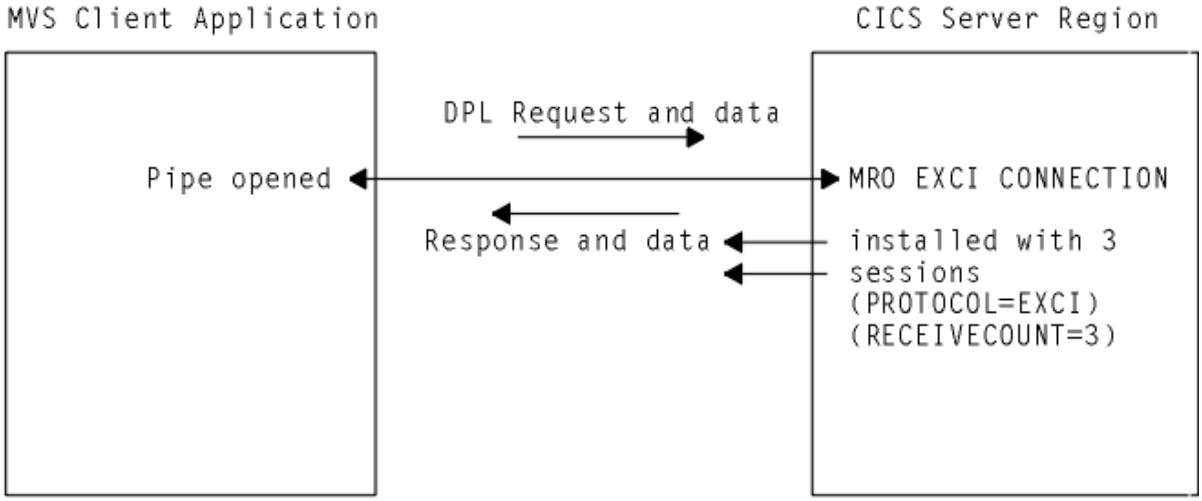


Figure 11. Stage 4: Status with one open pipe, processing a DPL call

In Figure 11 on page 13, 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.

Stage 5: Close_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 9 on page 12. 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).

Stage 6: Deallocate_Pipe (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 8 on page 12.

The application program stub, DFHXCSTB

The EXCI commands invoke the external CICS interface through an application programming stub, called DFHXCSTB, provided by CICS. You must include this stub when you link-edit your non-CICS program.

The EXCI CALL interface commands

In the description of each command that follows, the syntax box illustrates the assembler form of the command. The syntax shows VL,MF=(E,(1)) for each command, indicating the execute form of the CALL macro, with the parameter list storage area addressed by Register 1.

The commands are also supported by C®, COBOL, and PL/I programming languages, using the CALL conventions appropriate for these languages.

There are examples of these CALLs, in all the supported languages, in the sample client programs provided by CICS. For more information, see EXCI sample programs.

Initialize_User

Initialize the user environment. This includes obtaining authority to use IRC facilities. The environment is created for the lifetime of the TCB, so the command needs to be issued only once per user per TCB. Further commands from this user must be issued under the same TCB.

Syntax

```
CALL DFHXCIS,(version_number,return_area,user_token,call_type,
              user_name),VL,MF=(E,(1))
```

Parameters

version_number
A fullword binary input area indicating the version of the external CICS interface parameter list being used. It must be set to 1 in the client program.

The equated value for this parameter in the CICS-supplied copybook DFHXCPL *x* (where *x* indicates the language) is VERSION_1. See “Return area and function call EQUATE copybooks” on page 35 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 34.

user_token
A 1-word output area containing a 32-bit token supplied by the CICS external interface to represent the client program.

The user token corresponds to the *user-name* parameter. The client program must pass this token on all subsequent external CICS interface commands made for the user defined on the *user_name* parameter.

call_type
A 1-word input area indicating the function of the command. It must be set to 1 in the client program to indicate that this is an Initialize_User command.

The equated value for this call in the CICS-supplied copybook DFHXCPL *x* (where *x* indicates the language) is INIT_USER. See “Return area and function call EQUATE copybooks” on page 35 for copybook details.

user_name
An input area holding a name that identifies the user of the external CICS interface. Generally, this is the client program. If this user is to use a specific pipe, then the value in *user_name* must match that of the NETNAME attribute of the CONNECTION definition for the specific pipe.

Responses and reason codes

For all non-zero response codes, a unique reason code value identifies the reason for the response.

Note: All numeric response and reason code values are in decimal.

The following is a summary of the response and reason codes that the external CICS interface can return on the Initialize_User call:

Response OK
Command executed successfully (RC 0). Reason code:

0
Normal response

Response WARNING
The command executed successfully, but with an error (RC 4). Reason codes:

3
VERIFY_BLOCK_FM_ERROR

Initialize_User processing requires storage below 16MB to build the parameter list for the SSI Verify call, and an error has occurred during the FREEMAIN for this area.

4
WS_FREEMAIN_ERROR

An attempt to FREEMAIN working storage has resulted in a z/OS FREEMAIN error.

Response RETRYABLE
The command failed because of setup errors but can be reissued (RC 8). Reason code:

201
NO_CICS_IRC_STARTED

An Initialize_User command has been issued on a z/OS image that has had no IRC activity since the previous IPL, and the external CICS interface cannot determine the CICS SVC number.

Response USER_ERROR
The command failed because of an error in either the client or the server (RC 12). Reason codes:

401
INVALID_CALL_TYPE

An invalid call-type parameter value is specified on this EXCI request.

402
INVALID_VERSION_NUMBER

The version_number parameter does not specify a value of 1 or 2.

403
INVALID_USER_NAME

The user_name parameter consists of all blank characters (X'40').

410
DFHMEBM_LOAD_FAILED

During Initialize_User processing, the external CICS interface attempted to load the main message module in preparation for issuing external CICS interface messages, and the load of this module failed.

411
DFHMET4E_LOAD_FAILED

The load of message module, DFHMET4E, has failed. During Initialize_User processing, the external CICS interface attempted to load its message table in preparation for issuing messages. The load of this module failed.

412
DFHXCURM_LOAD_FAILED

During Initialize_User processing, the external CICS interface attempted to load the user-replaceable module, DFHXCURM. The load of this module failed.

413
DFHXCTRA_LOAD_FAILED

During Initialize_User processing, the external CICS interface attempted to load the trap module (DFHXCTRA). The load of this module has failed.

419
CICS_AFCB_PRESENT

An Initialize_User request has been issued on a TCB that has already been used by CICS . The external CICS interface cannot share a TCB with CICS , ensuring that a CICS application program cannot issue EXCI requests.

420
DFHXCOPT_LOAD_FAILED

During Initialize_User processing, the external CICS interface attempted to load its options module, DFHXCOPT. The load of this module failed.

421
RUNNING_UNDER_AN_IRB

The EXCI call is issued under a z/OS IRB, which is not permitted.

422
SERVER_ABENDED

While processing a DPL request, the CICS server application program abended without handling the error.

423
SURROGATE_CHECK_FAILED

A DPL request has been issued specifying a USERID parameter.

424
RRMS_NOT_SUPPORTED

A DPL request omitting the SYNCONRETURN option has been made on a system that is not running OS/390® Version 2 Release 5 or higher.

425
UOWID_NOT_ALLOWED

A DPL request omitted the SYNCONRETURN option, but specified a value of UOWID. This combination of parameters is not permitted on a DPL request.

426
INVALID_TRANSID2

A DPL request has been issued with a **TRANSID2** parameter that consists of all blanks.

427
INVALID_CCSID

A DPL request has been issued with a **CCSID** parameter that specifies an invalid value.

428
INVALID_ENDIAN

A DPL request has been issued with a endian parameter that specifies an invalid value.

429
DFHXCEIX_LOAD_FAILED

During Initialize_User processing, the external CICS interface attempted to load the module (DFHXCEIX). The load of this module has failed.

430
DFHXCPRX_LOAD_FAILED

During Initialize_User processing, the external CICS interface attempted to load the module (DFHXCPRX). The load of this module has failed.

Response SYSTEM_ERROR
The command failed (RC 16). Reason codes:

601
WS_GETMAIN_ERROR

During Initialize_User processing, a GETMAIN for working storage failed.

602
XCGLOBAL_GETMAIN_ERROR

During Initialize_User processing, a GETMAIN failed for a critical control block (XCGLOBAL).

603
XCUSER_GETMAIN_ERROR

During Initialize_User processing, a GETMAIN request failed for the user control block (XCUSER).

605
VERIFY_BLOCK_GM_ERROR

During Initialize_User processing, a GETMAIN failed for an EXCI internal control block.

606
SSI_VERIFY_FAILED

A VERIFY call to the z/OS subsystem interface (SSI) to obtain the current CICS SVC number failed.

607
CICS_SVC_CALL_FAILURE

During Initialize_User processing, a call to the currently installed CICS SVC failed.

622
ESTAE_SETUP_FAILURE

To protect itself from possible program checks the external CICS interface establishes a z/OS ESTAE. In this case, the z/OS ESTAE macro has failed.

623
ESTAE_INVOKED

A program check is encountered during call processing, and the ESTAE is invoked.

627
INCORRECT_SVC_LEVEL

The release level of the CICS SVC (DFHCSVC) is not the same (or higher) than the release level of the external CICS interface.

For more information about response codes, see “EXCI call response code values” on page 34.

For information about the reason codes, see Response and reason codes returned on EXCI calls.

Allocate_Pipe

Allocate a single session, or pipe, to a CICS region. This command does not connect the client program to a CICS region; this happens on the **Open_Pipe** command.

You can allocate up to 250 pipes in an EXCI address space. The default limit is 100 pipes. However, you can increase this using the **LOGONLIM** parameter when you define CICS as a z/OS subsystem. See EXCI pipe allocation in Installing.

This limit is set to prevent EXCI clients from monopolizing MRO resources, which could prevent CICS systems from using MRO. The limit is applied in both MRO and cross system MRO (XCF/MRO) environments.

An **ALLOCATE_PIPE** request results in an MRO LOGON request being issued and there is a limit on the total number of MRO LOGON requests allowed from all address spaces. This is particularly critical when using XCF/MRO, where the limit on the number of members in a XCF group also limits the total number of MRO LOGON requests.

Syntax

```
CALL DFHXCIS,(version_number,return_area,user_token,call_type,
              pipe_token,CICS applid,allocate_opts),VL,MF=(E,(1))
```

Parameters

version_number
A fullword binary input area indicating the version of the external CICS interface parameter list being used. It must be set to 1 in the client program.

The equated value for this parameter in the CICS-supplied copybook DFHXCPL *x* (where *x* indicates the language) is VERSION_1. See Table 3 on page 35 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 34.

user_token
The 1-word token returned on the **Initialize_User** command.

call_type
A 1-word input area indicating the function of the command. It must be set to 2 in the client program to indicate that this is an **Allocate_Pipe** command.

The equated value for this call in the CICS-supplied copybook DFHXCPL *x* (where *x* indicates the language) is ALLOCATE_PIPE. See Table 3 on page 35 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 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 specify the applid either on this parameter to the **Allocate_Pipe** call, or on the **URMCICS** parameter in the user-replaceable module, DFHXCURM (DFHXCURM is always invoked during **Allocate_Pipe** processing). You can also use the **URMCICS** parameter in DFHXCURM to override an applid specified on the **Allocate_Pipe** call. See The EXCI user-replaceable module for information about the **URMCICS** 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 CALL with null parameters” on page 36.

allocate_opts
A 1-byte input area to represent options specified on this command. The options specify which type of session is to be used—specific or generic. X'00' represents a specific session. X'80' represents a generic session.

The equated value for these options in the CICS-supplied copybook DFHXCPL *x* (where *x* indicates the language) are SPECIFIC_PIPE and GENERIC_PIPE. See Table 3 on page 35 for copybook details.

Responses and reason codes

For all non-zero response codes, a unique reason code value identifies the reason for the response.

Note: All numeric response and reason code values are in decimal.

The following is a summary of the response and reason codes that the external CICS interface can return on the Allocate_Pipe call:

Response OK
Command executed successfully (RC 0). Reason code:

0
Normal response

Response USER_ERROR
The command failed because of an error in either the client or the server (RC 12). Reason codes:

401
INVALID_CALL_TYPE

402
INVALID_VERSION_NUMBER

404
INVALID_USER_TOKEN

421
RUNNING_UNDER_AN_IRB

Response SYSTEM_ERROR
The command failed (RC 16). Reason codes:

604
XCPIPE_GETMAIN_ERROR

608
IRC_LOGON_FAILURE

622
ESTAE_SETUP_FAILURE

623
ESTAE_INVOKED

628
IRP_LEVEL_CHECK_FAILURE

For information about response codes, see “EXCI call response code values” on page 34.

For information about the reason codes, see Response and reason codes returned on EXCI calls.

Open_Pipe

Cause IRC to connect an allocated pipe to a receive session.

Open_Pipe causes IRC to connect an allocated pipe to a receive session of the appropriate connection defined in the CICS region named either on the Allocate_Pipe command, or in DFHXCURM. The appropriate connection is either of the following:

- The EXCI connection with a NETNAME value equal to the **user_name** parameter on the Initialize_User command (that is, you are using a specific connection, dedicated to this client program)
- The EXCI connection defined as generic

In an XCF environment, the Open_Pipe command causes the interregion communication program, DFHIRP, to connect to the LPAR that receives the request. This request is asynchronous, so although the Open_Pipe command can receive a good return code, the subsequent DPL_Request call might fail.

If you shut down CICS without the support of the supplied shutdown-assist transaction (CESD) or an equivalent, and sessions remain open, CICS might not be able to shut its IRC facility in an orderly manner. A normal shutdown of CICS without the support of the shutdown assist transaction waits if any EXCI sessions are not closed. CICS issues message DFHIR2321 indicating the following information:

- The netname of the session if it is on a specific connection
- The word GENERIC if the open sessions are on a generic connection

If you use the supplied shutdown-assist transaction, CESD, sessions that remain open do not present a problem to normal shutdown, because CESD issues an immediate close of IRC. Provided that at least one DPL_Request call has been issued on the session, message DFHIR2321 also shows the job name, step name, and procedure name of the client job that is using the session, and the z/OS ID of the z/OS image on which the client program is running.

Syntax

```
CALL DFHXCIS,(version_number,return_area,user_token,call_type,
              pipe_token),VL,MF=(E,(1))
```

Parameters

version_number
A fullword binary input area indicating the version of the external CICS interface parameter list being used. It must be set to 1 in the client program.

The equated value for this parameter in the CICS-supplied copybook DFHXCPL *x* (where *x* indicates the language) is VERSION_1. See topic Table 3 on page 35 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 34.

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 topic Table 3 on page 35 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 code:

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

When all the EXCI connections of a CICS server region are in use, the CICS server region receives two responses, RESP2=202 (NO_PIPE) and RESP2=203 (NO_CICS) depending upon the following:

1. When using **OPEN_PIPE** request to communicate with a local CICS system (using IRC/MRO), RESP2=202 (NO_PIPE) is raised.
2. When using **OPEN_PIPE** request to communicate with a remote CICS system on a different LPAR (using XCF), RESP2=203 (NO_CICS) is raised.

Response USER_ERROR
The command failed because of an error in either the client or the server (RC 12).

Reason codes:

401
INVALID_CALL_TYPE

402
INVALID_VERSION_NUMBER

404
INVALID_USER_TOKEN

418
INVALID_PIPE_TOKEN

421
RUNNING_UNDER_AN_IRB

Response SYSTEM_ERROR
The command failed (RC 16).

Reason codes:

608
IRC_LOGON_FAILURE

609
IRC_CONNECT_FAILURE

621
PIPE_RECOVERY_FAILURE

622
ESTAE_SETUP_FAILURE

623
ESTAE_INVOKED

For information about the response codes, see "EXCI call response code values" on page 34.

For information about the reason codes, see Response and reason codes returned on EXCI calls.

DPL_Request

Issue a distributed program link request across an open pipe connected to the CICS system on which the server (or target) application program resides.

The command is synchronous, and the TCB waits for a response from CICS. After a pipe is opened, any number of DPL requests can be issued before the pipe is closed. To the server program, the link request appears just like a standard **EXEC CICS LINK** request from another CICS region, and it is not aware that it is sent from a non-CICS client program using EXCI.

The syntax of the call is shown in three forms: the parameters that can be used when *version_number* is set to VERSION_1, the parameters that can be used when *version_number* is set to VERSION_2, and the parameters that can be used when *version_number* is set to VERSION_3.

Syntax

VERSION_1

```
CALL DFHXCIS,(version_number,return_area,user_token,call_type,
              pipe_token,pgmname,COMMAREA,COMMAREA_len,data_len,
              transid,uowid,userid,dpl_retarea,DPL_opts),VL,MF=(E,(1))
```

VERSION_2

```
CALL DFHXCIS,(version_number,return_area,user_token,call_type,
              pipe_token,pgmname,COMMAREA,COMMAREA_len,data_len,
              transid,uowid,userid,dpl_retarea,DPL_opts,
              transid2,ccsid,endian),VL,MF=(E,(1))
```

VERSION_3

```
CALL DFHXCIS,(version_number,return_area,user_token,call_type,
              pipe_token,pgmname,CHANNEL,0,0,
              transid,uowid,userid,dpl_retarea,DPL_opts,
              transid2,0,0),VL,MF=(E,(1))
```

Parameters

version_number
A fullword binary input area that indicates the version of the external CICS interface parameter list being used. This can be set to 1, 2, or 3 in the client program.

The equated value for this parameter in the CICS-supplied copybook DFHXCPL *x* (where *x* indicates the language) is either VERSION_1, VERSION_2, or VERSION_3. See “Return area and function call EQUATE copybooks” on page 35 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 34.

user_token
A 1-word input area that specifies the user token returned to the client program on the **Initialize_User** command.

call_type
A 1-word input area that indicates the function of the command. This must be set to 6 in the client program to indicate that the pipe is now being used for the DPL_Request call.

The equated value for this call in the CICS-supplied copybook DFHXCPL *x* (where *x* indicates the language) is DPL_REQUEST. See “Return area and function call EQUATE copybooks” on page 35 for copybook details.

pipe_token
A 1-word input area that specifies the token returned by EXCI on the **Allocate_Pipe** command. It represents the pipe being used for the **DPL_Request** call.

pgmname
The 8-character name of the CICS application program being called as the server program.

This is either the name as specified on a predefined PROGRAM resource definition installed in the CICS server region, or as it is known to a user-written autoinstall program if the program is to be autoinstalled. The program can be defined in the CICS server region as a local program, or it can be defined as remote. Programs defined as remote enable *daisy-chaining* , where EXCI-CICS DPL calls become EXCI-CICS-CICS DPL calls.

***COMMAREA* (or null_ptr)**
A variable length input area for the communications area (COMMAREA) between the client and server programs. The length is defined by *COMMAREA_len*.

This is the storage area that contains the data to be sent to the CICS application program. This area is also used to receive the updated COMMAREA from the CICS application program (the server program).

This parameter is optional. If it is not required, you must ensure that the CALL parameter list contains a null address for this parameter. How you do this depends on the language you are using for the non-CICS client program. For an example of a call that omits an optional parameter, see "Example of EXCI CALL with null parameters" on page 36.

COMMAREA_len
A fullword binary input area. This parameter specifies the length of the COMMAREA. It is also the length of the server program's COMMAREA (EIBCALEN).

If you specify a COMMAREA, you must also specify this parameter to define the length.

This value should not exceed 24 KB if the COMMAREA is to be passed between any two CICS servers (for any combination of product/version/release). Otherwise, if you are confident that the COMMAREA will not be passed on a further LINK request, you can use a COMMAREA up to 32 KB in length.

If you do not specify a COMMAREA, this parameter is ignored.

data_len
A fullword binary input area. This parameter specifies the length of contiguous storage, from the start of the COMMAREA, to be sent to the server program.

This parameter restricts the amount of data sent to the server program, and should be used to optimize performance if, for example, the COMMAREA is large but the amount of data being passed is small.

On return from the server program, the EXCI data transformer program ensures that the COMMAREA in the non-CICS client program is the same as that of the server program. This caters for the following conditions:

- The data returned is more than the data passed in the original COMMAREA.
- The data returned is less than the data passed in the original COMMAREA.
- There is no data returned because it is unchanged.
- The server is returning null data.

The value of *data_len* must not be greater than the value of *COMMAREA_len* . A value of zero is valid and results in no data being sent to the server program.

If you do not specify a COMMAREA, this parameter is ignored.

CHANNEL
A 16-character input area that contains the name of a channel that is to be made available to the called program. The acceptable characters are A - Z a - z 0 - 9 $ @ # / % & ? ! : | " = ¬ , ; < > . - and _. Leading and embedded blank characters are not permitted. If the name supplied is less than 16 characters, it is padded with trailing blanks up to 16 characters. If the channel does not exist, it is

created. As there is only one LINK level for an EXCI client, this channel remains in scope. For more information about channel scope, see The scope of a channel.

Channel names are always in EBCDIC. The set of allowed characters for channel names, as listed earlier, includes some characters that do not have the same representation in all EBCDIC code pages. Therefore, if channels are to be shipped between regions, it is advisable to restrict the characters that are used to name channels to A-Z a-z 0-9 & : = , ; < > . - and _.

You can specify the channel name DFHTRANSACTION to use a transaction channel. In CICS, a transaction channel does not go out of scope when the link level changes: it is always accessible in the transaction. For more information, see Channels and containers.

The program that issues the **DPL_REQUEST** command can specify the name of a channel on the command. The specified channel might already exist, created by the program using one or more **PUT CONTAINER** commands. Alternatively, the program can specify the name of a channel that does not currently exist, in which case a new empty channel is created.

