

BY MICHAEL H. CARROLL

CICS/ESA V4R1: External CICS Interface — Part II

While the EXCI call interface is more difficult to program, the flexibility and performance aspects of it contrast with the easier but stricter EXEC CICS interface.

AS I stated in Part I, (*Technical Support*, May 1998), the external CICS interface (EXCI) is an application programming interface (API) that enables a non-CICS program running in MVS (a client program) to call a program running in a CICS/ESA V4R1 region (a server program) and to pass and receive data by means of a communications area. The CICS application program is invoked as if linked to by another CICS application program (EXEC CICS LINK).

Part I examined the system interface and one element of the programming interface to EXCI. The EXEC CICS LINK command is the easy way to access CICS from an external client program. It does, however, have a couple of drawbacks. First, it is expensive in terms of establishing a connection to the CICS server on every invocation (it performs all six EXCI commands every time). Second, this method only works with GENERIC connections. While the EXCI call interface is more difficult to program, the flexibility and performance aspects of it contrast with the easier but stricter EXEC CICS interface.

THE LESS EASY WAY

I could have titled this section the “More Difficult Way” but this would deny the flexibility and performance aspects of this method. To use the EXCI call interface, the programmer must code at least five set-up and shutdown calls along with one or more DPL calls to connect to, use and disconnect from the server program. The six calls are shown in Figure 1 and must be invoked in the order shown.

The format of each call is dependent on the language used and the number of parameters that need to be passed. The program

called is DFHXCIS and when compiled the program must also be linked with stub module DFHXCSTB as was the EXEC CICS interface. Again, the program must also be linked AMODE(31). Examples of the various calls can be found in the sample programs listed in Part I.

The first four parameters on all calls are standard. Figure 2 outlines these parameters for an example COBOL INITIALISE_USER call. IBM supplies a number of copybooks for different programming languages to map some of these parameters and provide data areas for them. The copybooks are listed in Figure 3. The important elements in using this methodology can be summarized as follows:

- ◆ An INITIALISE_USER and ALLOCATE_PIPE can be used at the start of an EXCI session.
- ◆ AN OPEN_PIPE, one or more DPL_REQUESTs and a CLOSE_PIPE can be repeated at appropriate intervals.
- ◆ A DEALLOCATE_PIPE call can be used to terminate the EXCI session.

Based on this summary, a dedicated subroutine could be used to allow multiple non-CICS processes to have access to a CICS application. This has the advantage of curtailing the set-up overhead for each DPL call as highlighted for the EXEC CICS LINK interface. Additionally, both GENERIC and SPECIFIC EXCI connections are supported. Full details of all parameters and how they should be used can be found in the *CICS/ESA External CICS Interface V4R1* manual.

One particular aspect of parameter passing that may cause some difficulty is where DPL calls allow a “NULL” parameter to be passed. This is usually where the parameter is not required or you wish the default for this parameter to be used. C/C++ programmers will be familiar with the concept of passing a NULL pointer to a function or subroutine, which is essentially what I’m talking about here. Figure 4 provides an example of how to issue an EXCI DPL call with the userid and uowid parameters omitted in a COBOL program using a null parameter.

ERROR PROCESSING AND RECOVERY

After each call to EXCI, whether using the EXCI call or the EXEC CICS LINK interface, the five-word return area contains information on the status (success or otherwise) of the call. The layout of this area is as follows:

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

The DFHXCPLx copybook provides a standard return area for use with various program languages.

Once again, IBM supplies copybooks with equates for all the possible return codes that the interface can return. These copybooks are detailed in Figure 5. Possible returned values for the response field are highlighted in Figure 6. Generally speaking, a zero return code indicates success, except for a DPL_REQUEST call when the DPL return codes should be checked also to ensure satisfactory completion.

SECURITY

CICS applies security checks in a number of ways against requests received from an

Figure 1: EXCI Call Commands

EXCI Call	Number of Parameters	Description
INITIALISE_USER	5	Initialize the user environment, including obtaining authority to use IRC facilities.
ALLOCATE_PIPE	7	Allocate a single session, or pipe, to a CICS region.
OPEN_PIPE	5	Cause IRC to connect an allocated pipe to a receive session of the appropriate connection defined in the CICS region named on the ALLOCATE_PIPE command.
DPL_REQUEST	14	Issue a distributed program link request across an open pipe connected to the CICS system on which the server (or target) application program resides. More than one of these calls can be issued consecutively.
CLOSE_PIPE	5	Disconnect an open pipe from CICS. The pipe remains in an allocated state, and its tokens remain valid for use by the same user.
DEALLOCATE_PIPE	5	Deallocate a pipe from CICS. On completion of this command, the pipe can no longer be used, and its associated tokens are invalid.

MVS client program. These are:

- ◆ MRO logon and connect security, performed by DFHIRP
- ◆ link security, performed by the CICS server region
- ◆ user security checking in the server application program

MRO Logon

DFHIRP, the CICS interregion communication program, performs security checks against users who want to log on to IRP (specific connections only) and connect to a CICS region.

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

To enable your client program to logon successfully to IRP and to connect to the target server region, you must ensure that:

1. The batch region’s userid is defined as a user profile to your security package.
2. The batch region’s userid is authorized to its own DFHAPPL.batch_user_name RACF FACILITY class profile(s) with UPDATE authority. Use the appropriate mechanism in any other security package you may be using (e.g., ACF2).
3. The batch region’s userid is authorized to the DFHAPPL.applid RACF FACILITY class profile of the target CICS server region with READ authority.

