Scenarios and Programmer Information

This chapter covers the following topics:

- COBOL Scenarios
- PL/I Scenarios
- Aborting RPC Server Customer Code and Returning Error to RPC Client
- Automatic Syncpoint Handling

COBOL Scenarios

- Scenario I: Calling an Existing COBOL Server
- Scenario II: Writing a New COBOL Server

Scenario I: Calling an Existing COBOL Server

To call an existing COBOL server

1. Use the *IDL Extractor for COBOL* to extract the Software AG IDL and, depending on the complexity, also a server mapping file. See *When is a Server Mapping File Required?* in the EntireX Workbench documentation.
2. Build an EntireX RPC client using any EntireX wrapper. See *EntireX Wrappers*. For a quick test you can:
   - use the IDL Tester; see *EntireX IDL Tester* in the EntireX Workbench documentation
   - generate an XML mapping file (XMM) and use the XML Tester for verification; see *EntireX XML Tester* in the XML/SOAP Wrapper documentation

See *Client and Server Examples for z/OS CICS* in the COBOL Wrapper documentation for COBOL RPC Server examples.

Scenario II: Writing a New COBOL Server

To write a new COBOL server

1. Use the *COBOL Wrapper* to generate a COBOL server skeleton and, depending on the complexity, also a server mapping file. See *When is a Server Mapping File Required?* in the EntireX Workbench documentation. Write your COBOL server and proceed as described under *Using the COBOL Wrapper for the Server Side*.
2. Build an EntireX RPC client using any EntireX wrapper. See *EntireX Wrappers*. For a quick test you can:

- use the IDL Tester; see *EntireX IDL Tester* in the EntireX Workbench documentation
- generate an XML mapping file (XMM) and use the XML Tester for verification; see *EntireX XML Tester* in the XML/SOAP Wrapper documentation

See *Client and Server Examples for z/OS CICS* in the COBOL Wrapper documentation for COBOL RPC Server examples.

PL/I Scenarios

- Scenario III: Calling an Existing PL/I Server
- Scenario IV: Writing a New PL/I Server

Scenario III: Calling an Existing PL/I Server

To call an existing PL/I server

1. Use the *IDL Extractor for PL/I* to extract the Software AG IDL.
2. Build an EntireX RPC client using any EntireX wrapper. See *EntireX Wrappers*. For a quick test you can:
   - use the IDL Tester; see *EntireX IDL Tester* in the EntireX Workbench documentation
   - generate an XML mapping file (XMM) and use the XML Tester for verification; see *EntireX XML Tester* in the XML/SOAP Wrapper documentation

See *Client and Server Examples for z/OS CICS* for PL/I RPC Server examples.

Scenario IV: Writing a New PL/I Server

To write a new PL/I server

1. Use the *PL/I Wrapper* to generate a PL/I server skeleton. Write your PL/I server and proceed as described under *Using the PL/I Wrapper for the Server Side*.
2. Build an EntireX RPC client using any EntireX wrapper. See *EntireX Wrappers*. For a quick test you can:
   - use the IDL Tester; see *EntireX IDL Tester* in the EntireX Workbench documentation
   - generate an XML mapping file (XMM) and use the XML Tester for verification; see *EntireX XML Tester* in the XML/SOAP Wrapper documentation

See *Client and Server Examples for z/OS CICS* in the PL/I Wrapper documentation for PL/I RPC Server examples.

Aborting RPC Server Customer Code and Returning Error to RPC Client

This section covers the following topics:

- Using EXEC CICS ABEND ABCODE
- Using EXEC CICS ABEND CANCEL
- Using RETURN-CODE Special Register (COBOL only)

Using EXEC CICS ABEND ABCODE

This approach applies to all CICS scenarios (all programming languages and all interface types); see *Supported Interface Types*.

The CICS feature `EXEC CICS ABEND ABCODE(myabend)` may be used to indicate application error codes. According to IBM CICS standards, `ABEND` codes starting with the letter A are reserved for CICS itself and should not be used in your RPC server.

The CICS RPC Server follows these IBM CICS standards and sends back the RPC protocol message

1. 10010018 `Abnormal termination during program execution`. This is returned when an `ABEND` code starting with the letter "A" is received from CICS, which is a CICS `ABEND`.
2. 10010045 `CICS ABEND myabend was issued`. This is returned when an `ABEND` code starting with a letter other than "A" is received from CICS, which is an application error situation forced by your RPC server.

Using EXEC CICS ABEND CANCEL

This approach applies to all CICS scenarios (all programming languages and all interface types) if impersonation is used (`YES|AUTO`). See *Supported Interface Types* and *Impersonation*. If impersonation is not set, `EXEC CICS ABEND CANCEL` cannot be used.