Note: The two parameters following CHANNEL must be null. In addition, the **ccsid** and **endian** parameters must be null.

***transid* (or null_ptr)**
A 4-character input area that contains the id of the CICS mirror transaction under which the server program is to run. This transaction must be defined to the CICS server region, and its definition must specify:

PROGRAM(DFHMIRS)
The initial program must be the CICS supplied mirror program DFHMIRS.

Failure to specify DFHMIRS as the initial program means that a COMMAREA passed from the client application program is not passed to the CICS server program. Also, the DPL request fails and the client application program receives a response of SYSTEM_ERROR and reason SERVER_PROTOCOL_ERROR.

PROFILE(DFHCICSA)
The DFHCICSA profile specifies the correct value for the INBFMH parameter, which must be specified as INBFMH(ALL) for a mirror transaction.

When the CICS server region receives a DPL request, it attaches the mirror transaction and invokes DFHMIRS. The mirror program then passes control to the requested server program, passing the COMMAREA supplied by the client program. The COMMAREA passed to the server program is primed with the data only, the remainder of the COMMAREA being set to nulls.

The purpose of the transid parameter is to distinguish between different invocations of the server program. This enables you to run different invocations of the server program under transactions that specify different attributes. For example, you can vary the transaction priorities, or the security requirements.

A transid is optional. By default, the CICS server region uses the CICS-supplied mirror transaction, CSMI. If you do not want to specify *transid* , you must ensure that the CALL parameter list contains a null address for this parameter. How you do this depends on the language you are using for the non-CICS client program. For an example of a call that omits an optional parameter, see “Example of EXCI CALL with null parameters” on page 36.

If you issue multiple requests in the same z/OS unit-of-recovery, the same transid must be used in all of them.

***uowid* (or null_ptr)**
An input area that contains a unit-of-work identifier, using the APPC architected format, that is passed on the DPL_Request for correlation purposes.

For DPL requests that are committed when the CICS program returns control to the z/OS application, this parameter is optional.

For DPL requests that are part of an RRMS unit-of-recovery, null_ptr must be specified. The unit-of-work identifier that is already associated with the RRMS unit-of-recovery is used, if there is one; if not,

CICS (or another RRMS resource manager) generates a unique unit-of-work identifier and associates it with the RRMS unit-of-recovery.

If you do not want to specify a *uowid* parameter , you must ensure that the CALL parameter list contains a null address for this parameter. How you do this depends on the language you are using for the non-CICS client program. For an example of a call that omits an optional parameter, see "Example of EXCI CALL with null parameters" on page 36.

The *uowid* parameter is passed to the CICS server region, which uses it as the UOWID for the first unit of work executed by the CICS server program. If the server program issues intermediate sync points before returning to the client program, CICS uses the supplied *uowid* for the subsequent units of work, but with the two-byte sequence number incremented for each new logical unit of work. If the CICS server program updates remote resources, the client-supplied UOWID is distributed to the remote systems that own the resources.

The *uowid* parameter is supplied on the EXCI CALL interface for correlation purposes only, to allow units of work that originated from a particular client program to be identified in CICS . The *uowid* is not provided for recovery purposes between CICS and the client program.

The *uowid* can be a maximum of 27 bytes long and has the following format:

- A 1-byte length field that contains the overall length of the UOWID (excluding this field).
- A 1-byte length field that contains the length of the logical unit name (excluding this field).
- A logical unit name field of variable length up to a maximum of 17 bytes.

  To conform to APPC architecture rules, the LUNAME must be of the form *AAAAAAAA.BBBBBBBB* , where *AAAAAAAA* is optional and:

  – AAAAAAAA and BBBBBBBB are 8-byte names separated by a period
  – If AAAAAAAA is omitted, the period must also be omitted
  – AAAAAAAA and BBBBBBBB must be type-1134 symbol strings (that is, character strings consisting of one or more EBCDIC uppercase letters A - Z and 0 - 9, the first character of which must be an uppercase letter).
- The clock value; the middle 6 bytes of an 8-byte store clock (STCK) value.
- A 2-byte sequence number.

If you omit a unit-of-work identifier (by specifying a null pointer), and the DPL request is not part of an RRMS unit-of-recovery, the external CICS interface generates one for you, consisting of the following:

- A 1-byte length field set to X'1A'.
- A 1-byte LU length field set to X'11'.
- A 17-byte LU name consisting of:

  – An 8-byte eyecatcher set to DFHEXCIU.
  – A 1-byte field that contains a period (.)
  – A 4-byte field that contains the z/OS system identifier (SYSID), in characters, under which the client is running.
  – A 4-byte field that contains the address space id (ASID) in which the z/OS client program is running. The field contains the four character EBCDIC representation of the 2-byte hex address space id.
- The clock value; the middle 6 bytes of an 8-byte store clock (STCK) value
- A 2-byte sequence number set to X'0001'.

***userid* (or null_ptr)**
An 8-character input area that contains the RACF® userid for user security checking in the CICS region. The external CICS interface passes this userid to the CICS server region for user resource and command security checking in the server application program.

A userid is required only if the MRO connection specifies the ATTACHSEC(IDENTIFY) attribute. If the connection specifies ATTACHSEC(LOCAL), the CICS server region applies link security checking. See Intercommunication security for information about link security on MRO connections.

See also EXCI security for information about external CICS interface security considerations.

This parameter is optional. However, if you do not specify a userid, the external CICS interface passes the security userid under which the client program is running. For example, if the client program is running as a z/OS batch job, the external CICS interface obtains and passes the userid specified on the USER parameter of the JOB statement.

If you specify a userid, the userid under which the EXCI job is running is subject to a surrogate user check. This check is performed by the external CICS interface to ensure that the client job userid is authorized to use the userid specified on the DPL call. For more information about surrogate user security checking, see EXCI security.

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 CALL with null parameters" on page 36.

If you issue multiple requests in the same z/OS unit-of-recovery, the same userid must be used in all of them. If the unit-of-recovery also includes EXEC CICS calls, you should allow the userid on all DPL_requests to default to the security userid under which the client program is running.

dpl_retarea

A 12-byte output area into which the DPL_Request processor places responses to the DPL request. Generally, these responses are from CICS , but in some cases the error detection occurs in the external CICS interface, which returns exception conditions that are the equivalent of those returned by an **EXEC CICS LINK** command.

This field is only meaningful in the following circumstances:

- The response field of the EXCI return-area has a zero value
- The EXCI return-area indicates that the server program has abended (response=USER_ERROR and reason=SERVER_ABENDED).

The 12 bytes form three fields, providing the following information:

Field 1 (*fullword value*)

This field is a fullword that contains a RESP value from the DPL_Request call. See "Error codes" on page 40 for the RESP values that can be returned on a DPL_Request call.

If the DPL_Request call reaches CICS , this field contains the EIBRESP value, otherwise it contains an equivalent response set by the external CICS interface. If this field is set by the external CICS interface, RESP is further qualified by a RESP2 value in the second field.

A zero value is the normal response, which equates to EXEC_NORMAL in the return codes copybooks.

Field 2 (*fullword value*)

This field is a fullword that can contain a RESP2 value from the link request, further qualifying the RESP value in field 1.

If the DPL_Request call reaches CICS , the RESP2 field generally is null (CICS does not return RESP2 values across MRO links). However, if the RESP field indicates SYSIDERR (value 53), this field provides a reason code. See "Dpl_retarea return codes" on page 35 for more information.

If the RESP field is set by the external CICS interface, it is further qualified by a RESP2 value in this second field. For example, if the *data_len* parameter specifies a value greater than the *COMMAREA_len* parameter, the external CICS interface returns the RESP value 22 (which equates to EXEC_LENGERR in the return codes copybooks), and a RESP2 value of 13.

See the LINK conditions in LINK for full details of the possible RESP and RESP2 values.

Note: The data transformer program makes special use of the RESP2 field. If any error occurs in the transformer, the error is returned in RESP2.

Field 3 (*fullword value*)
The third field, a 4-character field, contains the following information:

- The abend code if the server program abended
- Four blanks if the server program did not abend.

If a server program abends, it is backed out to its last syncpoint, which can be the start of the task, or an intermediate syncpoint. The server program can issue intermediate syncpoints because SYNCONRETURN is forced.

***DPL_opts* (or null_ptr)**
A 1-byte input area that indicates options to be used on the DPL_Request call.

If you omit this parameter, it defaults to the value X'00'. If you want to omit *DPL_opts* and let it default, ensure that the CALL parameter list contains a null address for this parameter. How you do this depends on the language you are using for the non-CICS program. For an example of a call that omits an optional parameter, see “Example of EXCI CALL with null parameters” on page 36.

Currently, the *DPL_opts* parameter applies only to resource recovery, using the following values:

X'00'
Indicates that you specified NOSYNCONRETURN, because you want the client batch program to control resource recovery, using 2-phase commit protocols supported by z/OS RRS. With this option, the CICS server region does not perform a syncpoint when the server program returns control to CICS . Furthermore, the CICS server application program must not take any explicit syncpoints, otherwise it is abended by CICS . For more information, see “Resource recovery” on page 8.

X'80'
Indicates that SYNCONRETURN is required in the CICS server region.

SYNCONRETURN specifies that the server region is to take a syncpoint on successful completion of the server program, independently of the client program. This option does not prevent a server program from taking its own explicit syncpoints.

The equated value for this parameter in the CICS-supplied copybook DFHXCPL *x* (where *x* indicates the language) is SYNCONRETURN. See “Return area and function call EQUATE copybooks” on page 35 for copybook details.

***transid2* (or null_ptr) Applicable to VERSION_2 or later**
A 4-character input area that contains a CICS transaction id.

The server program runs under a CICS -supplied mirror transaction, CSMI or CPMI. However the transaction id made available to the server program through the EIBTRNID field in the Exec Interface Block is the one specified by the **transid2** parameter. The **transid2** parameter is ignored if the **transid** parameter is specified. The following table gives an example of different combinations of **transid** and **transid2** :

Table 1. Use of transid2

transid	transid2	program executes under	EIBTRNID seen by program
UTRN	omitted	UTRN	UTRN
UTRN	UEIB	UTRN	UTRN
omitted	omitted	CSMI	CSMI
omitted	UEIB	CSMI	UEIB

The **transid2** parameter is useful for server programs that access Db2®, because EIBTRNID is used to determine which DB2ENTRY definition to use. Previously, EIBTRNID could only be set by

using **transid** , which then required you to define a mirror transaction to CICS . Using **transid2** , EIBTRNID is set, but the mirror program executes under the CICS provided definition CSMI.

***ccsid* (or null_ptr) VERSION_2 only**
A fullword binary input area that indicates the Coded Character Set Identifier (CCSID) of the character data contained in the COMMAREA. The ccsid parameter must be specified if character data has to be converted when the COMMAREA is passed to, or returned from, the server program. The parameter can take the following values:

-1 (X'FFFFFFFF')
Indicates that conversion of character data is required and that the source CCSID is defined in the conversion template installed in the server.

1 <= ccsid <= 65535
Indicates that conversion of character data is required and that the value specified overrides the source CCSID defined in the conversion template installed in the server.

***endian* (or null_ptr) VERSION_2 only**
A fullword binary input area that indicates the format, big endian or little endian, for binary data contained in the COMMAREA. Big endian indicates that the leftmost byte contains the most significant digits, as used, for example, in System 390 architecture. Little endian indicates that the rightmost byte contains the most significant digits, as used, for example, in Intel architecture. The endian parameter should be specified if binary data has to be converted when the COMMAREA is passed to, or returned from, the server program. If the ccsid parameter indicates that conversion is required, but endian is not specified (defaults to null), conversion of binary data depends on what is specified in the DFHCNV conversion template installed in the server. The parameter can take the following values:

16909060 (X'01020304')
Binary data is held in big endian format.

67305985 (X'04030201')
Binary data is held in little endian format.

Responses and reason codes

For all non-zero response codes, a unique reason code value identifies the reason for the response.

Note: All numeric response and reason code values are in decimal.

The following is a summary of the response and reason codes that the external CICS interface can return on the DPL call:

Response OK
Command executed successfully (RC 0). Reason code:

0
NORMAL

Response WARNING
The command executed successfully, but with an error (RC 4). Reason codes:

6
IRP_IOAREA_FM_FAILURE

7
SERVER_TERMINATED

Response RETRYABLE
The command failed because of setup errors but can be reissued (RC 8). Reason codes:

203
NO_CICS

204
WRONG_MVS_FOR_RRMS

205
RRMS_NOT_AVAILABLE

Response USER_ERROR
The command failed because of an error in either the client or the server (RC 12). Reason codes:

401
INVALID_CALL_TYPE

402
INVALID_VERSION_NUMBER

404
INVALID_USER_TOKEN

406
PIPE_NOT_OPEN

407
INVALID_USERID

408
INVALID_UOWID

409
INVALID_TRANSID

414
IRP_ABORT_RECEIVED

415
INVALID_CONNECTION_DEFN

416
INVALID_CICS_RELEASE

417
PIPE_MUST_CLOSE

418
INVALID_PIPE_TOKEN

421
RUNNING_UNDER_AN_IRB

422
SERVER_ABENDED

423
SURROGATE_CHECK_FAILED

425
UOWID_NOT_ALLOWED

426
INVALID_TRANSID2

427
INVALID_CCSID

428
INVALID_ENDIAN

431
COMMAREA_LEN_NOT_ALLOWED

432
DATA_LEN_NOT_ALLOWED

433
CCSID_NOT_ALLOWED

434
ENDIAN_NOT_ALLOWED

Response SYSTEM_ERROR
The command failed (RC 16). Reason codes:

612
TRANSFORM_1_ERROR

613
TRANSFORM_4_ERROR

614
IRP_NULL_DATA_RECEIVED

615
IRP_NEGATIVE_RESPONSE

616
IRP_SWITCH_PULL_FAILURE

617
IRP_IOAREA_GM_FAILURE

619
IRP_BAD_IOAREA

620
IRP_PROTOCOL_ERROR

622
ESTAE_SETUP_FAILURE

623
ESTAE_INVOKED

624
SERVER_TIMEDOUT

625
STIMER_SETUP_FAILURE

626
STIMER_CANCEL_FAILURE

629
SERVER_PROTOCOL_ERROR

630
RRMS_ERROR

631
RRMS_SEVERE_ERROR

632
XCGUR_GETMAIN_ERROR

Close_PIPE

Disconnect an open pipe from CICS. The pipe remains in an allocated state, and its tokens remain valid for use by the same user. To reuse a closed pipe, the client program must first reissue an Open_Pipe command using the pipe token returned on the Allocate_Pipe command for the pipe.

Pipes should be closed when not in use because this prevents CICS from shutting down its IRC facility in an orderly manner. Therefore, the Close_Pipe command should be issued as soon as possible after all DPL_Request calls have completed.

Syntax

```
CALL DFHXCIS,(version_number,return_area,user_token,call_type,
              pipe_token),VL,MF=(E,(1))
```

Parameters

version_number
A fullword binary input area indicating the version of the external CICS interface parameter list being used. It must be set to 1 in the client program.

The equated value for this parameter in the CICS-supplied copybook DFHXCPL *x* (where *x* indicates the language) is VERSION_1. See "Deallocate_Pipe" on page 32 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 34.

user_token
The 1-word input area specifying the token, returned to the client program by EXCI on the Initialize_User command, that represents the user of the pipe being closed.

call_type
A 1-word input area indicating the function of the command. This must be set to 4 in the client program to indicate that this is a Close_Pipe command.

The equated value for this call in the CICS-supplied copybook DFHXCPL *x* (where *x* indicates the language) is CLOSE_PIPE. See "Return area and function call EQUATE copybooks" on page 35 for copybook details.

pipe_token
A 1-word input area specifying the token, returned to the client program by EXCI on the original Allocate_Pipe command, that represents the pipe being closed.

Responses and reason codes

For all non-zero response codes, a unique reason code value identifies the reason for the response.

Note: All numeric response and reason code values are in decimal.

The following is a summary of the response and reason codes that the external CICS interface can return on the Close_Pipe call:

Response OK
Command executed successfully (RC 0). Reason code:

0
NORMAL

Response WARNING
The command executed successfully, but with an error (RC 4). Reason codes:

2
PIPE_ALREADY_CLOSED

Response USER_ERROR
The command failed because of an error in either the client or the server (RC 12). Reason codes:

401
INVALID_CALL_TYPE

402
INVALID_VERSION_NUMBER

404
INVALID_USER_TOKEN

418
INVALID_PIPE_TOKEN

421
RUNNING_UNDER_AN_IRB

Response SYSTEM_ERROR
The command failed (RC 16). Reason codes:

610
IRC_DISCONNECT_FAILURE

622
ESTAE_SETUP_FAILURE

623
ESTAE_INVOKED

For information about response codes, see "EXCI call response code values" on page 34.

For information about the reason codes, see Response and reason codes returned on EXCI calls.

Deallocate_Pipe

Deallocate a pipe from CICS. On completion of this command, the pipe can no longer be used, and its associated tokens are invalid. This command should be issued for pipes that are no longer required. This command frees storage associated with the pipe.

Note: After a successful **Open_Pipe** request, when your client program finishes using the pipe, you must first issue a **Close_Pipe** command and then a **Deallocate_Pipe** command to free the pipe. If you issue a **Deallocate_Pipe** command without first closing an open pipe with **Close_Pipe**, your request fails.

Syntax

```
CALL DFHXCIS,(version_number,return_area,user_token,call_type,
              pipe_token),VL,MF=(E,(1))
```

Parameters

version_number
A fullword binary input area indicating the version of the external CICS interface parameter list being used. It must be set to 1 in the client program.

The equated value for this parameter in the CICS-supplied copybook DFHXCPL*x* (where *x* indicates the language) is VERSION_1. See "Return area and function call EQUATE copybooks" on page 35 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 34.

user_token
A 1-word input area containing the token returned on the **Initialize_User** command.

call_type
A 1-word input area indicating the function of the command. This must be set to 5 in the client program to indicate that this is a **Deallocate_Pipe** command.

The equated value for this call in the CICS-supplied copybook DFHXCPL*x* (where *x* indicates the language) is DEALLOCATE_PIPE. See "Return area and function call EQUATE copybooks" on page 35 for copybook details.

pipe_token
A 1-word input area containing the token passed back on the original **Allocate_Pipe** command, that represents the pipe now being deallocated.

Responses and reason codes

For all non-zero response codes, a unique reason code value identifies the reason for the response.

Note: All numeric response and reason code values are in decimal.

The following is a summary of the response and reason codes that the external CICS interface can return on the **Deallocate_Pipe** call:

Response OK
Command executed successfully (RC 0). Reason code:

0
NORMAL

Response WARNING
The command succeeded successfully, but with an error (RC 4). Reason codes:

5
XCPIPE_FREEMAIN_ERROR

6
IRP_IOAREA_FM_FAILURE

Response USER_ERROR
The command failed because of an error in either the client or the server (RC 12). Reason codes:

401
INVALID_CALL_TYPE

402
INVALID_VERSION_NUMBER

404
INVALID_USER_TOKEN

405
PIPE_NOT_CLOSED

418
INVALID_PIPE_TOKEN

421
RUNNING_UNDER_AN_IRB

Response SYSTEM_ERROR
The command failed (RC 16). Reason codes:

611
IRC_LOGOFF_FAILURE

622
ESTAE_SETUP_FAILURE

623
ESTAE_INVOKED

For information about response codes, see “EXCI call response code values” on page 34.

For information about the reason codes, see Response and reason codes returned on EXCI calls.

EXCI call response code values

This table shows the values that can be returned in the response field. All values are in decimal.

Table 2. EXCI response codes (returned in response field of return_area)

Code	Meaning	Explanation
0	OK	For all EXCI CALL commands other than the DPL_request, the call was successful. If an OK response is received for a DPL_request, you must also check *dpl_retarea* to ensure CICS did not return a condition code. If the EIBRESP field of *Dpl_retarea* is zero, the DPL call was successful.
4	WARNING	The external CICS interface detected an error, but this did not stop the CALL command completing successfully. The reason code field describes the error detected.
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 z/OS re-IPL. Each reason code value for a RETRYABLE response documents whether the CALL can be reissued directly, or whether the pipe being used has to be closed and reopened first.
12	USER_ERROR	The EXCI CALL command failed. This class of error means there is an error either in the client program, or in the CICS server program, or in the CICS server region. An example of an error in the CICS server system would be a failed security check, or an abend of the CICS server program, in which case the abend code is set in the abend code field of *dpl_retarea* . Each reason code value for a response of USER_ERROR explains whether the command can be reissued directly, or whether the pipe being used has to be closed and reopened first.
16	SYSTEM_ ERROR	The EXCI CALL command failed. This class of error means that the external CICS interface has detected an error. The reason code value identifies the specific error. If the error can be corrected, then the command can be reissued. Each reason code value for a SYSTEM_ERROR response explains whether the command can be reissued directly, or whether the pipe being used has to be closed and reopened first.

Return area for the EXCI CALL interface

This is the format of the 5-word return area for the EXCI CALL interface.

1. 1–word response field.
2. 1–word reason field.
3. Two 1–word subreason fields—subreason field-1 and subreason field-2.
4. 1–word CICS message pointer field. This is zero if there is no message present. If a message is present, this field contains the address of the storage area containing the message, which is formatted as follows:
   - A 2-byte LL field. LL is the length of the message plus the length of the LLBB field.
   - A 2-byte BB field, set to binary zero.
   - A variable length field containing the text of the message.

Return area and function call EQUATE copybooks

CICS provides four language-specific copybooks that map the storage areas for the *return_area* and *dpl_retarea* parameters of the EXCI CALL commands. The copybooks also provide EQUATE statements for each type of EXCI CALL.

These copybooks, and the libraries they are supplied in for the supported languages, are shown in Table 3 on page 35. All the libraries are shown using the format CICSTS*nn*, where *nn* represents the CICS version.

