Administering the CICS RPC Server

The EntireX z/OS CICS® RPC Server allows standard RPC clients to communicate with RPC servers on the operating system z/OS under CICS. It supports the programming languages COBOL and PL/I.

This chapter covers the following topics:

- Customizing the RPC Server
- Configuring the RPC Server
- Locating and Calling the Target Server
- Using SSL or TLS with the RPC Server
- User Exit COBUEX02
- Autostart/Stop during CICS Start/Shutdown
- Multiple RPC Servers in the same CICS

Customizing the RPC Server

The following elements are used for setting up the CICS RPC Server:

- ERXMAIN Control Block
- ERXMAIN Macro
- RPC Online Maintenance Facility
- IBM LE Runtime Options

ERXMAIN Control Block

- defines a setup of the CICS RPC Server that is persistent over CICS restarts
- is defined with parameters of the *`ERXMAIN` Macro*; see column 1 in the table under *Configuring the RPC Server*
- contains the following important settings:
  - connection information such as broker ID, see `BKRN`, server address, see `CLZN`, `SRVN` and `SVCN`
  - location and usage of server-side mapping container; see `SVM` and *Usage of Server Mapping Files*
  - scalability parameters such as `endworker`, `minworker` and `maxworker`, see `ENDW`, `MINW` and `MAXW`

- etc.

ERXMAIN Macro

- creates an *ERXMAIN Control Block*, a persistent setup of the CICS RPC Server
- needs to be assembled to define a setup
- is defined in Assembler program EMAINGEN (in EXP970.SRCE) - use this for assembling; see *Build the ERXMAIN Control Block* under *Installing EntireX RPC Servers under CICS*

RPC Online Maintenance Facility

- provides commands (see column 2 in the table below) to vary most of the permanently defined parameters in the *ERXMAIN Control Block* currently in use. All modifications are lost if CICS is restarted. Use *ERXMAIN Macro* for permanent modifications
- allows you to try out new setups of the CICS RPC Server easily without the need to reassemble the ERXMAIN Control Block.
- supports
  - starting
  - stopping
  - pinging
  - monitoring
  - activating trace

  of the CICS RPC Server. See *RPC Online Maintenance Facility*.

IBM LE Runtime Options

Depending on the feature the CICS RPC Server needs to support (see table below) additional runtime options for IBM’s Language Environment need to be set. For a full description of LE runtime options, see z/OS V1R4.0 Lang Env Prog Guide.

Feature	LE Runtime Options	Description
Trap abends of called RPC server programs	`ABTERMENC(RETCODE)` (1)	Required to also trap the LE abends within a server program.
Level of information if called RPC server program terminates by unhandled condition	`TERMTHDACT(UADUMP)` (1)	Forces a U4039 system dump for abends not trapped by the server.
Force `HANDLE ABEND LABEL` getting control for COBOL runtime error and others	`USRHDLR=(CEEWUCHA)` (1)	The server traps abends using `CICS HANDLE ABEND`. With Enterprise COBOL for z/OS, errors resulting from the COBOL runtime (table overflow, for example) bypass the CICS abend handler. Setting `CEEWUCHA` enables the CICS abend handler to also trap the COBOL runtime errors.
Call RPC server programs with AMODE 24 as well	`ALL31(OFF),STACK(,,BELOW)`	If not specified, AMODE 31 is supported.

Note:
(1) Set internally by the CICS RPC Server using application-specific `CSECT CEEUOPT`. The options can be changed if `CEEUOPT` is replaced on CICS RPC Server load module `RPCSRVC` with IBM Linkage Editor.

There are various ways of specifying LE runtime options, for example installation-specific, region-specific (`CEEROPT` available in the `DFHRPL` concatenation) or application-specific (linked `CSECT CEEUOPT`) etc.

Configuring the RPC Server

The following rules apply for the *`ERXMAIN` Macro* syntax (column 1 in table below):

- keywords are given in uppercase
- there are no abbreviations for keywords

The following rules apply for the RPC Online Maintenance Facility commands (column 2 in table below):

- Underscored letters in a command indicate the minimum number of letters that can be used for abbreviation.

  For example, in `brok`erid=localhost, `brok` is the minimum number of letters that can be used as an abbreviation, i.e. the commands `brokerid=localhost` and `brok=localhost` are equivalents.