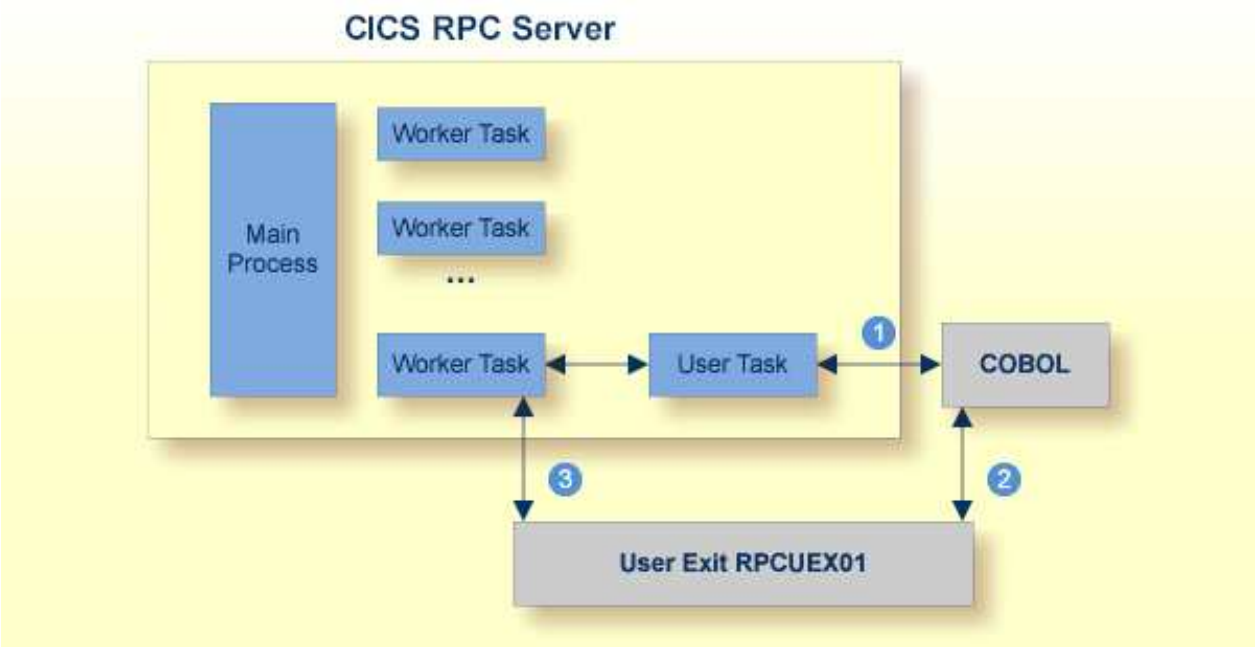
If the customer server code uses the CICS feature `EXEC CICS ABEND CANCEL` to abort for particular error situations, the RPC server cannot trap the abort and is not able to send back an error to the RPC client. The RPC client gets a Broker timeout without any further information about the RPC server abort. In this case, to notify the RPC client you need to call `RPCUEX01` (immediately before `CICS ABEND CANCEL`) in the customer server code to inform the CICS RPC server that your program is about to abort with `CICS ABEND CANCEL`. EntireX does not recommend using `EXEC CICS ABEND CANCEL`. However, if you do need to call an existing COBOL program with `EXEC CICS ABEND CANCEL`, this can be done if the `RPCUEX01` call is added. Whenever possible use `EXEC CICS ABEND ABCODE` instead. See *Using `EXEC CICS ABEND ABCODE`*.

This section covers the following topics:

- Process Flow
- Usage
- Installation

Process Flow

The server invokes the server program using `CICS LINK PROGRAM` and expects that the program returns with `CICS RETURN`. However, if the program uses `CICS ABEND CANCEL` to abort for particular error situations, the RPC server cannot trap the abort. If your server program uses `CICS ABEND CANCEL` you need to call the delivered `RPCUEX01` to inform the server that your program is about to abort with `CICS ABEND CANCEL`.



1. The customer server program is invoked within the user task.
2. The customer server program decides to abort using `CICS ABEND CANCEL`. Immediately before calling `CICS ABEND CANCEL` it calls the `RPCUEX01`. After returning from `RPCUEX01` it performs `CICS ABEND CANCEL` to abort. The `CICS ABEND CANCEL` terminates the user task.
3. `RPCUEX01` posts the worker task and informs it about the abort of its associated user task. The worker task sends back the abort information to the RPC client.

Usage

The server program calls `RPCUEX01` with:

```
EXEC CICS LINK PROGRAM('RPCUEX01')
    COMMAREA(rpcuex01-commarea)
```

After execution, the server program is responsible for aborting the task. If the server program ends without terminating the task, unpredictable results may occur.

Layout of `rpcuex01-commarea`:

- **Return code**
  4-byte integer value. Value of -1 indicates failure.

- **Error text**
  128-byte text field containing the error description.

If the call of `RPCUEX01` fails, the user program must not abort the task.