Table 3. Supplied copybooks of return areas and equated names

Copybook name	Language	Library
DFHXCPLD	Assembler	CICSTS*nn*.CICS.SDFHMAC
DFHXCPLH	C	CICSTS*nn*.CICS.SDFHC370
DFHXCPLO	COBOL	CICSTS*nn*.CICS.SDFHCOB
DFHXCPLL	PL/I	CICSTS*nn*.CICS.SDFHPL1

Return codes

All the possible return codes are contained in a CICS-supplied copybook, which you must include in the program source of your external, non-CICS program.

The names of the copybooks for the supported languages, and the libraries that they are supplied in, are shown in Table 4 on page 35. All the libraries are shown using the format CICSTS*nn*, where *nn* represents the CICS version.

Table 4. Supplied copybooks of RESPONSE and REASON codes

Copybook name	Language	Library
DFHXCRCD	Assembler	CICSTS*nn*.CICS.SDFHMAC
DFHXCRCH	C	CICSTS*nn*.CICS.SDFHC370
DFHXCRCO	COBOL	CICSTS*nn*.CICS.SDFHCOB
DFHXCRCL	PL/I	CICSTS*nn*.CICS.SDFHPL1

z/OS provides copybooks for use with the interfaces described in "Use of sync points in the client program" on page 11. These are described in z/OS MVS Programming: Resource Recovery and z/OS MVS Programming: Callable Services for High-Level Languages.

Dpl_retarea return codes

These are the same as for CICS-to-CICS EXEC CICS DPL commands but with the following additions for the EXCI call interface.

Table 5. Exceptional conditions. RESP and RESP2 values returned to dpl_retarea

Condition	RESP2	Meaning
LENGERR	22	COMMAREA_LEN_TOO_BIG
LENGERR	23	COMMAREA_BUT_NO_COMMAREA_LEN

SYSIDERR also can be returned on an EXCI DPL_Request, if the DPL_Request specifies a program defined in the CICS server region as a remote program, and the link between the server and the remote CICS region is not open. In this situation, SYSIDERR is returned in the first word of the DPL_Retarea (code 53). The reason code qualifying SYSIDERR is placed in the second word of this area (the equivalent of a RESP2 value).For SYSIDERR, the information stored in this field is derived from bytes 1 and 2 of the CICS

EIBRCODE field. For example, if a remote link is not available, the EIBRCODE value stored in bytes 2 and 3 of the DPL_Retarea RESP2 field is X'0800' . For a list of the SYSIDERR reason codes that can be returned in the RESP2 field, see the SYSIDERR section of the notes on EIBRCODE in Attention identifier constants.

TERMERR also may be returned on an EXCI DPL request if the DPL request was to a program defined as remote, and an unrecoverable error occurs during conversation with the mirror on the remote CICS system. For example, suppose client program BATCH1 issues an EXCI DPL request to CICSA for program PROG1, which is defined as remote, and the request is function-shipped to CICSB where the program resides. If the session between CICSA and CICSB fails, or CICSB itself fails while executing the program PROG1, then TERMERR is returned to CICSA, and in turn to BATCH1.

No unique EXCI_DPL_RESP2 values are returned for TERMERR, PGMIDERR, NOTAUTH, and ROLLBACK.

Example of EXCI CALL with null parameters

If you omit an optional parameter, such as *userid* on a DPL_Request, you must ensure that the parameter list is built with a null address for the missing parameter.

The example that follows illustrates how to issue an EXCI DPL_Request with the *userid* and *uowid* parameters omitted in a COBOL program.

DPL CALL without userid and uowid (COBOL): In this example, the DPL parameters used on the call are defined in the WORKING-STORAGE SECTION, as follows:

DPL parameter	COBOL variable	Field definition
version_number	01 VERSION-1	PIC S9(8) COMP VALUE 1.
return_area	01 EXCI-RETURN-CODE.	(structure)
user_token	01 USER-TOKEN	PIC S9(8) COMP VALUE ZERO.
call_type	03 DPL-REQUEST	PIC S9(8) COMP VALUE 6.
pipe_token	01 PIPE-TOKEN	PIC S9(8) COMP VALUE ZERO.
pgmname	01 TARGET-PROGRAM	PIC X(8) VALUE "DFHœAXCS".
commarea	01 COMMAREA.	(structure)
commarea_len	01 COMM-LENGTH	PIC S9(8) COMP VALUE 98.
data_len	01 DATA-LENGTH	PIC S9(8) COMP VALUE 18.
transid	01 TARGET-TRANSID	PIC X(4) VALUE "EXCI".
dpl_retarea	01 EXCI-DPL-RETAREA.	(structure)
dpl_opts	01 SYNCONRETURN	PIC X VALUE X'80'.

The variable used for the null address is defined in the LINKAGE SECTION:

```
LINKAGE SECTION.
        01 NULL-PTR USAGE IS POINTER.
```

Using the data names specified in the WORKING-STORAGE SECTION, 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 CICSTS*xx*.CICS.SDFHSAMP, where *xx* represents the CICS version. For an example of how to omit the same parameters from the DPL call in the other supported languages, see the following sample programs:

DFH$AXCC
The assembler sample

DFH$PXCC
The PL/I sample

DFH$DXCC
The C sample.

The EXCI EXEC CICS interface

The EXCI EXEC CICS interface provides several commands. For example, the interface provides a single, composite command, **EXEC CICS LINK PROGRAM** command, that performs all six commands of the EXCI CALL interface in one invocation. Each time you issue an **EXEC CICS LINK PROGRAM** command in a client application program, the external CICS interface invokes each of the six EXCI calls on your behalf. The **EXEC CICS LINK PROGRAM** command is similar but not identical to the distributed program link command of the CICS command-level application programming interface.

EXCI also provides the ability to process data using channel and container commands. A channel together with its set of containers can then be passed on the **EXEC CICS LINK PROGRAM** command or on a call API **DPL_REQUEST**, as an alternative to using a communications area to transfer data or information from one program to another.

Illustration of the EXCI EXEC CICS interface

This diagram illustrates the EXEC CICS interface, and how it resolves to the six EXCI CALLs.

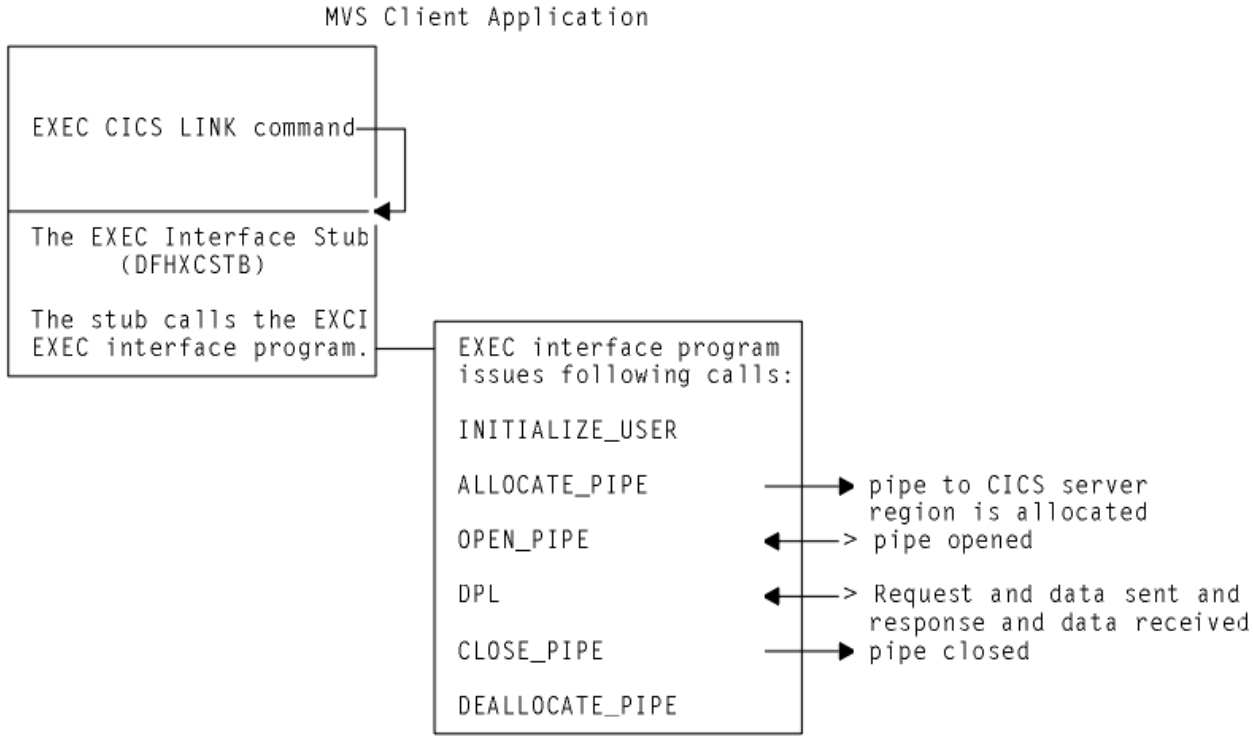


Figure 12. Illustration of the external CICS interface using the EXEC CICS command

1. The z/OS Client Application issues an **EXEC CICS LINK** command.
2. The EXEC interface stub DFHXCSTB calls the EXCI EXEC interface program which issues the following calls:

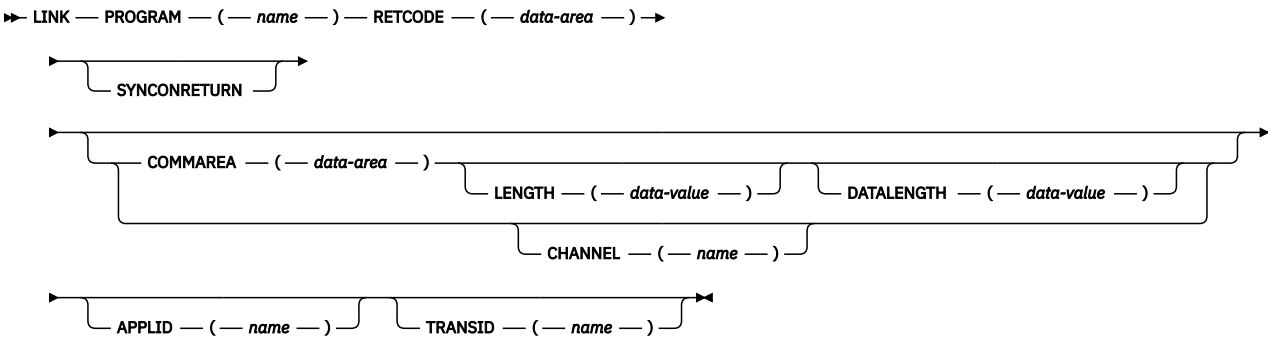
   INITIALIZE_USER
   ALLOCATE_PIPE
   OPEN_PIPE
   DPL
   CLOSE_PIPE
   DEALLOCATE_PIPE

EXEC CICS LINK command (EXCI)

Link from a z/OS client program to the specified server program in a server CICS region.

Format

LINK



Notes:

Error conditions: CCSIDERR , CHANNELERR , CODEPAGEERR , CONTAINERERR , LENGERR, LINKERR, NOTAUTH, PGMIDERR, RESUNAVAIL, ROLLEDBACK, SYSIDERR, TERMERR, WARNING

Comments

With the exception of the **APPLID** and **RETCODE** parameters, the external CICS interface parameters for an **EXEC CICS LINK** command are the same as for a CICS-CICS DPL command.

This information describes only those parameters that you can use with the external CICS interface. For programming information about the **EXEC CICS LINK PROGRAM** command, see LINK.

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 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 **URMCICS** parameter. You can also use the **URMCICS**

parameter in DFHXCURM to override an applid specified on the **LINK** command. See The EXCI user-replaceable module for information about the **URMCICS** parameter.

CHANNEL(***name*** **)**
Specifies the name (1 - 16 characters) of a channel that is to be made available to the called program. The acceptable characters are A - Z a - z 0 - 9 $ @ # / % & ? ! : | " = ¬ , ; < > . - and _. Leading and embedded blank characters are not permitted. If the name supplied is less than 16 characters, it is padded with trailing blanks up to 16 characters. If the channel does not exist, it is created. As there is only one LINK level for an EXCI client, this channel remains in scope. For more information about channel scope, see The scope of a channel.

Channel names are always in EBCDIC. The set of allowed characters for channel names, as listed earlier, includes some characters that do not have the same representation in all EBCDIC code pages. Therefore, if channels are to be shipped between regions, it is advisable to restrict the characters that are used to name channels to A-Z a-z 0-9 & : = , ; < > . - and _.

You can specify the channel name DFHTRANSACTION to use a transaction channel. In CICS, a transaction channel does not go out of scope when the link level changes: it is always accessible in the transaction. For more information, see Channels and containers.

The program that issues the **LINK** command can specify the name of a channel on the command. The specified channel might already exist, created by the program using one or more **PUT CONTAINER** commands. Alternatively, the program can specify the name of a channel that does not currently exist, in which case a new empty channel is created.

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 Passing data to other programs for more information about passing data to CICS application programs.

DATALENGTH(***data-value*** **)**
Specifies a halfword binary value that is the length of a contiguous area of storage from the start of the COMMAREA. If the amount of data in a COMMAREA is small, but the COMMAREA itself is large, specify DATALENGTH to improve performance.

LENGTH(***data-value*** **)**
Specifies a halfword binary value that is the length in bytes of the COMMAREA.

This value should not exceed 24 KB if the COMMAREA is to be passed between any two CICS servers (for any combination of product/version/release), otherwise, if you are confident that the COMMAREA will not be passed on a further LINK request, you can use a COMMAREA up to 32763 in length.

PROGRAM(***name*** **)**
Specifies the program name (1-8 characters) of the CICS server application program to which control is to be passed unconditionally. The specified name must either have been defined as a program to CICS , or the CICS server region must be capable of autoinstalling a definition for the named program.

Note the use of quotation marks:

```
EXEC CICS LINK PROGRAM('PROGX')
```

PROGX is in quotation marks because it is the program name.

```
EXEC CICS LINK PROGRAM(DAREA)
```

DAREA is not in quotes because it is the name of a data area that contains the 8-character program name.

RETCODE(***data-area*** **)**
Specifies a 20-byte area into which the external CICS interface places return code information. This area is formatted into five 1–word fields as follows:

RESP
The primary response code indicating whether the external CICS interface LINK command caused an exception condition during its execution.

RESP2
The secondary response code that further qualifies, where necessary, some of the conditions raised in the RESP parameter.

ABCODE
Contains a valid CICS abend code if the server program abended in the server region.

MSGLEN
Indicates the length of the message (if any) issued by the CICS server region during the execution of the server program. Note that the length is the actual length of the message text only, and does not include this 1—word length field.

MSGPTR
This is the address of the message text returned by the CICS server region.

Note: MSGLEN and MSGPTR are only valid on a LINKERR condition, with the RESP2 value 414.

SYNCONRETURN
Specifies that the CICS server region, named on the APPLID parameter, is to take a syncpoint on successful completion of the server program.

TRANSID(*name*)
Specifies the name of the mirror transaction that the remote region is to attach, and under which it is to run the server program. If you omit the TRANSID option, the CICS server region attaches CSMI.

Note: The TRANSID option specified on the LINK command overrides any TRANSID option specified on the program resource definition installed in the CICS server region.

While you can specify your own name for the mirror transaction initiated by DPL requests, the transaction *must* be defined in the server region, and the transaction definition must specify the mirror program, DFHMIRS. Defining your own transaction to invoke the mirror program gives you the freedom to specify appropriate values for some other options on the transaction resource definition.

See also the important rules about specifying transid with a DPL_Request in DPL_Request. .

Error codes

Most of the exception conditions that are returned on the external CICS interface LINK command are the same as for the CICS-to-CICS distributed program link command. Those that are the same, and their corresponding numeric values are as follows:

LENGERR
22

PGMIDERR
27

SYSIDERR
53

NOTAUTH
70

TERMERR
81

ROLLEDBACK
82

CONTAINERERR
110

RESUNAVAIL
121

CHANNELERR
122

CCSIDERR
123

CODEPAGEERR
125

These exception condition codes are returned in the RESP field.

RESP and RESP2: References to the RESP and RESP2 fields in this section are to the first two fields of the RETCODE parameter.

The exception conditions that are specific to the external CICS interface are as follows:

- The RESP2 values on the error condition LENGERR is specific to the external CICS interface.
- The exception conditions WARNING and LINKERR are specific to the external CICS interface.

The WARNING and LINKERR exceptions are a result of responses to individual EXCI calls issued by the external CICS interface in response to an **EXEC CICS LINK** command. These WARNING and LINKERR exceptions correspond to EXCI call responses as indicated in the descriptions.

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 Response and reason codes returned on EXCI calls 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 The EXCI user-replaceable module for details of DFHXCURM.

Exception conditions and RESP2 values specific to EXEC CICS LINK for EXCI

Table 6 on page 42 lists all the exception conditions and RESP2 values that are specific to the **EXEC CICS LINK** command for the external CICS interface.

Table 6. Exception conditions. RESP and RESP2 values returned from the EXEC API.

Condition (RESP)	RESP2	Meaning
LENGERR (22)	22	COMMAREA length greater than 32763 bytes specified
	23	COMMAREA specified but no LENGTH parameter specified
WARNING (4)	401	Invalid *call_type* parameter value specified on Close_Pipe or Deallocate_Pipe call
	402	Invalid *version_number* parameter specified on Close_Pipe or Deallocate_Pipe call
	404	Invalid *user_token* specified on Close_Pipe or Deallocate_Pipe call
	405	A Deallocate_Pipe call has been issued against a pipe that is not yet closed
	418	An invalid pipe token has been issued on a Close_Pipe or Deallocate_Pipe call
	421	A Close_Pipe or Deallocate_Pipe command has been issued under an IRB
	610	There has been a CICS IRP logoff failure on a Deallocate_Pipe call
	611	There has been a CICS IRC disconnect failure on a Close_Pipe call
	622	There has been a z/OS ESTAE setup failure on a Close_Pipe or Deallocate_Pipe call
	623	A program check on a Close_Pipe or Deallocate_Pipe call has caused the ESTAE to be invoked
LINKERR (88)	201	Command has been issued on a z/OS image which has had no IRC activity since the previous IPL
	202	There are no available sessions
	203	CICS has not yet been brought up, or (2) has not yet opened IRC, or (3) no generic connection is installed, or (4) no specific connection is installed with the required netname.
	204	An **EXEC CICS LINK** command without the SYNCONRETURN option has been issued specifying a CICS system on a different z/OS image.
	205	An **EXEC CICS LINK** command without the SYNCONRETURN option has been issued when RRS is not available
	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
	408	Error in UOWID parameter
LINKERR (88)	409	Transid consisting of all blanks or zero has been passed
	410	Load of message module, DFHMEBMX, failed
	411	Load of message module, DFHMET4E, failed
	412	Load of DFHXCURM failed

Table 6. Exception conditions. RESP and RESP2 values returned from the EXEC API. *(continued)*

Condition (RESP)	RESP2	Meaning
	413	Load of DFHXCTRA failed
	414	If run as a CICS-to-CICS linked-to program, this server program would have resulted in an error with an appropriate message sent to the terminal. Running the program as an EXCI server program returns the message addressed by the MSGPTR field of the RETCODE area.
	415	Target connection is an MRO connection, not an EXCI connection.
	416	Command has been issued against a CICS region running under a release of CICS earlier than CICS for MVS/ESA 4.1.
	417	Command has been issued against a pipe in the MUST CLOSE state. Further EXCI **EXEC CICS LINK** commands will have unpredictable results and are, therefore, not permitted.
	418	Pipe_token does not address an XCPIPE control block, or there is a mismatch between user_token and pipe_token.
	419	CICS runs, or did run, under the TCB that this command is attempting to use. This is not permitted and the command fail.
	420	Load of DFHXCOPT failed.
	421	The command has been issued under a z/OS IRB, which is not permitted.
	422	The server has bended.
	423	Surrogate user check failed
	424	An **EXEC CICS LINK** command without the SYNCONRETURN option has been issued on a system that does not support RRMS
	425	A DPL request omitted the SYNCONRETURN option, but specified a value of UOWID.
	601	A GETMAIN of working storage failed. This error leads to user abend 408
	602	A GETMAIN failed. This error leads to user abend 403.
	603	A GETMAIN failed. This error leads to user abend 410
	604	A GETMAIN failed.
	605	A GETMAIN for the VERIFY block failed. This error leads to user abend 409.
	606	An SSI verify request (to obtain CICS SVC instruction) failed. This error leads to user abend 405.
	607	An SVC call failed. This error leads to user abend 406.
	608	Logon to IRP failed
	609	Connect to IRP failed
	610	Disconnect from IRP failed
	611	Logoff from IRP failed
	612	Invalid data input to transformer_1
	613	Invalid data input to transformer_4
LINKERR (88)	614	CICS has responded but has not sent any data.
	615	CICS cannot satisfy the request.

