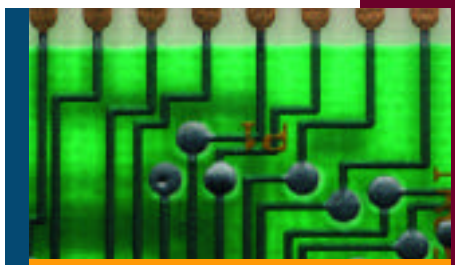
CICS/ESA V4R1: The External CICS Interface — Part I



BY MICHAEL H. CARROLL

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 4.1 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 another CICS application program (EXEC CICS LINK).

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

A *pipe* is a one-way communication path between a sending process and a receiving process. In an external CICS interface implementation, the client program represents the sending process and the CICS server region represents the receiving process.

Unless the CICS region is running in a sysplex under MVS/ESA 5.1 and therefore is able to use cross-system MRO (XCF/MRO), the client program and the CICS server region (the region where the server program runs or is defined) must be in the same MVS image. Although the external CICS interface does not support the cross-memory access method, it can use the XCF access method provided by XCF/MRO in CICS/ESA 4.1. See the *CICS/ESA Intercommunication Guide* for information about XCF/MRO.

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

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

THE SYSTEM INTERFACE

To use EXCI, special set-up of an MRO connection definition is required. A systems programmer can achieve this by using CEDA to define the resources required.

Figure 1 shows a typical CONNECTION definition for an EXCI link and Figure 2 shows the corresponding partial SESSION definition. These need a little further explanation.

First, both definitions should be defined in their own group. Secondly, the Connection name should be unique and not conflict with any existing CICS system id. In Figures 1 and 2 I avoid this conflict by using my own name!

Figure 1: CEDA Define CONNECTION

```
OVERTYPE TO MODIFY                                          CICS RELEASE = 0410
   CEDA  ALter Connection( MIKE )
     Connection      : MIKE
     Group           : EXCIGRP
     DEscription  ==> SAMPLE EXCI CONNECTION
   CONNECTION IDENTIFIERS
     Netname      ==>
     INDsys       ==>
   REMOTE ATTRIBUTES
     REMOTESYSTem ==>
     REMOTEName   ==>
     REMOTESYSNet ==>
   CONNECTION PROPERTIES
     ACcessmethod ==>    IRc             Vtam | IRc | INdirect | Xm
     PRotocol     ==>    Exci            Appc | Lu61 | Exci
     Conntype     ==>    Generic         Generic | Specific
     SInglesess   ==>    No              No | Yes
     DAtastream   ==>    User            User | 3270 | SCs | STrfield | Lms
     RECordformat ==>    U               U | Vb
     Queuelimit   ==>    No              No | 0-9999
     Maxqtime     ==>    No              No | 0-9999
   OPERATIONAL PROPERTIES
     AUtoconnect  ==>    No              No | Yes | All
     INService    ==>    Yes             Yes | No
   SECURITY
     SEcurityname ==>
     ATtachsec    ==>    Local           Local | Identify | Verify | Persistent
                                         | Mixidpe
     BINDPassword    :                   PASSWORD NOT SPECIFIED
     BINDSecurity ==>    No              No | Yes
     Usedfltuser  ==>    No              No | Yes
   RECOVERY
     PSrecovery   ==>                    Sysdefault | None
```

Figure 2: CEDA Define SESSION

```
OVERTYPE TO MODIFY                                          CICS RELEASE = 0410
   CEDA  ALter Sessions(PIPES)
     Sessions        : PIPES
     Group           : EXCIGRP
     DEscription  ==> PIPES FOR CONNECTION MIKE
   SESSION IDENTIFIERS
     Connection   ==>    MIKE
     SESSName     ==>
     NETnameq     ==>
     MOdename     ==>
   SESSION PROPERTIES
     Protocol     ==>    Exci                Appc | Lu61 | Exci
     MAximum      ==>    000 , 000           0-999
     RECEIVEPfx   ==>    <
     RECEIVECount ==>    005                 1-999
     SENDPfx      ==>
     SENDCount    ==>                        1-999
     SENDSize     ==>    04096               1-30720
+    RECEIVESize  ==>    04096               1-30720

     ..............
```

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.

The critical elements in the CONNECTION definition are as follows:

- Accessmethod - must equal IRC.
- Protocol - set to EXCI.
- Conntype - GENERIC or SPECIFIC.
- Netname - only set if Conntype is SPECIFIC.