Link Security

The target CICS server region performs link security checking against requests from the client program. These security checks cover transaction attach security (when attaching the mirror transaction) and resource and command security checking within the server application program. The link userid that CICS uses

Figure 2: First Four Parameters on All EXCI Calls

```
e.g., INITIALISE_USER call in COBOL:
CALL 'DFHXCIS' USING    VERSION-1
                        EXCI-RETURN-CODE
                        USER-TOKEN
                        INIT-USER
                        APPLICATION.
```

Parameter	Input/Output	Size	Description
VERSION-1	I	Full-word binary	Indicates the version of the external CICS interface parameter list being used. It must be set to 1. The DFHXCPLx copybook has a predefined field for this value.
EXCI-RETURN-CODE	O	5 words	Area to receive response and reason codes, and a message pointer field.
USER-TOKEN	O	1 word	Token returned on the Initialise_User command. Must be used on all subsequent calls for this session.
call-type	I	1 word	Area indicating the function of the command. DFHXCPLx copybook supplies values for all call-types e.g., INIT-USER.

Figure 4: Example of Null Parameter Passing

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
-----
version_number    01 VERSION-1          PIC S9(8) COMP VALUE 1.
return_area      01 RETAREA.           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 "DFH$AXCS".
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 DPL-RETAREA.        structure
dpl_opts        01 SYNCONRETURN       PIC X VALUE X"80".
```

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

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

Using the data names specified in WORKING-STORAGE SECTION as described above, and the NULL-PTR name as described in the LINKAGE SECTION, the following invocation of the DPL function omits the uowid and the userid parameters, and replaces them in the parameter list with the NULL-PTR variable:

```
DPL-SECTION.
*
  SET ADDRESS OF NULL-PTR TO NULLS.
*
CALL 'DFHXCIS' USING  VERSION-1      RETAREA      USER-TOKEN
                     DPL-REQUEST    PIPE-TOKEN    TARGET-PROGRAM
                     COMMAREA      COMM-LENGTH   DATA-LENGTH
                     TARGET-TRANSID NULL-PTR     NULL-PTR
                     DPL-RETAREA    SYNCONRETURN.
```

Figure 3: Copybooks Supplied for EXCI Programs

Copybook name	Language
DFHXCPLD	Assembler
DFHXCPLH	C
DFHXCPLP	COBOL
DFHXCPLL	PL/I

To use the EXCI call interface, the programmer must code at least five set-up and shutdown calls along with one or more DPL calls to connect to, use and disconnect from the server program.

for these security checks is the batch region's userid.

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

- ◆ the mirror transaction, usually CSMI
- ◆ the server program and any files, queues or other programs it accesses
- ◆ any systems programming interface commands that the server program might invoke

User Security

The target CICS server region performs user security checking against the userid passed on a DPL CALL request. User security checking is performed only when connections specify ATTACHCSEC(IDENTIFY). User security is performed in addition to any link security.

For user security, in addition to any authorizations you make for link security, you must also authorize the userid specified on the DPL CALL request. Note that there is no provision for specifying a userid on the EXEC CICS LINK command. In this case, the external CICS interface passes the batch region's userid. For further information check your security package documentation on CICS security.


Figure 5: Copybooks Supplied for EXCI Return Codes

Copybook name	Language
DFHXCRCD	Assembler
DFHXCRCH	C
DFHXCRCO	COBOL
DFHXCRCL	PL/I

SUMMARY

I hope this two-part review has helped you understand the concepts behind CICS V4R1's External CICS Interface. Perhaps you already have an application in mind for EXCI. It may seem daunting, but I can assure you that once you've grasped the initial understanding of how this mechanism works it will be just like writing another batch and/or CICS program! Have fun!

REFERENCES

- IBM-supplied sample EXCI programs shipped with CICS V4R1
- CICS/ESA External CICS Interface Version 4, Release 1
- Document Number SC33-1390-00
- CICS/ESA Application Programming Guide Version 4, Release 1
- Document Number SC33-1169-00 

NaSPA member Michael H. Carroll has 19 years of experience in the data processing industry. He's worked in both manufacturing and financial businesses, performing systems and applications programming for mainframe and client/server environments. Michael is a consultant currently working on Y2K projects for a large financial institution in Ireland.

©1998 Technical Enterprises, Inc. For reprints of this document contact sales@naspa.net.

Figure 6: Standard Return Codes From EXCI Invocations

Response	Meaning	Explanation
0	OK	For all EXCI commands, other than DPL_REQUEST, the command was successful. If OK is returned for a DPL_REQUEST, then the DPL return area must also be checked.
4	WARNING	EXCI detected an error but the command completed successfully. Check the reason code field for information about the error.
8	RETRYABLE	The EXCI call failed. This error usually indicates a problem with the environment not EXCI or the server program. Reason code field given describes the error detected.
12	USER_ERROR	The EXCI command has failed. Usually a problem in the server region or server program (e.g., a security check failure or an abend). Reason code gives details.
16	SYSTEM_ERROR	The EXCI command has failed. EXCI itself has detected an error. Again reason code contains details.