Table 6. Exception conditions. RESP and RESP2 values returned from the EXEC API. *(continued)*

Condition (RESP)	RESP2	Meaning
	616	IRP_SWITCH_PULL request (to read data sent from CICS into a larger input/output area) has failed.
	617	A GETMAIN for a larger input/output area failed
	619	IRP has had a problem with the input/output area passed from the client program
	620	IRP has disconnected from EXCI
	621	A DISCONNECT command is issued in an error situation following an IRP CONNECT. The DISCONNECT has failed, indicating a serious error.
	622	XCPRH ESTAE setup command failed This error leads to user abend 402.
	623	XCPRH ESTAE invoked due to program check during the processing of this command. ESTAE attempts backout and takes a SYSMDUMP. Further requests are permitted although the pipe is now in a MUST CLOSE state.
	624	The DPL request has been passed to CICS but the time specified in DFHXCOPT has been exceeded. The request is canceled.
	625	A z/OS STIMERM macro call failed.
	626	A z/OS STIMERM CANCEL request failed.
	627	The CICS SVC is at the incorrect level. This error leads to user abend 407.
	628	DFHIRP is at the incorrect level.
	629	A response to a DPL request has been returned by CICS but the external CICS interface does not understand the response.
	630	An unexpected return code was received from RRMS when processing an **EXEC CICS LINK command** without the SYNCONRETURN option.
	631	An unexpected error was encountered when processing an **EXEC CICS LINK** command without the SYNCONRETURN option.
	632	A GETMAIN for DFHXCGUR's working storage failed while processing an **EXEC CICS LINK** command without the SYNCONRETURN option.
	633	An INQUIRE_CHANNEL request to obtain the channel token failed due to an external CICS interface error.
	903	An XCEIP ESTAE setup command failed.
	904	XCEIP ESTAE invoked.

See Return codes 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-supplied translators.

For information about compiling and link-editing external CICS interface client programs, see Compiling and link-editing EXCI client programs.

EXEC CICS DELETE CHANNEL command (EXCI)

Delete a named channel and all the containers that are in it.

DELETE CHANNEL (EXCI)

Conditions: CHANNELERR

Description

DELETE CHANNEL (EXCI) deletes the specified channel and all the containers that are in it. When you delete a channel and its containers:

- Any data that is in the containers is discarded.
- All storage that relates to the channel and to its containers is released.

The application program that issues the **DELETE CHANNEL** command must be the program that owns the channel. The program that owns the channel is the program that created the channel by naming it on one of the following commands:

- **LINK PROGRAM CHANNEL**
- **MOVE CONTAINER CHANNEL TOCHANNEL**
- **PUT CONTAINER CHANNEL**

An application program cannot delete the following channels:

- Any channel that the application program did not create
- Any channel that is read-only
- The transaction channel DFHTRANSACTION

Options

CHANNEL(data-value)
Specifies the 1–16 character name of the channel that is to be deleted. Every container that is owned by the channel is deleted, and the channel itself is deleted.

RETCODE(data-area)
Specifies a 20-byte area into which the external CICS interface places return information. This area is formatted into five 1–word fields. This command will return information in the first two words. These words will contain the RESP and RESP2 codes. The other three words will be set to nulls.

RESP
The primary response code, which indicates whether the external CICS interface **DELETE CHANNEL** command caused an exception condition during its execution.

RESP2
The secondary response code that further qualifies, where necessary, some of the conditions that are raised in the *RESP* parameter.

Conditions

122 CHANNELERR
RESP2 values:

2
The channel specified on the **CHANNEL** option could not be found.

3
The channel specified on the **CHANNEL** option is a read-only channel.

5
The channel specified on the **CHANNEL** option is the transaction channel.

6
The channel specified on the **CHANNEL** option is not owned by the calling program.

904
XCEIP ESTAE invoked.

EXEC CICS DELETE CONTAINER command (EXCI)

Delete a named channel container.

DELETE CONTAINER (EXCI)

Conditions: CHANNELERR, CONTAINERERR, INVREQ

Description

DELETE CONTAINER (EXCI) deletes a container from a channel and discards any data that it contains.

The container is identified by name and by the channel for which it is a container - the channel that "owns" it.

Options

CHANNEL(data-value)
Specifies the name (1–16 characters) of the channel that owns the container. You can specify the channel name DFHTRANSACTION to use the transaction channel.

CONTAINER(data-value)
Specifies the name (1–16 characters) of the container to be deleted.

RETCODE(data-area)
Specifies a 20-byte area into which the external CICS interface places return information. This area is formatted into five 1–word fields. This command will return information in the first two words. These words will contain the RESP and RESP2 codes. The other three words will be set to nulls.

RESP
The primary response code, which indicates whether the external CICS interface **DELETE CONTAINER** command caused an exception condition during its execution.

RESP2
The secondary response code that further qualifies, where necessary, some of the conditions that are raised in the *RESP* parameter.

Conditions

122 CHANNELERR
RESP2 values:

2
The channel specified on the **CHANNEL** option could not be found.

3
The channel specified on the **CHANNEL** option is read-only.

904
XCEIP ESTAE invoked.

110 CONTAINERERR
RESP2 values:

10
The container named on the **CONTAINER** option could not be found.

16 INVREQ
RESP2 values:

4
The command was issued outside the scope of a currently-active channel.

30
You cannot delete a CICS-defined read-only container.

EXEC CICS ENDBROWSE CONTAINER command (EXCI)

End a browse of the containers that are associated with a channel.

ENDBROWSE CONTAINER (EXCI)

Conditions: TOKENERR

Description

ENDBROWSE CONTAINER (EXCI) ends a browse of the containers that are associated with a channel, and invalidates the browse token.

Options

BROWSETOKEN(data-value)
Specifies, as a fullword binary value, the browse token to be deleted.

RETCODE(data-area)
Specifies a 20-byte area into which the external CICS interface places return information. This area is formatted into five 1–word fields. This command will return information in the first two words. These words will contain the RESP and RESP2 codes. The other three words will be set to nulls.

RESP
The primary response code, which indicates whether the external CICS interface **ENDBROWSE CONTAINER** command caused an exception condition during its execution.

RESP2
The secondary response code that further qualifies, where necessary, some of the conditions that are raised in the *RESP* parameter.

Conditions

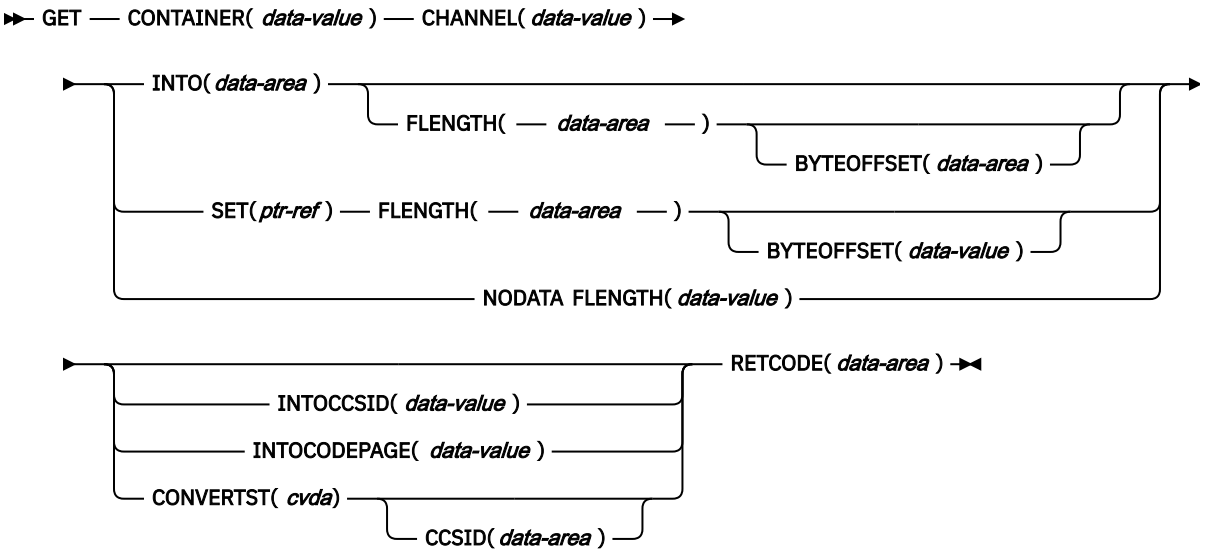
112 TOKENERR
RESP2 values:

3
The browse token is not valid.

EXEC CICS GET CONTAINER command (EXCI)

Retrieve data from a named channel container.

GET CONTAINER (EXCI)



Conditions: CCSIDERR, CHANNELERR, CODEPAGEERR, CONTAINERERR, INVREQ, LENGERR

Description

GET CONTAINER (EXCI) reads the data associated with a specified channel container.

The container that holds the data is identified by name and by the channel for which it is a container; the channel that "owns" it.

Options

BYTEOFFSET(*data-value*)
Specifies the offset in bytes where the data returned starts. For CHAR containers, the **BYTEOFFSET** value is used as an offset into the data in the requested code page. If you use a code page with multibyte characters, depending on the **BYTEOFFSET** value you specify, the data returned might have partial characters at the beginning, end, or both. In this situation, your application program must be able to handle and interpret the data returned. If the value specified is less than zero, zero is assumed.

CCSID(*data-area*)
Returns a fullword that contains the Coded Character Set Identifier (CCSID) of the data returned by the **CONVERTST(NOCONVERT)** option. You can use this option to retrieve containers with a DATATYPE of CHAR, without converting the data. If a DATATYPE of BIT is specified for the container, this value is zero.

CHANNEL(*data-value*)
Specifies the name (1 - 16 characters) of the channel that owns the container. You can specify the channel name DFHTRANSACTION to use the transaction channel.

CONTAINER(*data-value*)
Specifies the name (1 - 16 characters) of the container that holds the data to be retrieved.

CONVERTST(*cvda*)
Specifies the required data conversion status.

NOCONVERT
The container data is retrieved without being converted.

FLENGTH(*data-area*)
As an input field, **FLENGTH** specifies, as a fullword binary value, the length of the data to be read. As an output field, **FLENGTH** returns the length of the data in the container. **FLENGTH** is an input or an output field depending on which of the **BYTEOFFSET** , **INTO** , **SET** , or **NODATA** options you specify.

BYTEOFFSET option specified
FLENGTH is both an input and an output field.

On **input** , **FLENGTH** specifies the maximum length of the data that the program accepts. The data returned begins at the offset specified by the **BYTEOFFSET** value. If the value specified is less than zero, zero is assumed.

On **output** (that is, on completion of the retrieval operation) CICS sets the data area to the length of the data returned. The maximum length returned is equal to the length of the data in the container minus the **BYTEOFFSET** value.

INTO option specified
FLENGTH is both an input and an output field.

On **input** , **FLENGTH** specifies the maximum length of the data that the program accepts. If the value specified is less than zero, zero is assumed. If the length of the data exceeds the value specified, the data is truncated to that value and the LENGERR condition occurs. If the length of the data is less than the specified value, the data is copied but no padding is performed.

You do not need to specify **FLENGTH** if the length can be generated by the compiler from the **INTO** variable. If you specify both **INTO** and **FLENGTH** , **FLENGTH** specifies the maximum length of the data that the program accepts.

On **output** (that is, on completion of the retrieval operation) CICS sets the data area, if specified, to the actual length of the data in the container. If the container holds character data that has been converted from one CCSID to another, this is the length of the data after conversion.

SET or NODATA option specified
FLENGTH is an output-only field. It must be present and must be specified as a data-area.

On completion of the retrieval operation, the data area is set to the actual length of the data in the container. If the container holds character data that has been converted from one CCSID to another, this is the length of the data after conversion.

INTO(*data-area*)
Specifies the data area into which the retrieved data is placed.

INTOCCSID(*data-value*)
Specifies the Coded Character Set Identifier (CCSID) into which the character data in the container is converted, as a fullword binary number. If you prefer to specify an IANA name for the code page, or if you prefer to specify the CCSID as alphanumeric characters, use the **INTOCODEPAGE** option instead.

For CICS Transaction Server for z/OS applications, the CCSID is typically an EBCDIC CCSID. However, it is possible to specify an ASCII CCSID if, for example, you want to retrieve ASCII data without it being automatically converted to EBCDIC.

If **INTOCCSID** and **INTOCODEPAGE** are not specified, the value for conversion defaults to the CCSID of the EXCI job. The default CCSID of the EXCI job is specified on the **LOCALCCSID** parameter of DFHXCOPT.

Only character data can be converted, and only then if a DATATYPE of CHAR was specified on the **PUT CONTAINER** command used to place the data in the container. A DATATYPE of CHAR is implied if **FROMCCSID** or **FROMCODEPAGE** is specified on the **PUT CONTAINER** command.

For more information about data conversion with channels, see Data conversion with channels.

For an explanation of CCSIDs, see Preparing for code page conversion with channels.

INTOCODEPAGE(*data-value*)
Specifies an IANA-registered alphanumeric charset name or a Coded Character Set Identifier (CCSID) for the code page into which the character data in the container is to be converted, using up to 40

alphanumeric characters, including appropriate punctuation. Use this option instead of the CCSID option if you prefer to use an IANA-registered charset name, as specified in the Content-Type header for an HTTP request. CICS converts the IANA name into a CCSID, and the subsequent data conversion process is identical. Also use this option if you prefer to specify the CCSID in alphanumeric characters, rather than as a fullword binary number.

Where an IANA name exists for a code page and CICS supports its use, the name is listed with the CCSID. For more information, see Preparing for code page conversion with channels.

NODATA
Specifies that no data is retrieved. Use this option to discover the length of the data in the container (returned in **FLENGTH**).

The length of character data may change if data conversion takes place. Therefore, if character data is to be converted into any CCSID *other than that of this region* , when you specify **NODATA** , you should also specify **INTOCCSID** . This ensures that the correct length of the converted data is returned in **FLENGTH**.

RETCODE(data-area)
Specifies a 20-byte area into which the external CICS interface places return information. This area is formatted into five 1–word fields. This command will return information in the first two words. These words will contain the RESP and RESP2 codes. The other three words will be set to nulls.

RESP
The primary response code, which indicates whether the external CICS interface **GET CONTAINER** command caused an exception condition during its execution.

RESP2
The secondary response code that further qualifies, where necessary, some of the conditions that are raised in the *RESP* parameter.

SET(*ptr-ref*)
Specifies a data area in which the address of the retrieved data is returned.

The external CICS interface maintains the data area until any of the following occurs:

- A subsequent **GET CONTAINER** command with the **SET** option, for the same container in the same channel, is issued by any program that can access this storage.
- The container is deleted by a **DELETE CONTAINER** command.
- The container is moved by a **MOVE CONTAINER** command.
- The channel and the containers that are in it are deleted by a **DELETE CHANNEL** command.

Beware of linking to other programs that might issue one of these commands.

Do not issue a **FREEMAIN** command to release this storage.

If your application needs to keep the data, the application should move the data into its own storage.

Conditions

123 CCSIDERR
RESP2 values:

1
The CCSID specified on the **INTOCCSID** option is outside the range of valid CCSID values.

2
The CCSID specified on the **INTOCCSID** option and the CCSID of the container are an unsupported combination. (The CCSID of the container is the value that was specified using either **FROMCODEPAGE** or **FROMCCSID** , or defaulted, when the container was built.)

3
The data was created with a data type of BIT. Code page conversion is not possible. The data was returned without any code page conversion.

4
One or more characters could not be converted. The character has been replaced by a blank in the converted data.

5
There was an internal error in the code page conversion of a container.

122 CHANNELERR
RESP2 values:

2
The channel specified on the **CHANNEL** option could not be found.

904
XCEIP ESTAE invoked.

125 CODEPAGEERR
RESP2 values:

1
The code page specified on the **INTOCODEPAGE** option is not supported.

2
The code page specified on the **INTOCODEPAGE** option and the code page of the channel are an unsupported combination.

3
The data was created with a data type of BIT. Code page conversion is not possible. The data was returned without any code page conversion.

4
One or more characters could not be converted. The character has been replaced by a blank in the converted data.

5
There was an internal error in the code page conversion of a container.

110 CONTAINERERR
RESP2 values:

10
The container named on the **CONTAINER** option could not be found.

16 INVREQ
RESP2 values:

2
The **INTOCCSID** option was specified without the **CHANNEL** option, and there is no current channel (because the program that issued the command was not passed one.) **INTOCCSID** is valid only on **GET CONTAINER** commands that specify a channel.

4
The **CHANNEL** option was not specified, there is no current channel (because the program that issued the command was not passed one), and the command was issued outside the scope of a currently-active BTS activity.

5
The CONVERTST cvda value is invalid.

22 LENGERR
RESP2 values:

11
The length of the program area is shorter than the length of the data in the container. When the area is smaller, the data is truncated to fit into it.

12
The offset is greater than, or equal to, the length of the container.

EXEC CICS GETNEXT CONTAINER command (EXCI)

Browse the containers that are associated with a channel.

GETNEXT CONTAINER (EXCI)

Conditions: END, TOKENERR

Description

GETNEXT CONTAINER (EXCI) returns the name of the next container that is associated with a channel.

Note:

1. You can use successive **GETNEXT CONTAINER (EXCI)** commands to retrieve the names of all the channel's containers that existed when the **STARTBROWSE CONTAINER (EXCI)** command was run. However, the names of any containers that are deleted after the **STARTBROWSE** command and before they were returned by a **GETNEXT** command are not returned.
2. The names of any containers that are created on (or moved to) this channel or activity after the **STARTBROWSE** command is run might, or might not, be returned.
3. The order in which containers are returned is undefined and might change. As best practice, applications should not rely on the order of returned containers.

Options

BROWSETOKEN(data-value)
Specifies, as a fullword binary value, a browse token returned on a previous **STARTBROWSE CONTAINER (EXCI)** command.

CONTAINER(data-area)
Returns the 16-character name of the next data-container.

RETCODE(data-area)
Specifies a 20-byte area into which the external CICS interface places return information. This area is formatted into five 1–word fields. This command will return information in the first two words. These words will contain the RESP and RESP2 codes. The other three words will be set to nulls.

RESP
The primary response code, which indicates whether the external CICS interface **GETNEXT CONTAINER** command caused an exception condition during its execution.

RESP2
The secondary response code that further qualifies, where necessary, some of the conditions that are raised in the *RESP* parameter.

Conditions

83 END
RESP2 values:

2
There are no more containers.

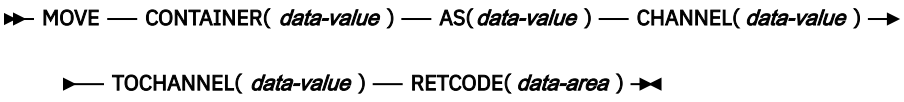
112 TOKENERR
RESP2 values:

3
The browse token is not valid.

EXEC CICS MOVE CONTAINER command (EXCI)

Move a container (and its contents) from one channel to another.

MOVE CONTAINER (EXCI)



Conditions: CHANNELERR, CONTAINERERR, INVREQ

Description

MOVE CONTAINER (EXCI) moves a container from one channel to another. After the move, the source container no longer exists.

The source and target containers are identified by name and by the channels that own them. The channel that owns the source container is identified by the **CHANNEL** option. The channel that owns the target container is identified by the **TOCHANNEL** option.

You can move a container:

- From one channel to another
- Within the same channel (This has the effect of renaming the container.)

You can use **MOVE CONTAINER** , instead of **GET CONTAINER** and **PUT CONTAINER** , as a more efficient way of transferring data between channels.

Note:

1. The source channel must be within the scope of the program that issues the **MOVE CONTAINER** command.
2. If the target channel does not exist, within the scope of the program that issues the **MOVE CONTAINER** command, it is created.
3. If the source container does not exist, an error occurs.
4. If the target container does not already exist, it is created. If the target container already exists, its previous contents are overwritten.
5. If you try to overwrite a container with itself, nothing happens. That is, if you specify the same value for the **CONTAINER** and **AS** options and specify the same value for the **CHANNEL** and **TOCHANNEL** options, so that the same channel is specified, the source container is not changed nor deleted. No error condition is raised.

Options

AS(data-value)
Specifies the name (1–16 characters) of the target container. If the target container already exists, its contents are overwritten.

The acceptable characters are A-Z a-z 0-9 $ @ # / % & ? ! : | " = ¬ , ; < > . - and _. Leading and embedded blank characters are not permitted. If the name supplied is less than 16 characters, it is padded with trailing blanks up to 16 characters.

Container names are always in EBCDIC. The allowable set of characters for container names, listed above, includes some characters that do not have the same representation in all EBCDIC code pages. Therefore, it is recommended that, if containers are to be shipped between regions, the characters used in naming them should be restricted to A-Z a-z 0-9 & : = , ; < > . - and _.

CHANNEL(data-value)
Specifies the name (1–16 characters) of the channel that owns the source container. You can specify the channel name DFHTRANSACTION to use the transaction channel.

CONTAINER(data-value)
Specifies the name (1–16 characters) of the source container that is to be moved.

RETCODE(data-area)
Specifies a 20-byte area into which the external CICS interface places return information. This area is formatted into five 1–word fields. This command will return information in the first two words. These words will contain the RESP and RESP2 codes. The other three words will be set to nulls.

RESP
The primary response code, which indicates whether the external CICS interface **MOVE CONTAINER** command caused an exception condition during its execution.

RESP2
The secondary response code that further qualifies, where necessary, some of the conditions that are raised in the *RESP* parameter.

TOCHANNEL(data-value)
Specifies the name (1–16 characters) of the channel that owns the target container. If you are specifying a new channel, remember that the acceptable characters are A-Z a-z 0-9 $ @ # / % & ? ! : | " = ¬ , ; < > . - and _. Leading and embedded blank characters are not permitted. If the name supplied is less than 16 characters, it is padded with trailing blanks up to 16 characters. If the channel does not exist, it is created. This new channel remains in scope until the link level changes. For more information about channel scope, see The scope of a channel.

Channel names are always in EBCDIC. The allowable set of characters for channel names, listed above, includes some characters that do not have the same representation in all EBCDIC code pages. Therefore, if channels are to be shipped between regions, it is advisable to restrict the characters that are used to name them to A-Z a-z 0-9 & : = , ; < > . - and _.

You can specify the channel name DFHTRANSACTION to use a transaction channel. A transaction channel does not go out of scope when the link level changes: it is always accessible in the task. For more information, see Channels and containers.