COBOL example for calling `RPCUEX01`:

```
  01 UEX01-AREA.
    05 RETCODE                          PIC  S9(9) BINARY.
    05 ERRORTEXT                        PIC  X(128).
 ...
    MOVE -1 TO RETCODE
    MOVE 'ERX: No Commarea access' TO ERRORTEXT
    EXEC CICS LINK PROGRAM('RPCUEX01')
              COMMAREA(UEX01-AREA)
              RESP(RESP)
              RESP2(RESP2)
              END-EXEC
    IF RESP NOT = 0
        DISPLAY 'Error invoking RPCUEX01:'
        GO TO MAIN-EXIT
    END-IF
    IF RETCODE IS < 0
        DISPLAY 'Error from RPCUEX01:'
             ' ERRTXT  = ' ERRORTEXT
        GO TO MAIN-EXIT
    END-IF
*   Now cancel the task...
    EXEC CICS ABEND CANCEL END-EXEC
```

Installation

The program `RPCUEX01` must reside in the CICS load library concatenation. The following PPT entry is required:

```
DEFINE PROGRAM(RPCUEX01) GROUP(EXX)
  DESCRIPTION(RPC user exit to abort RPC programs)
  LANGUAGE(C)
```

Using RETURN-CODE Special Register (COBOL only)

This approach applies to the following CICS scenarios:

- *CICS with `DFHCOMMAREA` Calling Convention* (COBOL Wrapper | Extractor)
- *CICS with `DFHCOMMAREA` Large Buffer Interface* (COBOL Wrapper | Extractor)

CICS applications that use the DFHCOMMAREA as communication area (`EXEC CICS LINK` applications) may return error codes if the LINKed application has a C main entry and if this application is running in the same CICS (non-DPL program) as the CICS RPC Server. Under these circumstances, IBM's Language Environment for C provides the application return code to `EIBRESP2`, where it can be detected by the CICS RPC Server.

The following provided modules need to be linked to your application.

- ERXRCSRV, a C main module that calls the intermediate COBOL subroutine RCCALL and catches the error from your RPC server and provides it to the CICS RPC Server. This module is available as source in the source data set EXP970.SRCE as well as precompiled in the load data set EXP970.LD00, so a C compiler is not needed.
- RCCALL, a COBOL subroutine calling your RPC server. This module is available as source in the CICS example server data set EXP970.DVCO.

A step-by-step description is given below, but for ease of use we recommend using the job RCIGY. See below.

To set up *your server* to be able to return application errors manually

1. Change the CALL statement of the RCCALL program below which your RPC server is called instead of "MyCobol" below

```
 IDENTIFICATION DIVISION.
   PROGRAM-ID.    RCCALL.

*****************************************************************
*
*  CICS RPC Server
*
*  Returning Application Errors from RPC Server to RPC Client
*
*  This program calls your target COBOL Server.
*
*  For further information and explanation refer to
*  - "Writing Applications with the COBOL Wrapper"
*  in the delivered documentation.
*
*  $Revision: n.n $
*
*
*  Copyright (C) 1997 - 20nn Software AG, Darmstadt, Germany
*  and/or Software AG USA, Inc., Reston, VA, United States of
*  America, and/or their licensors.
*
*****************************************************************

 ENVIRONMENT DIVISION.

 DATA DIVISION.
   WORKING-STORAGE SECTION.

   LINKAGE SECTION.

   01 DFHCOMMAREA.
      10 DFHCOMM-DUMMY                  PIC X.

 PROCEDURE DIVISION USING DFHCOMMAREA.

 MAIN SECTION.
      CALL "my-cobol" USING DFHEIBLK DFHCOMMAREA.

 MAIN-EXIT.
      EXIT PROGRAM.

 END PROGRAM RCCALL.
```

2. In your RPC server, do not use `EXEC CICS RETURN`, because this prevents the return of the application error code to the CICS RPC server. If you are using a COBOL RPC server generated with the COBOL Wrapper, comment out or remove this line.

3. Compile the `RCCALL` program with a COBOL compiler supported by the COBOL Wrapper.

4. Link the compiled `RCCALL` program, the delivered `ERXRCSRV` module and your RPC server together to a CICS program to be called by the CICS RPC Server. See also *Using the COBOL Wrapper for the Server Side* for supported CICS scenarios.

To set up your server to be able to return application errors using job `RCIGY`

- Execute `RCIGY` as provided in the CICS example source data set EXP970.DVCO.

  This enhanced job will

  1. modify `RCCALL` as needed (step 1 from the manual approach, see above),

  2. add the modified `RCCALL` code to your COBOL input source (step 2 from the manual approach, see above),

  3. link edit with `ERXRCSRV` (step 3 from the manual approach, see above).

Automatic Syncpoint Handling

The CICS RPC Server issues a `SYNCPOINT` command under the following circumstances:

- After a successful non-conversational request or an end-of-conversation, the server issues a `SYNCPOINT COMMIT` command. If you are running under CICS with impersonation, this `SYNCPOINT` command is not executed by the server, but by CICS when the user task is terminated. See *Impersonation*.

- After abnormal termination of a non-conversational request or a conversation due to an error, the server performs a `SYNCPOINT ROLLBACK` command to back out any pending database modifications.