In the example I've set the Conntype to GENERIC. A generic connection is an MRO link with a number of sessions to be shared by multiple EXCI users. For a generic connection you cannot specify the NETNAME attribute. **Note:** You must install only one generic EXCI connection in an individual CICS region.

On the other hand, a SPECIFIC connection is an MRO link with one or more sessions dedicated to a single user in a client program. This restricts the link to one user client program only.

The SESSION definition effectively defines the number of EXCI pipes our CONNECTION supports. The important attributes are:

- Connection - must be the same as specified in CONNECTION definition.
- Protocol - must be set to EXCI.
- Receivecount - this sets the maximum number of pipes available.
- Sendcount - must be blank for an EXCI session.

The maximum number of pipes (Receivecount) is 25 for all connection types and only one GENERIC connection can be defined per CICS system. There is no such restriction for SPECIFIC connection types and a mixture of one generic with one or more specific connections is allowed within a single CICS region.

Having defined the connection and session definitions, these can be installed using CEDA manually or automatically at CICS start-up (add group to start-up list). For the interface to be active, the IRC must be open (use CEMT S IRC OPE).

To obtain information about batch jobs linked to CICS through MRO, you use the CEMT INQUIRE IRBATCH command. This command enables you to identify the names of batch jobs currently connected to CICS through the interregion communication (IRC) facility. For jobs using the external CICS interface, the identification consists of

```
jobname.stepname.procname - mvsid
```

Either *stepname* or *procname* or both may not be present, indicated by the periods (.) being adjacent to one another. The *mvsid* identifies the MVS system on which the job is running. If XCF/MRO is in use, the job can reside on a different MVS image to that on which CICS is running.

Program definitions for EXCI targets (server programs) must be established in RDO also. When an EXCI client program links to a CICS program, it does so indirectly. Firstly, it invokes IBM supplied module DFHMIRS under transaction CSMI which

Figure 3: Standard JCL Procedures For EXCI Client Programs

JCL Procedure	Description
DFHEXTAL	the assembler procedure for assembler versions of client programs
DFHEXTDL	the C procedure for C versions of client programs
DFHEXTPL	the PL/I procedure for PL/I versions of client programs
DFHEXTVL	the COBOL procedure for VS COBOL II versions of client programs
DFHYXTDL	the procedure for C versions of client programs running under Language Environment/370
DFHYXTPL	the procedure for PL/I versions of client programs running under Language Environment/370
DFHYXTVL	the procedure for VS COBOL II versions of client programs running under Language Environment/370

then LINKs to the server program defined in CICS. There are no special requirements on the RDO definition of such server programs. They can be written in COBOL, PL/1, C or Assembler and are restricted to the DPL subset of the CICS application programming interface. See the *CICS V4R1 Applications Programming Guide* for details.

THE PROGRAMMING INTERFACE

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

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

- allocate and open sessions to a CICS system from non-CICS programs running under MVS/ESA
- issue DPL requests on these sessions from the non-CICS programs
- close and deallocate the sessions on completion of the DPL requests

The six EXCI commands are as follows:

1. Initialize User
2. Allocate_Pipe
3. Open_Pipe
4. DPL call
5. Close_Pipe
6. Deallocate_Pipe

The EXEC CICS interface provides a single, composite command — EXEC CICS LINK PROGRAM — that performs all six commands of the EXCI CALL interface in one invocation. This command takes the same form as the distributed program link command of the CICS command-level API. This, however, can only be used with generic connections.

There are sample programs supplied by IBM as part of CICS V4R1 that show the use of the EXEC CICS LINK version of EXCI. Sample client and server programs are supplied in a number of languages.

WHICH INTERFACE?

Given the two methods of programming EXCI, which type is appropriate? IBM's guidelines on this matter are fairly clear.

For low-frequency or single DPL requests, you are recommended to use the EXEC CICS LINK command. It is easier to code and therefore less prone to programming errors. Note that each invocation of an EXEC CICS LINK command causes the external CICS interface to perform all the functions of the CALL interface, which results in unnecessary overhead.

For multiple or frequent DPL requests from the same client program, you are recommended to use the EXCI CALL interface. This is more efficient, because you need only perform the Initialize_User and Allocate_Pipe commands once, at or near the beginning of your program, and the Deallocate_Pipe once on completion of all DPL activity. In between these functions, you can open and close the pipe as necessary, and while the pipe is opened, you can issue as many DPL calls as you want.

Therefore, if you are employing a heavy use, batch-type external interface use the EXCI CALL mechanism, while for low-frequency access, use the EXEC CICS LINK methodology.