Conditions

122 CHANNELERR
RESP2 values:

1
The name specified on the **TOCHANNEL** option contains an illegal character or combination of characters.

2
The channel specified on the **CHANNEL** option could not be found.

3
The channel specified on the **CHANNEL** option is read-only.

904
XCEIP ESTAE invoked.

110 CONTAINERERR
RESP2 values:

10
The container named on the **CONTAINER** option could not be found.

18
The name specified on the **AS** option contains an illegal character or combination of characters.

16 INVREQ
RESP2 values:

30
You cannot move a CICS-defined read-only container.

31
You cannot move a container to (that is, overwrite) an existing, CICS-defined, read-only container.

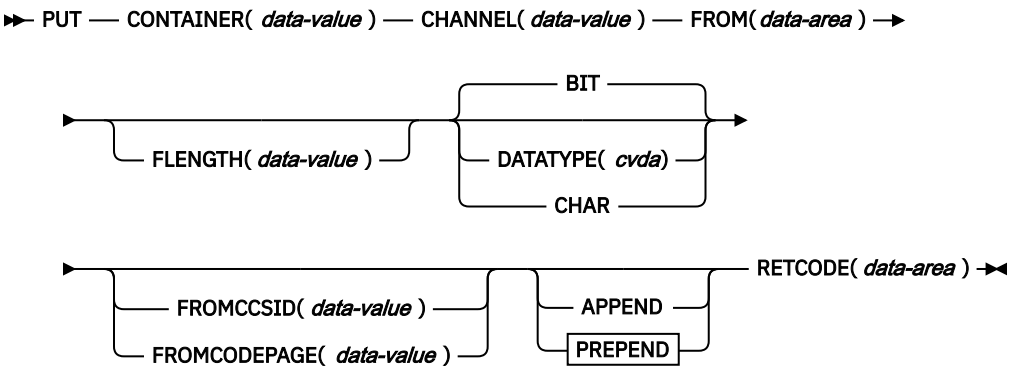
EXEC CICS PUT CONTAINER command (EXCI)

Place data in a named channel container.

Syntax

The syntax differs between releases. The affected command options are represented as a fragment in the main diagram. Some command options are available in specific releases; see "Applicable releases for command options" on page 55 for the affected options.

PUT CONTAINER (EXCI)



Applicable releases for command options

Some command options are available in specific releases, as identified below:

- 6.2 6.3 Beta PREPEND

Conditions: CCSIDERR, CHANNELERR, CODEPAGEERR, CONTAINERERR, INVREQ, LENGERR

Description

PUT CONTAINER (EXCI) places data in a container associated with a specified channel. The container is identified by name. The channel that owns the container is identified by the CHANNEL option.

If the named container does not exist, it is created. If the named channel does not exist, it is created.

6.2 6.3 Beta If the named container exists, its previous contents are overwritten, unless you specify the APPEND or PREPEND option.

6.1 If the named container exists, its previous contents are overwritten, unless you specify the APPEND option.

There is no limit to the number of containers that can be associated with a channel. The size of individual containers is limited only by the amount of storage available. Container data is held in 64-bit storage. The total amount of 64-bit storage available to an EXCI job is governed by **MEMLIMIT**, and the external CICS interface limits the total amount of container data to 5% of the **MEMLIMIT** value.

CAUTION: If you create multiple large containers and pass them to CICS, unless the target CICS program is a non-LE assembler amode-64 application, the container data will be copied into 31-bit storage when accessed by the CICS application, so sufficient 31-bit storage must be available to contain the copied data, and you might limit the amount of storage available to other CICS applications.

Options

APPEND
Specifies that the data passed to the container is appended to the existing data in the container. If neither the APPEND nor the PREPEND option is specified, the existing data in the container is overwritten by the data passed to the container.

CHANNEL(*data-value*)
Specifies the name (1–16 characters) of the channel that owns the container. The acceptable characters are A-Z a-z 0-9 $ @ # / % & ? ! : | " = , ; < > . - and _. Leading and embedded blank characters are not permitted. If the name supplied is less than 16 characters, it is padded with trailing blanks up to 16 characters.

Channel names are always in EBCDIC. The set of allowed characters for channel names, as listed earlier, includes some characters that do not have the same representation in all EBCDIC code pages. Therefore, if channels are to be shipped between regions, it is advisable to restrict the characters that are used to name them to A-Z a-z 0-9 & : = , ; < > . - and _.

If the channel does not exist, it is created. You can specify the channel name DFHTRANSACTION to use a transaction channel in CICS. For more information about using a transaction channel with EXCI, see Channels and containers.

CONTAINER(*data-value*)
Specifies the name (1–16 characters) of the container into which data is placed.

The acceptable characters are A-Z a-z 0-9 $ @ # / % & ? ! : | " = , ; < > . - and _. Leading and embedded blank characters are not permitted. If the name supplied is less than 16 characters, it is padded with trailing blanks up to 16 characters.

Do not use container names that begin with DFH, unless requested to do so by CICS.

Container names are always in EBCDIC. The set of allowed characters for container names, as listed earlier, includes some characters that do not have the same representation in all EBCDIC code pages. Therefore, if containers are to be shipped between regions, it is advisable to restrict the characters used to name them to A-Z 0-9 & : = , ; < > . - and _.

DATATYPE(*cvda*)
Specifies the type of data to put into the container. This option applies only to new containers. If the container exists, its data type was established when it was created and cannot be changed. CVDA values are:

BIT
Bit data. The data in the container cannot be converted. This is the default value, unless **FROMCCSID** is specified.

CHAR
Character data. The data to store in the container is converted (if required) according to the setting in the **FROMCCSID** or **FROMCODEPAGE** value. If the **FROMCCSID** and **FROMCODEPAGE** options are not specified, it is assumed that the data is encoded in the CCSID of the EXCI job as specified in the **LOCALCCSID** parameter of DFHXCOPT.

All the data in a container is converted as if it were a single character string. For SBCS code pages, a structure consisting of several character fields is equivalent to a single-byte character string. However, for DBCS code pages this is not the case. If you use DBCS code pages, you must put each character string into a separate container to ensure that data conversion works correctly.

FLENGTH(*data-value*)
Specifies, as a fullword binary value, the length of the data area from which data is read.

FROM(*data-area*)
Specifies the data area from which the data is written to the container.

FROMCCSID(*data-value*)
Specifies the current Coded Character Set Identifier (CCSID) of the character data to put into the container, as a fullword binary number. If you prefer to specify an IANA name for the code page, or if you prefer to specify the CCSID as alphanumeric characters, use the **FROMCODEPAGE** option instead.

Use this option if the data to place in the container is not encoded in the CCSID of the EXCI job, as specified in the **LOCALCCSID** parameter of DFHXCOPT.

If the FROMCCSID option is specified, DATATYPE(DFHVALUE(CHAR)) is implied.

FROMCODEPAGE(*data-value*)
Specifies an IANA-registered alphanumeric charset name or a Coded Character Set Identifier (CCSID) for the current code page of the character data to put into the container, using up to 40 alphanumeric characters, including appropriate punctuation. Use this option instead of the CCSID option if you prefer to use an IANA-registered charset name, as specified in the Content-Type header for an HTTP request. CICS converts the IANA name into a CCSID, and the subsequent data conversion process is identical. Also use this option if you prefer to specify the CCSID in alphanumeric characters, rather than as a fullword binary number.

If the **FROMCODEPAGE** option is specified, DATATYPE(DFHVALUE(CHAR)) is implied.

6.2 6.3 Beta **PREPEND**
Specifies that the data passed to the container is prepended to the existing data in the container. If neither the PREPEND nor the APPEND option is specified, the existing data in the container is overwritten by the data passed to the container.

RETCODE(data-area)
Specifies a 20-byte area into which the external CICS interface places return information. This area is formatted into five 1–word fields. This command will return information in the first two words. These words will contain the RESP and RESP2 codes. The other three words will be set to nulls.

RESP
The primary response code, which indicates whether the external CICS interface **PUT CONTAINER** command caused an exception condition during its execution.

RESP2
The secondary response code that further qualifies, where necessary, some of the conditions that are raised in the *RESP* parameter.

Conditions

123 CCSIDERR
RESP2 values:

1
The CCSID specified on the **FROMCCSID** option is outside the range of valid CCSID values.

2
The CCSID specified on the **FROMCCSID** option and the CCSID of the container are an unsupported combination. The CCSID of the container is the value that was specified, or defaulted, on the first **PUT CONTAINER** command for this container. The first time each invalid combination is used, CICS issues error message DFHAP0802 , which contains the pair of CCSIDs.

4
One or more characters could not be converted. Each unconverted character has been replaced by a blank in the converted data.

5
There was an internal error in the code page conversion of a container. This error can occur only when the target of the PUT is an existing, CICS-created container.

122 CHANNELERR
RESP2 values:

1
The name specified on the **CHANNEL** option contains an illegal character or combination of characters.

3
The channel specified on the **CHANNEL** option is read-only.

904
XCEIP ESTAE invoked.

125 CODEPAGEERR
RESP2 values:

1
The code page specified on the **FROMCODEPAGE** option is not supported.

2
The code page specified on the **FROMCODEPAGE** option and the CCSID of the container are an unsupported combination. The CCSID of the container is the value that was specified using either **FROMCODEPAGE** or **FROMCCSID** , or defaulted, on the first **PUT CONTAINER** command for this container. The first time each invalid combination is used, CICS issues error message DFHAP0802 , which contains the pair of CCSIDs.

4
One or more characters could not be converted. Each unconverted character has been replaced by a blank in the converted data. This error can occur only when the target of the PUT is an existing container.

5
There was an internal error in the code page conversion of a container. This error can occur only when the target of the PUT is an existing, CICS-created container.

110 CONTAINERERR
RESP2 values:

18
The name specified on the **CONTAINER** option contains an illegal character or combination of characters.

16 INVREQ
RESP2 values:

1
The **DATATYPE** option was specified without the **CHANNEL** option, and there is no current channel (because the program that issued the command was not passed one.) **DATATYPE** is valid only on **PUT CONTAINER** commands that specify (explicitly or implicitly) a channel.

2
The **FROMCCSID** option was specified without the **CHANNEL** option, and there is no current channel (because the program that issued the command was not passed one.) **FROMCCSID** is valid only on **PUT CONTAINER** commands that specify (explicitly or implicitly) a channel.

30
You tried to write to a CICS-defined read only container.

32
A CVDA value other than CHAR or BIT was specified for **DATATYPE**.

33
An attempt was made to change the data-type of an existing container.

34
A data-type of BIT is invalid with a CCSID.

22 LENGERR
RESP2 values:

1
A negative number was specified on the **FLENGTH** option.

EXEC CICS QUERY CHANNEL command (EXCI)

Count the number of containers that are in a channel.

QUERY CHANNEL (EXCI)

Conditions: CHANNELERR

Description

QUERY CHANNEL (EXCI) counts the number of containers that are in a specified channel. You must specify the CHANNEL option and identify the channel explicitly. You can use the **QUERY CHANNEL** command with any channel, including the transaction channel (DFHTRANSACTION), and channels that EXCI created.

Options

CHANNEL(data-value)
Specifies the 1-16 character name of the channel.

CONTAINERCNT(data-area)
Returns, as a fullword binary value, a count of the number of containers that are in the specified channel.

RETCODE(data-area)
Specifies a 20-byte area into which the external CICS interface places return information. This area is formatted into five 1–word fields. This command will return information in the first two words. These words will contain the RESP and RESP2 codes. The other three words will be set to nulls.

RESP
The primary response code, which indicates whether the external CICS interface **QUERY CHANNEL** command caused an exception condition during its execution.

RESP2
The secondary response code that further qualifies, where necessary, some of the conditions that are raised in the *RESP* parameter.

Conditions

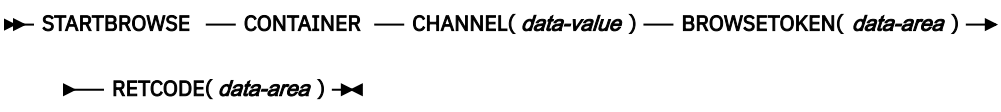
122 CHANNELERR
RESP2 values:

2
The channel specified on the CHANNEL option could not be found.

EXEC CICS STARTBROWSE CONTAINER command (EXCI)

Start a browse of the containers that are associated with a channel.

STARTBROWSE CONTAINER (EXCI)



Conditions: CHANNELERR

Description

STARTBROWSE CONTAINER (EXCI) initializes a browse token that can be used to identify the name of each data-container that is associated with a specified channel.

Note: The browse token should be used only by the program that issues the **STARTBROWSE** command.

The order in which containers are returned is undefined and might change. As best practice, applications should not rely on the order of returned containers.

Options

BROWSETOKEN(data-area)
Specifies a fullword binary data area, into which CICS will place the browse token.

CHANNEL(data-value)
Specifies the name (1-16 characters) of the channel whose containers are to be browsed. This must be the name of a channel created by the program that issues the **STARTBROWSE CONTAINER** command. You can specify the channel name DFHTRANSACTION to use the transaction channel.

The order in which containers are returned is undefined.

RETCODE(data-area)
Specifies a 20-byte area into which the external CICS interface places return information. This area is formatted into five 1–word fields. This command will return information in the first two words. These words will contain the RESP and RESP2 codes. The other three words will be set to nulls.

RESP
The primary response code, which indicates whether the external CICS interface **STARTBROWSE CONTAINER** command caused an exception condition during its execution.

RESP2
The secondary response code that further qualifies, where necessary, some of the conditions that are raised in the *RESP* parameter.

Conditions

122 CHANNELERR
RESP2 values:

2
The channel specified on the CHANNEL option could not be found.

Compiling and link-editing EXCI client programs

All programs that use the external CICS interface to pass DPL requests to a CICS server region must include the CICS-supplied program stub, DFHXCSTB.

Alternatively for COBOL programs, it is possible to call DFHXCSTB dynamically rather than link-editing the stub with the program.

The stub intercepts all external CICS interface commands, whether they are EXCI CALL interface commands or EXEC CICS interface 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 CICSTS*nn*.CICS.SDFHEXCI library, where *nn* represents the CICS version.

Note: The CICSTS*x*.CICS.SDFHEXCI also contains entries for DFHXCIE and DFHXCIS, which are aliases for DFHXCSTB.

To help you ensure that the stub is included, CICS provides a number of procedures, one for each language, which you can use for translating, compiling, and link-editing.

You must specify AMODE(31) for your EXCI client program.

The CICS-supplied procedures for compiling and link-editing client programs include the following parameters on the PARM statement of the linkage editor job step:

```
LNKPARM='AMODE(31),LIST,XREF'
```

Samples

To help with writing programs that use the external CICS interface, CICS provides sample z/OS client programs and a sample CICS server program. The samples show you how to code client applications that use both the **EXCI CALL** interface and **EXEC CICS LINK** command. For more information, see EXCI sample programs.

Job control language to run an EXCI client program

An EXCI client program runs in a z/OS address space, for example, as a batch job. When writing the JCL for your client program, you should note these requirements.

Requirements

- Include in the STEPLIB concatenation those libraries that contain the CICS-supplied external CICS interface modules and also the client program. All the libraries are shown using the format CICSTS*nn*, where *nn* represents the CICS version. The external CICS interface modules are supplied in CICSTS*nn*.CICS.SDFHEXCI, which contains the following:

  DFH$ATXC
  DFH$AXCC
  DFH$AXNC
  DFH$DXVC
  DFHMEBMX
  DFHMET4E
  DFHXCEIX
  DFHXCIE (alias of DFHXCSTB)
  DFHXCIS (alias of DFHXCTSB)
  DFHXCOPT
  DFHXCPRX
  DFHXCSTB
  DFHXCTRA
  DFHXCURM
- You are recommended to include a DD statement for SYSMDUMP. The external CICS interface uses SYSMDUMP for some error conditions.
- The **REGION** parameter must specify a large enough region size to allow for the size of the internal trace table specified by the **TRACESZE** parameter in the DFHXCOPT options table.
- If the EXCI client program uses channels and containers, instead of a COMMAREA, to pass data to CICS, the **MEMLIMIT** parameter must be specified as container data is stored in 64-bit storage above the bar. Storage for containers cannot exceed 5% of the **MEMLIMIT** value.
- Include a SYSPRINT or equivalent DD statement for any output from the client program.

Sample job for starting an EXCI client program

Figure 13 on page 62 shows a sample job that you can use or modify to start a client program.

```
//EXCI JOB (accounting_information),CLASS=A,TIME=1440,
// USER=userid,PASSWORD=pswd,REGION=100M
//*-------------------------------------------------------------*
//* JCL to execute an external CICS interface client program *
//*-------------------------------------------------------------*
//EXEC PGM=pgmname,REGION=nnM,MEMLIMIT=nnG
//STEPLIB DD DSN=CICSTSnn.CICS.EXCI.LOADLIB,DISP=SHR
// DD DSN=CICSTSnn.CICS.SDFHEXCI,DISP=SHR
//SYSPRINT DD SYSOUT=A
//SYSMDUMP DD DSN=SYS1.SYSMDP00,VOL=SER=volid,SPACE=(CYL,(1,1)),
// DISP=OLD,UNIT=3390
```

Figure 13. Sample job for starting an EXCI client program

Note:

1. The job user ID, specified on the **USER** parameter, must be defined to RACF.
2. In addition to being used for job step initiation security, the job user ID is also used for MRO logon and bind-time security checking.

   See EXCI security for information about security when using the external CICS interface.
3. See Setting up the EXCI sample programs for information about modifying the sample connection definitions before you run the sample application programs in an environment that does not have RACF installed and active.

CICS-supplied procedures for the EXCI

CICS provides nine procedures you can use to translate, compile, and link-edit your client programs.

One assembler procedure is provided. For the high-level languages, a variant that uses the stand-alone translator and one that uses the integrated translator is provided, as shown in the following list:

DFHEXTAL
The assembler procedure for assembler versions of client programs and that use the stand-alone translator.

DFHYXTDL
The procedure for C versions of client programs that are running under Language Environment® and using the stand-alone translator.

DFHYXTEL
The procedure for C++ versions of client programs that are running under Language Environment and using the stand-alone translator.

DFHYXTPL
The procedure for PL/I versions of client programs that are running under Language Environment and using the stand-alone translator.

DFHYXTVL
The procedure for COBOL versions of client programs that are running under Language Environment and using the stand-alone translator.

DFHZXTCL
The procedure for COBOL versions of client programs that are running under Language Environment and using the integrated translator.

DFHZXTDL
The procedure for C versions of client programs that are running under Language Environment and using the integrated translator.

DFHZXTEL
The procedure for C++ versions of client programs that are running under Language Environment and using the integrated translator.

DFHZXTPL
The procedure for PL/I versions of client programs that are running under Language Environment and using the integrated translator.

To ensure that the EXCI stub is included with your client program, all these procedures include a step, COPYLINK, that unloads the stub into a temporary data set defined with a block length suitable for the linkage-editor. This temporary data set is then concatenated with the temporary data set containing your object program on the SYSLIN DD statement in the LKED step.

These procedures are supplied in the CICSTS*nn*.CICS.SDFHPROC library, where *nn* represents the CICS version. You are recommended to copy them to SYS1.PROCLIB or another suitable procedure library.

EXCI programming considerations

There are some language requirements that apply to writing a z/OS client program that uses the external CICS interface. These affect programs written in PL/I and C. Also, for all languages, consider how you handle return codes before terminating your z/OS client program.

PL/I considerations

PL/I programs written to the external CICS interface must provide their parameters on the CALL to DFHXCIS in the form of an assembler-style parameter list.

The EXCI copybook for PL/I, DFHXCPLL, contains the necessary definition of the DFHXCIS entry point, as follows:

```
DCL DFHXCIS ENTRY OPTIONS(INTER ASSEMBLER);
```

The same rule applies for the EXCI LINK command, and in this case the CICS translator ensures that the correct parameter list is built.

For an example of an EXCI client program written in PL/I, see the source of the sample program, DFH$PXCC.

C considerations

C programs written to the external CICS interface must provide their parameters on the CALL to DFHXCIS in the form of an assembler-style parameter list. You ensure this by declaring the entry point to DFHXCIS with OS LINKAGE.

The EXCI copybook for 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.

Setting the return code (R15) at termination

The external CICS interface does not clear register 15 at termination, regardless of whether your client program executes normally or not. Therefore, even if your z/OS client program terminates normally after successfully using the external CICS interface, the job step could end with an undefined return code.

The external CICS interface is not obliged to clear register 15 and does not clear it upon exit to the client program that invokes it. The value in register 15 reflects a return code for the entire application, not just for the external CICS interface. If register 15 is cleared during exit from the external CICS interface, it might suggest that the client application works without any issue and this might lead the client to make incorrect decisions. By not setting register 15 when control returns from the external CICS interface, it is the responsibility of the client program to set it since it is aware of the application's overall state.

To ensure a meaningful return code is given at termination, set the job step return code before terminating your program. The sample client programs illustrate how you can do this, using the saved

response code from last call to the external CICS interface. For example, the COBOL sample DFH0CXCC program moves SAVED-RESPONSE to special register RETURN-CODE before terminating.

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.

Setting the return code (R15) at termination

The external CICS interface does not clear register 15 at termination, regardless of whether your client program executes normally or not. Therefore, even if your z/OS client program terminates normally after successfully using the external CICS interface, the job step could end with an undefined return code.

To ensure a meaningful return code is given at termination, set the job step return code before terminating your program. The sample client programs illustrate how you can do this, using the saved response code from last call to the external CICS interface. For example, the COBOL sample DFH0CXCC program moves SAVED-RESPONSE to special register RETURN-CODE before terminating.

Chapter 3. Configuring EXCI

The external CICS interface (EXCI) is an application programming interface that enables a non-CICS program (a client program) running in z/OS to call a program (a server program) running in a CICS region and to pass and receive data by using a communications area or by using a channel and a set of containers. This section provides instructions for configuring EXCI.

Setting up EXCI for static routing

You can statically route requests to CICS programs from applications that use the EXCI.

Before you begin

Before you begin, verify that the MVS parameter **Maxmember** is set to a high value. This parameter controls how many connections can be made to the DFHIRP00 resource.

Procedure

1. Add RDO group EXCIXXXX to the grouplist of the CICS region. If EXCIXXXX is not available, make a copy from the supplied DFH$EXCI RDO group.

   This group contains all connections required for EXCI functions and can support up to 100 connections for batch requests.
2. Add the RDO group for the application to the grouplist of the CICS region.
3. Assemble DFHXCOPT into the SDFHEXCI load library. Ensure that DFHXCOPT has SURROGATE=YES.
4. Assemble and compile your application programs.

   If your application program is written in Assembler, use the linkage editor parameters AMODE(31) and RMODE(ANY). Link the program into your application load library.
5. Configure the batch JCL to run your application program.
   a) Edit the JCL to specify to which CICS region the batch program will send the EXCI request:

   ```
   //step0010 EXEC PGM=program,PARM='applid,userid'
   ```

   applid is the CICS region and *userid* is a RACF user ID.

   b) Ensure your load library is concatenated as follows:

   6.2

   ```
   //STELIB DD Disp=shr,Dsn=SYS5C.CICn.CICS750.SDFHEXCI
                //DD Disp=shr,Dsn=Your.application.loadlib
   ```

   6.1

   ```
   //STELIB DD Disp=shr,Dsn=SYS5C.CICn.CICS740.SDFHEXCI
                //DD Disp=shr,Dsn=Your.application.loadlib
   ```
6. Run the batch program and check that the results are as expected.

Setting up EXCI for dynamic routing

You can dynamically route requests to CICS programs from applications that use the EXCI using CICSPlex® SM.

Procedure

1. Specify the following system initialization parameters in the CICS region:
   - DSRTPGM=EYU9XLOP
   - DTRPGM=EYU9XLOP
2. Update your RDO group as follows:
   a) Add an RDO entry for the EXCI server program, DFHMIRS.

      You can use another transaction instead of EXCI if required, but it must point to program DFHMIRS and have a profile of DFHCICSA. Model it after the EXCI transaction in group DFH$EXCI. This transaction will point to DFHMIRS program.

      Note: The user transaction when defined as remote will only work within an APPC configuration. Within an EXCI or MRO configuration the mirror transaction must run in the local CICS region, by specifying DYNAMIC(NO) and no REMOTE attributes. Routing the mirror transaction to another CICS region can impact performance and make problem determination more difficult.

   b) Add the RDO group to the terminal-owning region (TOR) LIST.

   Ensure that the TOR region has program autoinstall disabled.
3. Log into CICSPlex SM and define the transaction for the routing region into these CICSPlex SM resources: TRANGRP, WLMDEF, WLMGRP WLMSPEC.

   For more information, see Creating workload management definitions.
4. For each application-owning region (AOR), create the RDO group in the same way as the sample definition DFH$EXCI.

   This group must contain only the connections, sessions, and application programs, or equivalent transactions. If you used a different transaction instead of EXCI, you must specify it in this group.

Results

When an application issues a distributed program link, CICSPlex SM checks if the incoming transaction is under its control. When it finds that EXCI or its equivalent is valid in its transaction group, CICSPlex SM routes the transaction to one of the candidate AORs for processing.

An alternative to this approach is to define an RDO definition for the program or transaction with DYNAMIC=Yes. CICSPlex SM routes the request to the selected region.

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. See CONNECTION resources. For EXCI connections only, this option indicates whether the connection is generic or specific.
- EXCI on the PROTOCOL attribute of the CONNECTION and SESSIONS resource definitions. See CONNECTION resources and SESSIONS resources.

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 (EXCI) 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 EXCI batch jobs currently connected to CICS through the interregion communication (IRC) facility.

CICS returns job identifications in the form: `jobname.stepname.procname - mvsid`

Either `stepname`, or `procname`, or both might not be present, indicated by the periods (.) being adjacent to one another.

The `mvsid` identifies the z/OS system on which the job is running. If XCF/MRO is in use, the job can reside on a different z/OS image from that on which CICS is running.

Information about jobs using the external CICS interface is available only when the job has issued at least one DPL request. A non-zero task number indicates that a DPL request is currently active. A zero task number indicates an external CICS interface session is still open (connected) for that job, although no DPL request is currently active.

See CEMT - main terminal for more information about the CEMT command.

The EXCI user-replaceable module

The external CICS interface provides a user-replaceable module, DFHXCURM.

The load module and its source are supplied in CICS libraries: `CICSTS`*nn*`.CICS.SDFHEXCI` and `CICSTS`*nn*`.CICS.SDFHSAMP` (source), where *nn* reflects the release of CICS TS. For example, in CICS TS beta, the libraries are CICSTS64.CICS.SDFHEXCI and CICSTS64.CICS.SDFHSAMP. You can find information about assembling and link-editing user-replaceable programs in Assembling and link-editing user-replaceable programs.

DFHXCURM is started by the external CICS interface in the non-CICS region during the processing of **allocate_pipe** commands, and after the occurrence of any retryable error.

The retryable responses are:

- The target CICS region is not available.
- There are no pipes available on the target CICS region.
- There has been no IRC activity since the MVS IPL.

To retry after a retryable error, issue the EXCI call again.

As supplied, DFHXCURM is effectively a dummy program because of a branch instruction that bypasses the sample logic and returns control to the external CICS interface caller. To use the sample logic, remove the branch instruction and assemble and link edit the module. You can customize DFHXCURM to do the following actions:

- During allocate_pipe processing, you can change the specified CICS APPLID, to route the request to another CICS system.
- During allocate_pipe processing, you can direct the request to a different XCF group.
- When DFHXCURM is started after an error that can be tried again, you can store information about CICS availability. You can then use this information on the next invocation of DFHXCURM for allocate_pipe processing, so that you can decide which CICS system to route the request.

DFHXCURM is called using standard MVS register conventions, with register 1 containing the address of the parameter list, and register 14 the return address of the caller. The parameters addressed by register 1 are mapped in the EXCI_URM_PARMS DSECT, which is contained within the DFHXCPLD copybook. The parameters passed to DFHXCURM are as follows:

URMINV
The address of a fullword that contains the reason for the invocation of DFHXCURM, defined by the following equates:

```
URM_ALLOCATE     EQU 1  This invocation is for an Allocate_Pipe
URM_NO_CICS      EQU 2  The target CICS region is not available
URM_NO_PIPE      EQU 3  There are no pipes available
URM_NO_CICS_IRC  EQU 4  There has been no IRC activity since the MVS IPL
```

URMCICS
The address of an 8-byte area that contains the 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. Also, if the APPLID specified by one of these commands is not a valid specific applid, you must change the APPLID to that of a valid specific applid.

If the **CICS_applid** parameter is omitted from the allocate_pipe request, 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. If DFHXCURM is started 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 started for an EXEC CICS LINK command. For an external CICS interface allocate_pipe command, the program name is not known until the DPL call is issued.

URMOPTS
The address of a 1-byte area that contains the allocate options, which can be X'00' or X'80', as specified on the *allocate_opts* parameter. This address is valid for an Allocate_Pipe request only.

URMANCH
The address of a 4-byte area that is provided for use by DFHXCURM only. A typical use for this is to store a global anchor address of an area used to save information across a number of invocations of DFHXCURM. For example, you can GETMAIN the necessary storage and save the address in the 4-byte area addressed by this parameter. The initial value of the 4-byte area is set to zero.

There is one URMANCH parameter for each TCB in the address space using EXCI.

URMXCFG
The address of an 8-byte area that contains the XCF group name as specified in the **XCFGROUP** parameter of the DFHXCOPT table. Use this parameter to change the XCF group name when DFHXCURM is called during the processing of EXCI Allocate_Pipe commands. If the call to DFHXCURM failed but can be tried again, the area contains the value used when previously allocating the pipe. Changing the value has no affect.

The group name must be 8 characters long, padded on the right with blanks if necessary. Valid characters are A-Z 0-9 $ # @. Do not begin group names with the letters A, B, C, E, F, G, H, I "SYS." These names are used by IBM for its XCF groups. Also, do not use the name "UNDESIG", which is reserved for use by the system programmer in your installation.

It is advisable to use a group name beginning with the letters "DFHIR".

Using the EXCI options table, DFHXCOPT

The EXCI options table, which is generated by the DFHXCOPT macro, enables you to specify a number of parameters that are required by the external CICS interface.

DFHXCOPT options table: New format versus older format

The DFHXCOPT options table changed since it was first introduced and now includes a version number to allow more flexibility for future extensions. You need to be aware of this change if, for example, you plan to migrate a customized DFHXCOPT table from an earlier release of CICS.

To distinguish between the old and newer formats, the new-format table is link-edited with an alias called DFHXCOPE. The following sequence is used to load the options table:

1. CICS tries to load the DFHXCOPT table using its alias name of DFHXCOPE. If it finds and successfully loads a load module named DFHXCOPE, CICS assumes that the table is in the new format.
2. If CICS does not find a load module named DFHXCOPE (or finds it but fails to load it), it tries to load the table using its *base* name of DFHXCOPT. In this case, CICS assumes that the table is in the older format.

CICS provides a default DFHXCOPT table and supplies the source code of the default table in the `CICSTS`*nn*`.CICS.SDFHSAMP` library, where *nn* reflects the release of CICS TS. For example, the library is CICSTS64.CICS.SDFHSAMP for CICS TS beta. The load module of the default DFHXCOPT table, with its alias DFHXCOPE, is in the `CICSTS`*nn*`.CICS.SDFHEXCI` library, where *nn* reflects the release of CICS TS. For example, the library is CICSTS64.CICS.SDFHEXCI for CICS TS beta.

Creating customized DFHXCOPT table

You can tailor the source code of the CICS-supplied default DFHXCOPT table to your own requirements.

You must assemble and link-edit your customized DFHXCOPT table into a suitable library in the STEPLIB concatenation of the job that runs the z/OS client program.

Important: If you create your own, customized, DFHXCOPT table, ensure that you link-edit it using the DFHXCOPE alias. Using the standard DFHAUPLE procedure ensures that this happens. If you reassemble and link-edit your table without the alias, CICS will load the default table (found by means of its DFHXCOPE alias), rather than your customized table.

You can use your own version of the CICS DFHAUPLE procedure to do this. The DFHAUPLE procedure is supplied in the `CICSTS`*nn*`.CICS.SDFHINST` library, where *nn* reflects the release of CICS TS. For example, the library is CICSTS64.CICS.SDFHINST for CICS TS beta.

DFHXCOPT macro: Format and parameters

Unlike the tables you specify for CICS regions, the DFHXCOPT table cannot be suffixed.

The following table shows the format of the DFHXCOPT macro and its parameters.

DFHXCO	TYPE={**CSECT**|DSECT}
	[,ABENDBKOUT={**NO**|YES}]
	[,CICSSVC={**216**|*number*}]
	[,CONFDATA={**HIDE**|SHOW}]
	[,DURETRY={**30**|*number-of-seconds*}]
	[,GTF={**OFF**|ON}]
	[,LOCALCCSID={**037**|*CCSID*}]
	[,MSGCASE={**MIXED**|UPPER}]
	[,TIMEOUT={**0**|*number*}]

	[,TRACE={**OFF**|1|2|3}] [,TRACESZE={**16**|*number-of-kilobytes*}] [,TRAP={**OFF**|ON}] [,XCFGROUP={**DFHIR000**|name}] You must terminate your parameters with the following END statement.
END	DFHXCOPT

TYPE={CSECT|DSECT}
Indicates the type of table to be generated.

CSECT
A regular control section that is normally used.

DSECT
A dummy control section.

ABENDBKOUT={NO|YES}
Specifies whether a task that abends within the CICS server is to trigger an automatic rollback of the global unit of work. A global unit of work exists when an EXCI client program is controlling resource recovery through z/OS RRS (that is, SYNCONRETURN is *not* specified on the DPL request). In this case, you may well want the global unit of work to be marked for rollback if the CICS server program abends.

Note: ABENDBKOUT has no effect when SYNCONRETURN is specified on the DPL request.

NO
The global unit of work is not marked for rollback.

YES
When processing the abend of the server program, the CICS mirror program marks the global unit of work for backout.

In both cases, the EXCI client program receives a return code of 422, SERVER_ABENDED, on the EXCI DPL request.

CICSSVC={216|*number*}
Specifies the CICS type 3 SVC number being used for MRO communication. The default is 216.

The external CICS interface must use the same SVC number that is in use by the CICS MRO regions that reside in the z/OS image in which the client program is running.

0
Specify zero to indicate that the external CICS interface is to obtain the CICS SVC number from z/OS by means of a z/OS **VERIFY** command.

You should only specify zero when you are sure that at least one CICS region has logged on to DFHIRP during the life of the z/OS IPL.

number
Specify the CICS SVC number, in the range 200—255, that is in use for CICS interregion communications. This must be the SVC number that is installed in the z/OS image in which the client program is running (the local z/OS).

If no MRO CICS regions have ever logged on to DFHIRP in the local z/OS during the life of the IPL, you must specify a non-zero SVC number. If you specify zero, the external CICS interface requests the SVC from z/OS, which will fail if no CICS region has logged on to DFHIRP.

A non-zero value is required in those z/OS images that do not run any CICS regions, and the client program is issuing DPL requests to a server CICS region that resides in another z/OS. In these circumstances, the client program logs on to the local DFHIRP using the locally defined SVC, and communicates with the remote CICS region using XCF/MRO.

Note: All CICS regions using MRO within the same z/OS image must use the highest level of both DFHIRP and the CICS SVC, DFHCSVC. If your MRO CICSplex consists of CICS regions at different release levels, the DFHIRP and DFHCSVC installed in the LPA must be from the highest release level of CICS within the CICSplex.

CONFDATA={HIDE|SHOW}

Code this parameter to indicate whether the external CICS interface is to suppress (hide) user data that might otherwise appear in EXCI trace entries output to GTF or in EXCI dumps. This option applies to the tracing of the COMMAREA or CONTAINER data flowing between the EXCI client program and the CICS server program.

HIDE

EXCI is to 'hide' user COMMAREA or CONTAINER data from trace entries. Instead the trace entry contains a character string stating that the data has been suppressed.

SHOW

Data suppression is not in effect. User data is traced.

DURETRY={30|*number-of-seconds*|0}

Specifies the total time, in seconds, that the external CICS interface is to continue trying to obtain a z/OS system dump using the SDUMP macro.

DURETRY allows you to control whether, and for how long, the external CICS interface is to reissue the SDUMP if another address space in the same z/OS system is already taking an SDUMP when the external CICS interface issues an SDUMP request.

In the event of an SDUMP failure, the external CICS interface reacts as follows:

- If z/OS is already taking an SDUMP for another address space, and the DURETRY parameter is nonzero, the external CICS interface issues an MVS STIMERM macro to wait for five seconds, before retrying the SDUMP macro. The external CICS interface issues a message to say that it will retry the SDUMP every five seconds until the DURETRY time limit.
- If the SDUMP fails for any other reason such as the following, the external CICS interface issues a message to inform you that the SDUMP has failed, giving the reason why.
  - There are no SYS1.DUMP data sets available.
  - There are I/O errors preventing completion of the dump.
  - The DURETRY limit expires while retrying SDUMP.

30

30 seconds allows the external CICS interface to retry up to six times (once every five seconds).

number-of-seconds

Code the total number of seconds (up to 32767 seconds) during which you want the external CICS interface to continue retrying the SDUMP macro. The external CICS interface retries the SDUMP, once every five seconds, until successful or until retries have been made over a period equal to or greater than the DURETRY value.

0

Code a zero value if you do not want CICS to retry the SDUMP.

GTF={OFF|ON}

Specifies whether all trace entries normally written to the external CICS interface trace table are also to be written to a z/OS generalized trace facility (GTF) data set (if GTF trace is active).

OFF

Code this if trace entries are not to be written to GTF.

ON

Code this if trace entries are to be written to GTF.

LOCALCCSID={037|*CCSID*}
Specifies the default CCSID for the EXCI job. The CCSID is a value of up to 8 characters. If a CCSID value is not specified, the default LOCALCCSID is set to 037. For lists of valid CCSIDs, see the following information:

- CICS-supported conversions
- The relevant appendix in z/OS Unicode Services User's Guide and Reference

037
The default value for LOCALCCSID.

CCSID
Represents any other valid EBCDIC CCSID value.

MSGCASE={MIXED|UPPER}
Specifies whether the DFHEX*xxxx* messages are to be issued in mixed case or in 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.

DPL commands can pass a channel and set of containers to CICS. Commands that pass this information can involve multiple flows of data up to CICS to construct the container data. The timeout interval starts when the last flow of data is sent to CICS, which completes the data and initiates the invocation of the CICS server program.

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|3}
Specifies whether you want internal trace for the external CICS interface, and at what level.

OFF
Internal trace for the external CICS interface is not required. However, even with normal tracing switched off, exception trace entries are always written to the external CICS interface trace table in the CICS region.

1
Exception and level 1 trace entries are written to the external CICS interface trace table.

2
Exception, level 1, and level 2 trace entries are written to the external CICS interface trace table.

3
Exception, level 1, level 2, and level 3 trace entries are written to the external CICS interface trace table.

TRACESZE={16|*number-of-kilobytes*}
Specifies the size in kilobytes of the trace table that is used by the external CICS interface. This trace table is allocated in 31-bit storage (above the line) in the CICS region.

16
The default size of the trace table, and also the minimum size.

number-of-kilobytes
The number of kilobytes of storage to be allocated for the trace table, in the range 16 KB through 1 048 576 KB. Subpool 1 is used for the trace table storage, which exists for the duration of the job step TCB. The table is page-aligned and occupies a whole number of pages. If the value specified is not a multiple of the page size (4 KB), it is rounded up to the next multiple of 4 KB.

TRAP={OFF|ON}
Specifies whether the service trap module, DFHXCTRA, is to be used. DFHXCTRA is supplied as a user-replaceable module, in which IBM service personnel can add code to trap errors.

OFF
Code this if you do not want to use DFHXCTRA.

ON
Code this if you require DFHXCTRA.

XCFGROUP={DFHIR000|name}
Specifies the name of the cross-system coupling facility (XCF) group to be joined by this client program.

Note: XCF groups allow CICS regions in different MVS images within the same sysplex to communicate with each other across multi-region operation (MRO) connections. For introductory information about XCF/MRO, and instructions on how to set up XCF groups, see Cross-system multiregion operation (XCF/MRO).

Each client program can join a maximum of one XCF group.

DFHIR000
The default XCF group name.

name
The group name must be eight characters long, padded on the right with blanks if necessary. The valid characters are A-Z 0-9 and the national characters $ # and @. To avoid using the names IBM uses for its XCF groups, do not begin group names with the letters A through C, E through I, or the character string "SYS". Also, do not use the name "UNDESIG", which is reserved for use by the system programmer in your installation.

It is recommended that you use a group name beginning with the letters "DFHIR".

Chapter 4. Security for EXCI

CICS applies security checks in a number of ways against requests received from an MVS client program.

Using MRO logon and bind-time security

DFHIRP, the CICS interregion communication program, performs two security checks against users that want to either log on to IRP (specific connections only), or connect to a CICS region (also referred to as bind-time security).

About this task

Generic EXCI connections: The discussion about logon security checking in this section applies only to EXCI connections that are defined as SPECIFIC. The MRO logon security check is not performed for generic connections.

The MVS client program is treated just the same as another CICS region as far as MRO logon and connect (bind-time) security checking is concerned. This means that when the client program logs on to the interregion communication program, IRP performs logon and bind-time security checks against the user ID under which the client program is running. In the remainder of this information, this user ID is called the user ID of the batch region.

To enable your client program to log on to IRP successfully, and to connect to the target server region, first ensure that you define the user ID of the batch region in a user profile to RACF. After you define the user ID of the batch region to RACF, you can then give the batch region the appropriate logon and bind-time authorizations.

Procedure

Logon authorization:

- Authorize the user ID of the batch region to the DFHAPPL.*user_name* RACF FACILITY class profile, with UPDATE authority. The *user_name* part of the profile name is the user name defined on the INITIALIZE_USER command.
  - For the EXCI CALL interface, the *user_name* must be the name you specify on the *user_name* parameter of the INITIALIZE_USER command.

    Define FACILITY class profiles, with appropriate authorizations, for each user name specified in a client program if the program has INITIALIZE_USER commands for more than one user name.

    For example, if the *user_name* defined on an INITIALIZE_USER command is DCEUSER1, define the DFHAPPL profile in the FACILITY class as follows:

    ```
    RDEFINE FACILITY (DFHAPPL.DCEUSER1) UACC(NONE)
    ```

    If the batch region's user ID is CLIENTA, authorize the batch region to log on to IRP as follows:

    ```
    PERMIT DFHAPPL.DCEUSER1  CLASS(FACILITY) ID(CLIENTA)
            ACCESS(UPDATE)
    ```
  - For the EXEC CICS **LINK** command, the *user_name* is preset by the external CICS interface as DFHXCEIP. This does not require authorization for IRP logon because the EXEC CICS LINK interface uses a generic connection to which the logon security check does not apply.

  Failure to authorize the user ID of the batch region to the DFHAPPL profile of the specific user ID logging on to IRP causes Allocate_Pipe processing to fail with RESPONSE(SYSTEM_ERROR) REASON(IRC_LOGON_FAILURE). The subreason field-1 for a logon security check failure returns decimal 204.