Figure 4: EXCI Client EXEC CICS LINK Command

```
EXEC CICS LINK   PROGRAM(name)
                 RETCODE(data-area)
                 LENGTH(data-value)
                 DATALENGTH(data-value)
                 APPLID(name)
                 TRANSID('CSMI')
                 COMMAREA(data-area)
                 SYNCONRETURN
```

THE EASY WAY

The simplest way to establish a connection between a non-CICS client program and the CICS server program is via the EXEC CICS LINK command.

How can we invoke EXEC CICS LINK in a batch program? Well, it's quite simple really, the program has to be translated before compilation similar to a real CICS program so the commands are changed to the correct call sequences. The program must also be linked with a special stub program, DFHXCSTB and be AMODE(31). IBM supplies standard JCL procedures for translating, compiling and linking batch programs that will be used as EXCI clients. These procedures are outlined in Figure 3.

The EXEC CICS LINK command itself is slightly different to the normal command as it must conform with the DPL format required by RDO. The command is outlined in Figure 4. It assumes we are using a session we defined earlier with RDO. Some of the more important parameters are as follows:

- **APPLID** — specifies the Applid of the target CICS server region.
- **COMMAREA(data-area)** — specifies the communication area that is being made available to the invoked program.
- **DATALENGTH(data-value)** — specifies a halfword binary value that

Figure 5: Complete Example of EXEC CICS LINK

```
        WORKING-STORAGE SECTION.
      *
      * Declare EXEC level return code areas
      *
        COPY DFHXCPLO
      *
      *
      *
      01   TARGET-PROGRAM        PIC X(8) VALUE 'CSERVER'.
      01   TARGET-SYSTEM     PIC X(8) VALUE 'TESTCICS'.
      01   TARGET-TRANSID        PIC X(4) VALUE 'CSMI'.
      01   COMMAREA              PIC X(80).
      01   LINK-COM-LEN          PIC S9(4) COMP VALUE 80.
      01   LINK-DATE-LEN         PIC S9(8) COMP VALUE 4.

      .....

      PROCEDURE DIVISION.

      .....

           EXEC CICS LINK        PROGRAM(TARGET-PROGRAM)
                                 RETCODE(EXCI-EXEC-RETURN-CODE)
                                 LENGTH(LINK-COM-LEN)
                                 DATALENGTH(LINK-DAT-LEN)
                                 APPLID(TARGET-SYSTEM)
                                 TRANSID(TARGET-TRANID)
                                 COMMAREA(COMMAREA)
                                 SYNCONRETURN
           END-EXEC.
```

Figure 6: Sample Programs Supplied by IBM

The external CICS interface sample programs		
Language	**Name**	**Type of program**
Assembler	DFH$AXCC	Client program
Assembler	DFH$AXCS	Server program
COBOL	DFH0CXCC	Client program
PL/I	DFH$PXCC	Client program
C/370	DFH$DXCC	Client program

is the length of a contiguous area of storage, from the start of the COMMAREA, to be passed to the invoked program. If the amount of data being passed in a COM-MAREA is small, but the COMMAREA itself is large so that the linked-to program can return the requested data, you should specify DATALENGTH in the interest of performance.

- **LENGTH(data-value)** — specifies a halfword binary value that is the length in bytes of the communication area. Note that for "normal" EXEC CICS LINK commands fullword binary values are the norm, not halfword binary as is the case for EXCI.

- **PROGRAM(name)** — specifies the program name (one to eight 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.

- **RETCODE** — specifies a 20-byte area into which the external CICS interface places return code information. This area is formatted into five one-word fields and will be explained in Part II. A COBOL copybook (DFHXCPLO) is available to map this area.

- **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.

- **SYNCONRETURN** — specifies that the CICS server region, named on the APPLID parameter, is to take a syncpoint on successful completion of the server program. SYNCONRETURN is mandatory for an external CICS interface LINK command.

As you can see, this is not much different than a normal EXEC CICS LINK command. All data is passed to the CICS server program via the communications area and any data is passed back via this area. Figure 5 presents a complete piece of COBOL code that shows the data areas and command used correctly.

There are sample programs supplied by IBM as part of CICS V4R1 that show the use of the EXEC CICS LINK version of EXCI. Sample client and server programs are supplied in a number of languages. These are detailed in Figure 6.

NEXT TIME

In Part II, I will examine the EXCI CALL interface and the methods of error handling and recovery for the external CICS interface. Join me then!

REFERENCES

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.