Bind-time authorization

- Authorize the user ID of the batch region to the DFHAPPL.*applid* RACF FACILITY class profile of the target CICS server region, with READ authority.

  Failure to authorize the user ID of the batch region to the DFHAPPL.*applid* profile of the CICS server region causes Open_Pipe processing to fail with RESPONSE(SYSTEM_ERROR) REASON(IRC_CONNECT_FAILURE). The subreason field-1 for a bind-time security check failure returns decimal 176.

  See Bind security for information about the MRO logon and bind-time security checks, and for examples of how to define the RACF DFHAPPL profiles.

Link security for EXCI

The target CICS server region performs link security checking against requests from the client program.

These security checks cover transaction attach security (when attaching the mirror transaction), and resource and command security checking within the server application program. The link user ID that CICS uses for these security checks is the batch region's user ID.

To ensure these link security checks do not cause security failures, you must ensure that the link user ID is authorized to the following resource profiles, as appropriate:

- The profile for the mirror transaction, either CSMI for the default, or the mirror transaction specified on the *transid* parameter. This is required for transaction attach security checking.
- The profiles for all the resources accessed by the CICS server application program—files, queues (transient data and temporary storage), programs, and so on. This is required for resource security checking.
- The CICS command profiles for the SPI commands issued by the CICS server application program—INQUIRE, SET, DISCARD and so on. This is required for command security checking.

See CONNECT security checks for AORs for information about MRO link security checking.

User security for EXCI

The target CICS server region performs user security checking against the user ID passed on a DPL_ Request call. User security checking is performed only when connections specify ATTACHCSEC(IDENTIFY).

User security is performed in addition to any link security.

For user security, in addition to any authorizations you make for link security, you must also authorize the user ID specified on the DPL_Request call.

Note that there is no provision for specifying a user ID on the EXEC CICS LINK command. In this case, the external CICS interface passes the batch region's user ID. User security checking is therefore performed against the batch region's user ID if the connection definition specifies ATTACHSEC(IDENTIFY).

Note: If your connection resource definitions for the external CICS interface specify ATTACHSEC(IDENTIFY), your server programs will fail with an ATCY abend if you run them in an environment that does not have RACF, or an equivalent external security manager (ESM), installed and active.

If you want to run external CICS interface server programs without any security active, you must specify ATTACHSEC(LOCAL).

Surrogate user checking for EXCI

EXCI client jobs are subject to surrogate user checking (see Surrogate security). You must authorize the batch region's user ID as a surrogate of the user ID specified on all DPL_Request calls. This configuration means that the batch region's user ID must have READ access to a profile named *execution-userid*.DFHEXCI in the RACF SURROGAT general resource class (where *execution-userid* is the user ID that is specified on the DPL call).

For example, the following commands define a surrogate profile for a DPL user ID, and grant READ access to the EXCI batch region:

```
RDEFINE  SURROGAT execution_userid.DFHEXCI UACC(NONE)
PERMIT execution-userid.DFHEXCI CLASS(SURROGAT) ID(batch_region_userid) ACCESS(READ)
```

If no user ID is specified on the DPL_Request call, no surrogate user check is performed because the user ID on the DPL_Request call defaults to the batch region's user ID. For this bypass of surrogate user checking to be successful, ensure that you have correctly omitted the user ID on the DPL_Request call. See the example of EXCI CALLs with null parameters in The EXCI CALL interface for the correct way to specify a null pointer when you omit an EXCI call parameter.

If the batch region user ID and the CICS region user ID are different, link security checking is enforced. With link security, a nonauthenticated user ID passed on a DPL_Request call cannot acquire more authority than that allowed by the link security check. It can acquire only the same, or less, authority than that allowed by the link security check.

For more information about CICS security, see Securing CICS.

Chapter 5. Troubleshooting EXCI

Important: This information contains Diagnosis, Modification or Tuning information.

The external CICS interface (EXCI) provides diagnostic information that you can use to help with problem determination.

CICS provides the following diagnostic information:

- Trace
- System dumps
- 04xx abends for the external CICS interface
- The EXCI service trap, DFHXCTRA
- EXCI trace points

For details of EXCI messages and abend codes, see DFHEX messages and 04xx (external CICS interface) abend codes.

EXCI trace

The external CICS interface (EXCI) writes trace data to two destinations: an internal trace table and an external z/OS GTF data set. The internal trace table resides in the non-CICS address space. Trace data is formatted and included in any dumps produced by the EXCI.

Trace entries issued by the EXCI are listed in External CICS interface trace points.

To use z/OS GTF for EXCI tracing, CICS GTF trace must be active, z/OS GTF must be started in the z/OS image, and you must specify GTF=ON in the DFHXCOPT options table. If you use GTF trace for both the CICS server region and the EXCI region, the trace entries are interleaved, which can help you with problem determination in the CICS-EXCI environment.

Note: The EXCI maintains a separate trace table for each user TCB in an EXCI application program.

The EXCI does not support any form of auxiliary trace.

To format EXCI trace entries written to z/OS GTF, you can use the standard CICS trace formatting routine, DFHTR*nnn*, where *nnn* is a release identifier. For example, it is DFHTR760 for CICS TS beta. For reference information about this utility, see Trace utility print program (DFHTUnnn).

To format EXCI trace entries, you use the same FID and ID as for CICS (that is, FID=X'EF', and ID=X'F6C').

EXCI system dumps

The external CICS interface (EXCI) produces SYSMDUMPs for some error conditions and SDUMPs for other, more serious conditions. These dumps contain all the EXCI control blocks as well as trace entries.

Note: To capture SYSMDUMPs produced by the EXCI, ensure that you always include a DD statement for the SYSMDUMP data set in the client application program's JCL.

Forcing dumps

In addition to the dumps taken automatically by the EXCI, you can force a dump of an address space running a client application program by entering the z/OS **DUMP** command at the console. You can use the CICS IPCS VERBEXIT routine DFHPD*nnn* to format dumps taken this way.

You can also use the **DUMP** command to dump the CICS server address space as well as the client address space. Use the CICS IPCS VERBEXIT routine DFHPD*nnn* to format the dump that contains both address spaces.

Formatting system dumps

You can use the CICS IPCS VERBEXIT routine, DFHPD*nnn* to format the system dumps.

```
IPCS VERBEXIT DFHPDnnn keyword
```

DFHPD*nnn* is the dump formatting program, where *nnn* is a release identifier. For example, it is DFHPD760 for CICS TS beta. For reference information about the CICS dump utilities, see Dump utilities (DFHDUnnn and DFHPDnnn).

The following keywords are available for use when formatting an EXCI dump:

KE
Formats PSW and registers, and all EXCI control blocks.

LD
Formats a load map of where the EXCI modules are loaded in the address space, and gives their PTF level.

MRO
Formats the MRO control blocks for the EXCI address space, including common control blocks that reside in the common service area (CSA). This option also formats some MRO blocks that reside in the CICS address space for pipes connected to CICS.

PG
Formats the PG control blocks for channels and containers.

TR
Formats the EXCI 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.

Note: If the EXCI takes a system dump when there is more than one TCB in use, it dumps only the control blocks and trace table for the TCB that requested the dump. If you take a memory dump of the external CICS address space using a console command, the CICS IPCS VERBEXIT routine, DFHPD*nnn* formats the control blocks and trace tables for every TCB it finds in the dump.

For more information, see Formatting system dumps.

The EXCI service trap, DFHXCTRA

A user-replaceable program, DFHXCTRA, is available for use under the guidance of IBM Support. It is the equivalent of DFHTRAP used in CICS. It is invoked every time the external CICS interface (EXCI) writes a trace entry.

DFHXCTRA can perform one or all of the following actions:

1. Request the external CICS interface to write a trace entry on its behalf.
2. Instruct the external CICS interface to take an SDUMP.
3. Instruct the external CICS interface to skip writing the current trace entry to z/OS GTF.
4. Instruct the external CICS interface to disable DFHXCTRA.

The CICS-supplied sample version of DFHXCTRA performs all four 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 the `CICSTS`*nn*`.CICS.SDFHMAC` library, where `CICSTS`*nn* is your CICS release. For example, the library is CICSTS64.CICS.SDFHMAC for CICS TS beta. The parameter list passed to DFHXCTRA is defined in the copybook DFHXCTRD, which is supplied in the SDFHMAC library. DFHXCTRD also defines all the EXCI trace points for use by DFHXCTRA.

Problem determination with RRMS

Recoverable Resource Management Services (RRMS) is part of the z/OS operating system and comprises registration services, context services, and recoverable resource services (RRS). RRMS provides the context and unit of recovery management. RRMS is used to coordinate DPL requests, you can obtain additional problem determination information from RRMS.

RRS provides Interactive System Productivity Facility (ISPF) panels for you to work with RRS. To obtain debug information from RRMS, you can use ISPF dialogs as follows:

- You can browse the RRS log streams. RRS uses five log streams, including the resource manager data log, the restart log, the main UR state log, the delayed UR state log, and the archive log, that are shared by all the systems in a sysplex. For details about RRS log streams, see RRS log streams.
- You can display information about RRS resource managers.
- You can display information about RRS Units of Recovery.

For information about how to install and use the dialogs, see z/OS MVS Programming: Resource Recovery.

Chapter 6. Response and reason codes returned on EXCI calls

The EXCI call interface is part of the external CICS interface that enables a non-CICS program (a client program) running in z/OS to call a program (a server program) running in a CICS region and to pass and receive data by using a communications area or by using a channel and a set of containers. This reference gives details of the reason codes for the responses returned on the EXCI call interface.

See also The EXCI CALL interface.

Note: All numeric response and reason code values shown are in decimal.

Table 7. EXCI response codes (returned in response field of return_area)

Code	Meaning	Explanation
0	OK	For all EXCI CALL commands other than the DPL_request, the call was successful. If an OK response is received for a DPL_request, you must also check *dpl_retarea* to ensure CICS did not return a condition code. If the EIBRESP field of *Dpl_retarea* is zero, the DPL call was successful.
4	WARNING	The external CICS interface detected an error, but this did not stop the CALL command completing successfully. The reason code field describes the error detected.
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 z/OS 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.

Reason codes for response: 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. The reason codes for response

WARNING are 1 (PIPE_ALREADY_OPEN), 2 (PIPE_ALREADY_CLOSED), 3 (VERIFY_BLOCK_FM_ERROR), 4 (WS_FREEMAIN_ERROR), 5 (XCPIPE_FREEMAIN_ERROR), 6 (IRP_IOAREA_FM_FAILURE), 7 (SERVER_TERMINATED) and 8 (XFRASTG1_FM_FAILURE).

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.

3: VERIFY_BLOCK_FM_ERROR

Explanation
Initialize_User processing requires storage below 16MB to build the parameter list for the SSI Verify call, and an error has occurred during the **FREEMAIN** for this area.

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 memory dump of the batch region and use the memory dump, together with the return code from the z/OS **FREEMAIN**, to determine why the **FREEMAIN** is failing.

4: WS_FREEMAIN_ERROR

Explanation
An attempt to FREEMAIN working storage has resulted in a z/OS **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 memory dump of the batch region and use the memory dump, together with the return code from the z/OS **FREEMAIN** to determine why the **FREEMAIN** is failing.

5: XCPIPE_FREEMAIN_ERROR

Explanation
An attempt to FREEMAIN pipe storage has resulted in a z/OS **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 memory dump of the client application program address space, and use the memory dump, with the return code from the z/OS **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 a z/OS **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 memory dump of the batch region and use it with the return code from the z/OS **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.

8: XFRASTG1_FM_FAILURE

Explanation

An attempt to FREEMAIN the transmission area has resulted in a z/OS **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 XFRASTG1_FM_FAILURE is returned to your application only if the DPL request completes; otherwise, it is overwritten by subsequent response and reason codes.

User Response

If the problem persists, take a memory dump of the batch region and use it with the return code from the z/OS **FREEMAIN** to determine why the **FREEMAIN** is failing.

Reason codes for response: 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 command can be reissued without changing the client program after the environment error has been corrected. The reason codes for response RETRYABLE are 201 (NO_CICS_IRC_STARTED), 202 (NO_PIPE), 203 (NO_CICS), 204 (WRONG_MVS_FOR_RRMS), and 205 (RRMS_NOT_AVAILABLE).

201: NO_CICS_IRC_STARTED

Explanation

An Initialize_User command has been issued on a z/OS image that has had no IRC activity since the previous IPL, and the external CICS interface cannot determine the CICS SVC number.

System Action

The Initialize_User call fails, and the external CICS interface invokes the user-replaceable module, DFHXCURM.

User Response

Ensure that a CICS region in the z/OS image has logged on to IRC (that is, has started up with the system initialization parameter IRCSTRT=YES or has started IRC dynamically with an OPEN IRC command). Alternatively, if there is no local CICS region in the z/OS image, you must specify the SVC parameter that the external CICS interface is to use, by coding a CICSSVC parameter in the DFHXCOPT table. This situation can occur if you are using XCF to communicate to a CICS region in another z/OS image. Once the problem has been resolved, re-issue the Initialize_User request.

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 the subreason field-1 is non-zero (the IRP response code (R15)), the subreason field-2 contains the IRP reason code. For an explanation of the IRP return codes, see the interregion control blocks in the Data areas. The IRP return codes are in the DFHIRSPS copybook, listed under the heading IRC.

If the subreason code is 104 (hexadecimal 68), this code relates to IRERRNSS, which means that no secondary Connection Control Block (CCB) was found for the primary system. This indicates that the connection definition for the connecting job could not be found, typically because a generic EXCI connection definition has not being installed. To install a generic EXCI connection definition in the target CICS region, you can install the group DFH$EXCI to get the CICS-supplied generic EXCI connection definition. If your application program intends to use a specific pipe instead of a generic pipe, ensure that the **allocate_opts** parameter on Allocate_Pipe is set to X'00'. If **allocate_opts** is set to x'80', this instructs CICS to allocate a generic pipe.

The client program should deallocate the pipe. When you have corrected the problem, your client application program can reissue the allocate_pipe and Open_Pipe calls, or alternatively allocate and open a pipe to a different CICS region.

204: WRONG_MVS_FOR_RRMS

Explanation

A DPL request omitting the SYNCONRETURN option has been made specifying a CICS region that is on a different z/OS system from the batch program. Because the Recoverable Resource Management Services (RRMS) context is not recognized in the target system, the request is rejected.

System Action

The DPL request fails, and the external CICS interface invokes the user-replacable module, DFHXCURM.

User Response

Ensure that the batch program that issued the DPL request and the CICS region it was sent to are on the same z/OS system.

205: RRMS_NOT_AVAILABLE

Explanation

A DPL request omitting the SYNCONRETURN option has been made when the Resource Recovery Services (RRS) is not available.

There are two cases:

- When Resource Recovery Services (RRS) is not available.
- When Resource Recovery Services has restarted since the last DPL request omitting the SYNCONRETURN option, and there has been no intervening syncpoint.

Note: RRS is a part of Recoverable Resource Management Services (RRMS).

System Action

The DPL request fails, and the external CICS interface invokes the user-replacable module, DFHXCURM.

User Response

Retry the DPL request when Resource Recovery Services has restarted since the last DPL request omitting the SYNCONRETURN option, and there has been no intervening syncpoint.

Reason codes for response: USER_ERROR

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. 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. The reason codes for response USER_ERROR are 401 through 434.

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, 2, or 3.

System Action
The request is rejected.

User Response
Check the client application program and ensure that all EXCI calls specify the value of 1, 2, or 3 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 z/OS 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 z/OS return code, determine why the load failed. The most likely reason is that the message module, DFHMEBMX, is not in any library included in the STEPLIB of the batch job. Ensure that the `CICSTS`*nn*`.CICS.SDFHEXCI` library is included in the STEPLIB concatenation (where *nn* reflects the release of CICS: for example `CICSTS64.CICS.SDFHEXCI` for CICS TS beta. 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 z/OS 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 z/OS reason code, determine why the load failed. The most likely reason is that the message table, DFHMET4E, is not in any library included in the STEPLIB of the batch job. Ensure that the `CICSTS`*nn*`.CICS.SDFHEXCI` library is included in the STEPLIB concatenation (where *nn* reflects the release of CICS: for example `CICSTS64.CICS.SDFHEXCI` for CICS TS beta. 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 z/OS 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 z/OS reason code, determine why the load failed. The most likely reason is that module DFHXCURM is not in any library included in the STEPLIB of the batch job. Ensure the library containing the module is included in the STEPLIB concatenation, and restart the client application program.

413: DFHXCTRA_LOAD_FAILED

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 z/OS 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 z/OS reason code, determine why the load failed. The most likely reason is that DFHXCTRA is not in any library included in the STEPLIB of the batch job. Ensure the library containing the module is included in the STEPLIB concatenation, and restart the client application program.

414: IRP_ABORT_RECEIVED

Explanation

While processing a DPL request, an error occurred in the CICS server region, resulting in an abort FMH7 flow being returned to the external CICS interface.

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 does not support the external CICS interface.

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 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 TCB that has already been used by CICS. The external CICS interface cannot share a TCB 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 TCB (or daughter TCB), and issue the EXCI calls under that unique TCB.

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 z/OS 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 z/OS reason code, determine why the load failed. The most likely reason is that DFHXCOPT is not in any library included in the STEPLIB of the batch job. Correct the problem and restart the client application program.

421: RUNNING_UNDER_AN_IRB

Explanation
The EXCI call is issued under an z/OS IRB, which is not permitted.

System Action
The call is rejected.

User Response
Determine why the call was issued under an IRB and change the client application program.

422: SERVER_ABENDED

Explanation
While 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.

423: SURROGATE_CHECK_FAILED

Explanation
A DPL request has been issued specifying a USERID parameter. The userid specified was subject to a surrogate user check. The surrogate user check failed. The surrogate security check verifies whether the EXCI batch region's userid is authorized as a surrogate of the userid specified on the DPL call.

System Action
The DPL call is rejected. The RACF return code and reason code are returned in subreason field-1 and field-2. For RACF, these are documented in the z/OS Security Server RACROUTE Macro Reference.

User Response

Ensure that the EXCI batch region's userid has READ access to the profile *userid*.DFHEXCI in the SURROGAT general resource class, where *userid* is the userid specified on the DPL call.

See Surrogate user checking for more information.

424: RRMS_NOT_SUPPORTED

Explanation
A DPL request omitting the SYNCONRETURN option has been made on a system that is not running z/OS Release 5 (or a later, upward-compatible, release).

System Action
The call is rejected.

User Response
Ensure that the batch program is run on a system that is running the correct level of z/OS.

425: UOWID_NOT_ALLOWED

Explanation
A DPL request omitted the SYNCONRETURN option, but specified a value of UOWID. This combination of parameters is not permitted on a DPL request.

System Action
The DPL_Request is rejected.

User Response
Check the client application program and ensure that the correct combination of parameters is used on the DPL call.

426: INVALID_TRANSID2

Explanation
A DPL request has been issued with a *transid2* parameter that consists of all blanks.

System Action
The DPL request is rejected.

User Response
Check the client application program and ensure that the *transid2* parameter is specified correctly or has not been overwritten in some way.

427: INVALID_CCSID

Explanation
A DPL request has been issued with a *ccsid* parameter that specifies an invalid value.

System Action
The DPL request is rejected.

User Response
Check the client application program and ensure that the *ccsid* parameter is specified correctly or has not been overwritten in some way.

428: INVALID_ENDIAN

Explanation
A DPL request has been issued with a *endian* parameter that specifies an invalid value.

System Action
The DPL request is rejected.

User Response
Check the client application program and ensure that the *endian* parameter is specified correctly or has not been overwritten in some way.

429: DFHXCEIX_LOAD_FAILED

Explanation
During processing of an **EXEC CICS LINK** call, the external CICS interface attempted to load the module (DFHXCEIX). The load of this module has failed.

System Action
The **EXEC CICS LINK** call is rejected.

User Response
The most likely reason is that DFHXCEIX is not in any library included in the STEPLIB of the batch job. Ensure the library containing the module is included in the STEPLIB concatenation, and restart the client application program.

430: DFHXCPRX_LOAD_FAILED

Explanation
During Initialize_User processing, the external CICS interface attempted to load the module (DFHXCPRX). The load of this module has failed.

System Action
The Initialize_User call is rejected. The return code from the z/OS load macro (R15) is returned in the subreason field-1. The external CICS interface handles the error, and returns the abend (R1) that would have occurred in the subreason field-2.

User Response
Using the z/OS reason code, determine why the load failed. The most likely reason is that DFHXCPRX is not in any library included in the STEPLIB of the batch job. Ensure the library containing the module is included in the STEPLIB concatenation, and restart the client application program.

431: COMMAREA_LEN_NOT_ALLOWED

Explanation

A DPL request that specifies a **CHANNEL** parameter and a **COMMAREA_LEN** parameter has been issued.

When a channel is used to transfer data between programs, the **COMMAREA_LEN** parameter must be null.

System Action
The request is rejected.

User Response
Check your EXCI program. If it contains a DPL request that specifies a channel, ensure that the **COMMAREA_LEN** parameter is null.

432: DATA_LEN_NOT_ALLOWED

Explanation

A DPL request that specifies a **CHANNEL** parameter and a **DATA_LEN** parameter has been issued.

When a channel is used to transfer data between programs, the **DATA_LEN** parameter must be null.

System Action
The request is rejected.

User Response
Check your EXCI program. If it contains a DPL request that specifies a channel, ensure that the **DATA_LEN** parameter is null.

433: CCSID_NOT_ALLOWED

Explanation

A DPL request that specifies a **CHANNEL** parameter and a **CCSID** parameter has been issued.

When a channel is used to transfer data between programs, the **CCSID** parameter must be null.

System Action
The request is rejected.

User Response
Check your EXCI program. If it contains a DPL request that specifies a channel, ensure that the **CCSID** parameter is null.

Reason codes for response: 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. The reason codes for response SYSTEM_ERROR are 601 through 633.

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 z/OS abend (U0408) to force a SYSMDUMP. The return code from the z/OS GETMAIN request is returned in the return area.

User Response
Locate the GETMAIN return code in the dump, and use this and the rest of the dump to determine why the GETMAIN failed. A possible reason for this is that the region size specified for the job is too small. If this is the case, increase the region size and restart the client application program.

602: XCGLOBAL_GETMAIN_ERROR

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 z/OS abend (U0403) to force a SYSMDUMP. The return code from the z/OS GETMAIN request is returned in the return area.

User Response
Locate the GETMAIN return code in the dump, and use this and the rest of the dump to determine why the GETMAIN failed. A possible reason for this is that the region size specified for the job is too small. If this is the case, increase the region size and restart the client application program.

603: XCUSER_GETMAIN_ERROR

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 z/OS user abend (0410) to force a SYSMDUMP.

User Response
Use the return code from the GETMAIN, with the dump, to determine why the GETMAIN failed. A possible reason for this is that the region size of the job is too small. If this is the case, increase the region size and restart the client application program.

604: XCPIPE_GETMAIN_ERROR

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. A possible reason for this is that the region size for the job is too small. If this is the case, increase the region size and restart the client application program.

605: VERIFY_BLOCK_GM_ERROR

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 z/OS abend (U0409) to force a SYSMDUMP The return code from the z/OS GETMAIN request is returned in the return area.

User Response
Locate the GETMAIN return code in the dump, and use this and the rest of the dump to determine why the GETMAIN failed. A possible reason for this is that the region size specified for the job is too small. If this is the case, increase the region size and restart the client application program.

606: SSI_VERIFY_FAILED

Explanation
A VERIFY call to the MVS subsystem interface (SSI) to obtain the current CICS SVC number failed.

System Action
The Initialize_User request is terminated. The return code from the SSI call is returned in subreason field-1 of the return area. This error occurs before the external CICS interface dump services are initialized, therefore EXCI issues an z/OS user abend (0405) to force a SYSMDUMP.

User Response
Locate the return code in the dump, and use this with the rest of the dump and SSI documentation to determine why the VERIFY request failed. When the problem is resolved, restart the client application program.

607: CICS_SVC_CALL_FAILURE

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 z/OS user abend (0406) to force a SYSMDUMP.

User Response
Contact your IBM support center for assistance, with the return code and the dump available.

608: IRC_LOGON_FAILURE: XCPIPE_GETMAIN_ERROR

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 Data areas. 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 Data areas. 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 remains 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 Data areas. 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 Data areas. 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, while 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, while 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 while 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 while 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 z/OS 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 while processing the data returned by CICS, after the server application program has completed execution. This error results in the pipe being put into a "must close" state.

User Response
Use the return code to determine why the GETMAIN failed. A possible reason for this is that the region size of the job is too small. If this is the case, increase the region size and restart the batch job.

619: IRP_BAD_IOAREA

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 while 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 Data areas. The IRP return codes are in the DFHIRSPS copybook, listed under the heading IRC. Use the dump and IRP return codes to determine why the disconnect failed. You may also want to use the EXCI trace to determine the earlier error that caused the open pipe recovery routine to be invoked.

Note: The pipe is now in a "must close" state and if further calls are to be issued, the pipe must be closed and reopened again first.

622: ESTAE_SETUP_FAILURE

Explanation
To protect itself from possible program checks the external CICS interface establishes an z/OS ESTAE. In this case, the z/OS ESTAE macro has failed.

System Action
The call terminated, and the return code from the z/OS ESTAE command is returned in the EXCI subreason field-1. This error may occur before EXCI dump services are initialized, therefore an EXCI issues an z/OS abend (U0402) to force a SYSMDUMP.

User Response
Use the return code and the dump to determine why the ESTAE command failed. This may be an internal EXCI error and if the problem persists, contact your IBM support center.

623: ESTAE_INVOKED

Explanation
A program check is encountered during call processing, and the ESTAE is invoked.

System Action
The program check is handled by the EXCI ESTAE and an attempt is made to recover to a state that can support further EXCI calls. The z/OS abend code is returned in the EXCI subreason field-1 of the return area. To aid further diagnosis, a SYSMDUMP is taken.

User Response

Use the return code and the dump to determine why a program check occurred in the external CICS interface. The most likely reason for this is that the EXCI code abended while trying to access the client program's parameters. Use the EXCI trace to determine if any of the parameters might have caused this error. If this is not the case, this may be an error in the external CICS interface. Keep the dump and contact your IBM support center.

624: SERVER_TIMEDOUT

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 timeout 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 timeout, 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

To provide a TIMEOUT mechanism, the external CICS interface issues an z/OS 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 z/OS 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 z/OS STIMERM CANCEL command failed. This could be an external CICS interface error. Contact your IBM support center with details of the dump.

Note: The pipe is in a "must close" state after this error, and before attempting further calls must first be closed and reopened.

627: INCORRECT_SVC_LEVEL

Explanation

The release level of the CICS SVC (DFHCSVC) is not the same (or higher) than the release level of the external CICS interface.

System Action

The Initialize_User request is terminated. This error occurs before the external CICS interface SDUMP facilities are initialized, therefore EXCI issues an z/OS abend (U0407) to force a SYSMDUMP.

User Response

Determine the level of the CICS SVC being used and ensure it is the same release level as the external CICS interface, or higher. If the SVC number is allowed to default (CICSSVC=0 in DFHXCOPT), the SVC number being used is the SVC first used by a CICS region on the z/OS image. That is, the SVC used by the first CICS region to open the CICS interregion communications (IRC). If the SVC number is specified on CICSSVC in DFHXCOPT, the SVC number specified is at an incorrect level. For more information, see the description of the CICSSVC parameter in Using the EXCI options table, DFHXCOPT.

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 z/OS image. For more details about installing DFHIRP, see Installing the modules DFHIRP and DFHCSVC in the LPA in Installing.

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.

630: RRMS_ERROR

Explanation

An unexpected return code was received from Recoverable Resource Management Services (RRMS) while processing a DPL_Request.

System Action

DPL_Request processing is terminated. The value in subreason field-1 of the return area indicates which RRMS interface returned the unexpected return code:

1
CTXRCC

2
ATRRURD

3
CTXSDTA

The return code from the RRMS request is returned in subreason field-2. The external CICS interface issues message DFHEX0002, and takes a system dump.

User Response
Use the return code from the RRMS request and the dump, to determine why the request failed. This may be an internal EXCI error or a problem with RRMS and you may need the assistance of your IBM support center.

631: RRMS_SEVERE_ERROR

Explanation
During the processing of a DPL_Request, the EXCI code encountered an unexpected error while using its interface with Recoverable Resource Management Services (RRMS).

System Action
DPL_Request processing is terminated. The external CICS interface issues message DFHEX0002, and takes a system dump.

User Response
Use the dump, to determine why the request failed. This may be an internal EXCI error and you may need the assistance of your IBM support center.

632: XCGUR_GETMAIN_ERROR

Explanation
During DPL_Request processing, a GETMAIN request for working storage for module DFHXCGUR failed.

System Action
DPL_Request processing is terminated. The return code from the GETMAIN is returned in subreason field-1 of the return area. The external CICS interface issues message DFHEX0003, and takes a system dump.

User Response
Use the return code from the GETMAIN, and the dump, to determine why the GETMAIN failed. A possible reason is that the region size of the job is too small. If this is the case, increase the region size and restart the client application program.

633: INQUIRE_CHANNEL_FAILED

Explanation
During DPL_Request processing, an INQUIRE_CHANNEL request to obtain the channel token failed.

System Action
DPL_Request processing is terminated. The external CICS interface issues message DFHEX0002, and 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.

Chapter 7. EXCI samples: channel and containers sample applications

To show you how to code client applications that use both the **EXCI CALL** interface and **EXEC CICS LINK** command, CICS provides sample MVS client programs and a sample CICS server program. A set of EXCI channel and containers sample applications shows using channels and containers to pass data to and receive data from CICS.

About the EXCI channel and containers sample applications

The external CICS interface sample programs include two sample MVS client programs and a sample CICS server program. Data is passed between the clients and the server program using channels and containers.

Table 8. The external CICS interface sample programs

Language	Name	Type of program
Assembler[1]	DFH$AXNC	Client program
COBOL[2]	DFH0CXNC	Client program

Notes:

1. Assembler language programs are in source and executable form.
2. COBOL programs are provided in source form only.

The sample CICS server program

The sample CICS server program, DFH$AXNS, is provided in assembler only and is in source and executable form.

The sample MVS client program

The sample client program shows you how to code a simple MVS client application using the EXCI CALL interface and the **EXEC CICS LINK** command. The internal design of both client assembler and COBOL sample programs is the same.

Note: The assembler version of the client program uses BSAM, which requires the programs to be link-edited in RMODE(24), as a switch to AMODE(24) is made around the BSAM call. The assembler source code includes the required RMODE(24) statement. Normally, EXCI client programs run AMODE(31),RMODE(ANY). Therefore, the assembler version of the client program is unsuitable for use as Language Environment MAIN programs.

Each version of the client is divided into three separate sections as follows:

Section 1

Section 1 uses the EXEC interface to send containers to the server (CICS). The request is in a container called REQUEST_TYPE and will contain either "LINK1" or "LINK2". The server program (DFH£AXNS) uses this container. For the first LINK request, a container called EBCDIC_DATA is set up on channel FIRST_CHANNEL with a simple text string. This is then sent by an **EXEC CICS LINK** request.

If the request succeeds an **EXEC CICS QUERY CHANNEL** request is issued to check how many containers are on the channel. There should be three because the server program will have added an EXCI_RESPONSE container to the channel. A browse using **EXEC CICS STARTBROWSE CONTAINER**, **EXEC CICS GETNEXT CONTAINER** and **EXEC CICS ENDBROWSE CONTAINER** requests is then

performed to browse the names of the containers on the channel. This query and browse is not necessary but is added to demonstrate how to use the EXCI SPI commands.

The final part of section 1 involves issuing a number of container commands to set up container ASCII_DATA on channel SECOND_CHANNEL. This is sent again by an **EXEC CICS LINK** request.

Section 2
Section 2 uses the CALL interface. After **INIT_USER**, **ALLOCATE_PIPE**, and **OPEN_PIPE** have been executed, the DPL section is in a loop that executes twice. It repeats the sending of the containers sent in section 1 but uses the CALL interface instead.

In Section 2, the REQUEST_TYPE container will contain "CALL1" or "CALL2", not "LINK1" or "LINK2". The other containers will be as sent for Section 1.

Section 3
Section 3 deletes the channels used. This is not required, but is done to show best practise. Then **CLOSE_PIPE** and **DEALLOCATE_PIPE** are executed.

The server program, DFH£AXNS (in assembler) is invoked by both DFH£AXNC and DFH0CXNC. It uses the contents of the REQUEST_TYPE container to know whether this is the first or second invocation by either the EXEC or the CALL interface. If the container cannot be found, the program abends with abend code NCON, which indicates that the REQUEST_TYPE container could not be found. If this succeeds, the server program gets the appropriate container for the indicated request and returns a response in container EXCI_RESPONSE. This is tested by the client programs to make sure the response contains the text 'OK'. The program writes records to a TS main queue called DFHAXNSQ. This enables execution of the program to be confirmed and shows whether everything worked properly.

The assembler version of the client program is supplied pregenerated in an executable form. Both versions of the program accept two runtime parameters, as follows:

TARGET_SYSTEM
Specifies the server region APPLID.

If you use the pregenerated assembler version, you do not need to reassemble the program to specify the APPLID of your own CICS server region. You can also use the sample client programs with different CICS regions without needing to modify the programs each time.

USERID
Specifies the user ID to be used on the call interface DPL_request.

You specify these positional parameters on the PARM statement, separated by a comma.

Setting up the EXCI channel and containers sample programs

The sample external CICS interface programs are included on the CICS Transaction Server for z/OS distribution tape. Resource definitions that support the EXCI sample programs are included in the CICS system definition file (CSD) in groups DFH$EXCI.

About this task

The sample programs, shown in Table 8 on page 105, in source form in the *version*.CICS.SDFHSAMP library, where *version* is your version of CICS. For example, for CICS Transaction Server for z/OS, beta, this is CICSTS64.CICS.SDFHSAMP.

The sample assembler server program is also supplied in executable form in *version*.CICS.SDFHLOAD library, where *version* is your version of CICS. For example, for CICS Transaction Server for z/OS, beta, this is CICSTS64.CICS.SDFHLOAD.

The assembler client program is supplied in *version*.CICS.SDFHEXCI library, where *version* is your version of CICS. For example, for CICS Transaction Server for z/OS, beta, this is CICSTS64.CICS.SDFHEXCI.

Note: The resource definitions for the EXCI sample programs are included in the CSD but they are not included in the IBM-defined group list DFHLIST. 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.

Procedure

1. Install the following resource definition group:

   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$AXNS.

   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$EXCI specify ATTACHSEC(IDENTIFY). This security option causes the server program DFH$AXNS to fail with an ATCY abend if you run the sample programs in an environment that does not have RACF 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).
2. For transactions that are to be linked to from the batch program, specify the mirror program DFHMIRS as the program name in their transaction definitions.
3. 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.
4. If you want to use the COBOL version of the EXCI client program, translate, compile, and link-edit the program into a suitable library by using the DFHZXTCL or DFHYXTVL procedure.

Running the EXCI channel and containers sample applications

You can create a batch job to run the client program. You can also run the client program by using the pregenerated assembler version.

Before you begin

1. To use the COBOL version of the EXCI client program, you must translate, compile, and link-edit the program into a suitable library by using the DFHZXTCL ot DFHYXTVL procedure.
2. The resource definitions for the EXCI sample programs are included in the CSD but they are not included in the IBM-defined group list DFHLIST. 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.

Procedure

- Create a batch job to run the client program, based on the following sample JCL:

```
//EXCI    JOB (accounting_information),CLASS=A,TIME=1440,
//        USER=userid,PASSWORD=pswd,REGION=100M
//*============================================================*
//*    JCL to execute an external CICS interface client program   *
//*============================================================*
//          EXEC  PGM=pgmname,REGION=nnM,MEMLIMIT=nnG
//STEPLIB   DD    DSN=CICSTS54.CICS.EXCI.LOADLIB,DISP=SHR
//          DD    DSN=CICSTS54.CICS.SDFHEXCI,DISP=SHR
//SYSPRINT  DD    SYSOUT=A
//SYSMDUMP  DD    DSN=SYS1.SYSMDP00,VOL=SER=volid,SPACE=(CYL,(1,1)),
//                DISP=OLD,UNIT=3390
```

Figure 14. Sample job for starting an EXCI client program

- If you want to use the pregenerated assembler version of the client program, issue the following EXEC statement for the client program:

```
//*============================================================*
//ASM       EXEC  PGM=client_program_name,PARM='applid,userid',REGION=0M,MEMLIMIT=1G
```

where:

client_program_name
Specify the name of the client program, for example, DFH$AXNC.

applid
Specify the APPLID of your target CICS server region.

Note: If you omit *applid*, you must keep the comma that precedes the user ID.

userid
Specify the user ID for the DPL_request call.

Results

Figure 15 on page 108 shows an example of the output from DFH$AXNC if the pregenerated assembler version of the client program DFH$AXNC is executed successfully.

```
*==================== EXCI Sample Batch Client Program ======================*
*                                                                            *
*     Parameters: APPLID=IYK2Z2G1 .......... .......                         *
*                                                                            *
*  EXEC Level Processor.                                                     *
*    Setting up the EXEC level call.                                         *
*    The Link Request with channel FIRST_CHANNEL has completed successfully.   *
*    Checking response container sent by server.                             *
*    Response OK, continue processing.                                       *
*    Query channel command completed successfully.                           *
*    Correct number of containers returned.                                  *
*    Browse of channel names completed successfully.                         *
*    The Link Request with channel SECOND_CHANNEL has completed successfully.  *
*    Checking response container sent by server.                             *
*    Response OK, continue processing.                                       *
*                                                                            *
*  CALL Level Processor.                                                     *
*    Initialise_User call complete.                                          *
*    Allocate_Pipe call complete.                                            *
*    Open_Pipe call complete.                                                *
*    The connection has been successful.                                     *
*    Container EBCDIC_DATA was received correctly in channel FIRST_CHANNEL.    *
*    Container ASCII_DATA was received correctly in channel SECOND_CHANNEL.    *
*    Channels have been deleted.                                             *
*    Close_Pipe call complete.                                               *
*    Deallocate_Pipe call complete.                                          *
*                                                                            *
*================== End of EXCI Sample Batch Client Program =================*
```

Figure 15. Example output from successful execution of DFH$AXNC

Notices

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

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

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

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

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

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

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

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

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

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

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

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

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Client Relationship Agreement, IBM International Programming License Agreement, or any equivalent agreement between us.

The performance data discussed herein is presented as derived under specific operating conditions. Actual results may vary.

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

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

COPYRIGHT LICENSE:

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

Beta content

Content described as “beta” relates to a release of the program prior to it being made commercially available that may still be under development and therefore, potentially unreliable. Beta content may change or be removed, is not intended for production use, and is provided as-is, without liability, warranty or support, to the full extent permitted by applicable law.

Statements by IBM regarding its plans, directions, and intent are subject to change or withdrawal without notice at the sole discretion of IBM. Information regarding potential future products is intended to outline general product direction and should not be relied on in making a purchasing decision. The information mentioned regarding potential future products is not a commitment, promise, or legal obligation to deliver any material, code, or functionality. Information about potential future products may not be incorporated into any contract. The development, release, and timing of any future features or functionality described for IBM products remain at the sole discretion of IBM.

Licensed Program Specifications (LPS)

Where can I find license terms for Monthly License Charge (MLC)?

For more information about the license terms for MLC, see:

- 6.3 IBM CICS Transaction Server Licensed Program Specifications 6.3
- 6.2 IBM CICS Transaction Server Licensed Program Specifications 6.2
- 6.1 IBM CICS Transaction Server Licensed Program Specifications 6.1

Where can I find license terms for Value Unit Edition (VUE)?

For more information about the license terms for VUE, see:

- 6.3 License Information terms and conditions for Value Unit Edition 6.3
- 6.2 License Information terms and conditions for Value Unit Edition 6.2
- 6.1 License Information terms and conditions for Value Unit Edition 6.1

Programming interface information

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

Programming Interfaces that allow the customer to write programs to obtain the services of CICS Transaction Server for z/OS, Version 6 are included in the following sections of the online product documentation:

- Developing applications
- Developing system programs
- Securing CICS
- Developing for external interfaces
- Application development reference
- Reference: system programming
- Reference: connectivity

Information that is NOT intended to be used as a Programming Interface of CICS Transaction Server for z/OS, Version 6, but that might be misconstrued as Programming Interfaces, is included in the following sections of the online product documentation:

- Troubleshooting and support
- CICS TS diagnostics reference

If you access the CICS documentation in manuals in PDF format, Programming Interfaces that allow the customer to write programs to obtain the services of CICS Transaction Server for z/OS, Version 6 are included in the following manuals:

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

If you access the CICS documentation in manuals in PDF format, information that is NOT intended to be used as a Programming Interface of CICS Transaction Server for z/OS, Version 6, but that might be misconstrued as Programming Interfaces, is included in the following manuals:

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

Trademarks

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

Apache, Apache Axis2, Apache Maven, Apache Ivy, the Apache Software Foundation (ASF) logo, and the ASF feather logo are trademarks of Apache Software Foundation.

Gradle and the Gradlephant logo are registered trademark of Gradle, Inc. and its subsidiaries in the United States and/or other countries.

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

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

The registered trademark Linux® is used pursuant to a sublicense from the Linux Foundation, the exclusive licensee of Linus Torvalds, owner of the mark on a worldwide basis.

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

Red Hat®, and Hibernate® are trademarks or registered trademarks of Red Hat, Inc. or its subsidiaries in the United States and other countries.

Spring Boot is a trademark of Pivotal Software, Inc. in the United States and other countries.

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

Zowe™, the Zowe logo and the Open Mainframe Project™ are trademarks of The Linux Foundation.

The Stack Exchange name and logos are trademarks of Stack Exchange Inc.

Red Hat, JBoss, OpenShift, Fedora, Hibernate, Ansible, CloudForms, RHCA, RHCE, RHCSA, Ceph, and Gluster are trademarks or registered trademarks of Red Hat, Inc. or its subsidiaries in the United States and other countries.

Terms and conditions for product documentation

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

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

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

Commercial use
You may reproduce, distribute and display these publications solely within your enterprise provided that all proprietary notices are preserved. You may not make derivative works of these publications,

or reproduce, distribute or display these publications or any portion thereof outside your enterprise, without the express consent of IBM.

Rights

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

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

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

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

IBM online privacy statement

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

For the CICSPlex SM Web User Interface (main interface):

Depending upon the configurations deployed, this Software Offering may use session and persistent cookies that collect each user's user name and other PII for purposes of session management, authentication, enhanced user usability, or other usage tracking or functional purposes. These cookies cannot be disabled.

For the CICSPlex SM Web User Interface (data interface):

Depending upon the configurations deployed, this Software Offering may use session cookies that collect each user's user name and other PII for purposes of session management, authentication, or other usage tracking or functional purposes. These cookies cannot be disabled.

For the CICSPlex SM Web User Interface ("hello world" page):

Depending upon the configurations deployed, this Software Offering may use session cookies that do not collect PII. These cookies cannot be disabled.

For CICS Explorer®:

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

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

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

Index

Special Characters

Numerics

A

B

C

D

E

F

G

I

J

L

M

N

O

P

Q

R

S

T

U

V

X