ERXMAIN Macro Syntax	RPC Online Maintenance Facility Commands	Default	Values	Req/ Opt
BKRN	brokerid	ETB001	Broker ID used by the server. See *Using the Broker ID in Applications*. Example: BKRN=myhost.com:1971	R
CLZN	class	RPC	Server class part of the server address used by the server. The server address must be defined as a service in the broker attribute file (see *Service-specific Attributes* under *Broker Attributes*). Case-sensitive, up to 32 characters. Corresponds to CLASS attribute of the broker attribute file. Example: CLZN=MyRPC	R
SRVN	servername	SRV1	Server name part of the server address used by the server. The server address must be defined as a service in the broker attribute file. See *Service-specific Attributes* under *Broker Attributes*. Case-sensitive, up to 32 characters. Corresponds to SERVER of the broker attribute file. Example: SRVN=mySrv	R
SVCN	service	CALLNAT	Service part of the server address used by the server. The server address must be defined as a service in the broker attribute file. See *Service-specific Attributes* under *Broker Attributes*. Case-sensitive, up to 32 characters. Corresponds to SERVICE attribute of the broker attribute file. Example: SVCN=MYSERVICE	R
CODE	codepage	no codepage transferred	Depending on the internationalization approach, the codepage (locale string) where incoming data is provided to the COBOL server. Conversely, the COBOL server must provide outgoing data in the given codepage, otherwise unpredictable results occur. See *What is the Best Internationalization Approach to use?* under *Internationalization with EntireX* for information on which internationalization approach requires a codepage (locale string). By default, no codepage is transferred to the broker. For the most popular internationalization approach, *ICU Conversion*, the correct codepage (locale string) must be provided. This means it must: • follow the rules described under *Locale String Mapping* • be a codepage supported by the broker • be the codepage used in your environment for file and terminal IO, otherwise unpredictable results may occur. Example: CODE=ibm-273	O
COMP	compresslevel	N	Enforce compression when data is transferred between broker and server. See *Data Compression in EntireX Broker*. compresslevel= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8| 9 | Y | N **0-9** 0=no compression 9=max. compression **N** No compression. **Y** Compression level 6. Example: COMP=6	O
CYCL	restartcycles	15	Number of restart attempts if the broker is not available. This can be used to keep the CICS RPC Server running while the broker is down for a short time. A restart cycle will be repeated at an interval which is calculated as follows: timeout + ETB_TIMEOUT + 60 seconds where timeout is the RPC server parameter (see this table), and ETB_TIMEOUT is the environment variable (see *Environment Variables in EntireX*) When the number of cycles is reached and a connection to the broker is not possible, the RPC server stops. Example: CYCL=30	O

ERXMAIN Macro Syntax	RPC Online Maintenance Facility Commands	Default	Values	Req/ Opt
DPLY	deployment	NO	Activates the deployment service, see *Deployment Service*. Required to use the Server Mapping Deployment Wizard. See *Server Mapping Deployment Wizard* in the EntireX Workbench documentation. **YES** Activates the deployment service. The RPC server registers the deployment service in the broker. **NO** The deployment service is deactivated. The RPC server does not register the deployment service in the broker. Example: DPLY=YES	O
ENCR	encryptionlevel	0	Enforce encryption when data is transferred between client and server. Requires EntireX Security. See ENCRYPTION-LEVEL under *Broker ACI Fields*. **0** Encryption is enforced. **1** Encryption is enforced between server and broker kernel. **2** Encryption is enforced between server and broker kernel, and also between client and broker. Example: ENCR=2	O
ENDW	endworker	TIMEOUT	**NEVER** Defines worker model FIXED with a fixed number of worker threads. The number of active workers is defined with ERXMAIN macro parameter MINW. **TIMEOUT** Defines slow-shrinking worker model SCALE, where the number of worker threads is adjusted to the current number of client requests. With value TIMEOUT, all worker threads not used are stopped in the time specified by the ERXMAIN macro parameter TOUT, except for the minimum number of active workers specified with ERXMAIN macro parameter MINW. The upper limit of workers parallel active is restricted with ERXMAIN macro parameter MAXW. **IMMEDIATE** Defines fast-shrinking worker model SCALE, where the number of worker threads is adjusted to the current number of client requests. With value IMMEDIATE, worker threads not used are stopped immediately as soon as they have finished their conversation, except for the minimum number of active workers defined with ERXMAIN macro parameter MINW. The upper limit of workers active in parallel is restricted with ERXMAIN macro parameter MAXW. This parameter is forced to value TIMEOUT if impersonation is switched on, see *Impersonation* and ERXMAIN macro parameter IMPS. Example: ENDW=IMMEDIATE,MINW=2,MAXW=6	O
MINW	minworker	1	Minimum limit of tasks active in parallel. • For worker model SCALE: minimum number of workers active in parallel. Do not set a value higher than ERXMAIN macro parameter MAXW. • For worker model FIXED: number of workers active in parallel. Do not set a value higher than 31 without adjusting ERXMAIN macro parameter SIZE. See also ERXMAIN macro parameter ENDW. Example: MINW=2	O
MAXW	maxworker	10	Upper limit of tasks active in parallel. • For worker model SCALE: workers active in parallel. Do not set a value higher than 31 without adjusting ERXMAIN macro parameter SIZE. See also ERXMAIN macro parameter ENDW. • For *Impersonation*: workers and impersonated user tasks active in parallel. Do not set a value higher than 15 without adjusting ERXMAIN macro parameter SIZE. See also ERXMAIN macro parameter IMPS. Example: MAXW=2	O

ERXMAIN Macro Syntax	RPC Online Maintenance Facility Commands	Default	Values	Req/ Opt
ETBL	etblnk	CICSETB	Define the broker stub to be used. See *Administering Broker Stubs* for available stubs. Example: ETBL=CICSETB	O
EXIT	n.a.		At startup, the CICS RPC Server will call the user exit to synchronize its version. If successful, the CICS RPC Server will continue and call the user exit for the implemented events. See *User Exit COBUEX02*.	O

ERXMAIN Macro Syntax	RPC Online Maintenance Facility Commands	Default	Values	Req/ Opt
IMPS	impersonation	NO	Defines if RPC requests are executed under the user ID of the RPC client. Depending on settings, different levels of checks are done prior to RPC server execution. See also *Impersonation*. impersonation= NO | YES | AUTO [, sameuser | , anyuser] **NO** The RPC request is executed anonymously, which means the user ID of the RPC client is not used. RPC requests are executed under the user ID of the RPC server. **YES** The RPC request runs impersonated under the supplied *RPC client user ID*. For execution of the RPC request, the CICS RPC Server starts a separate impersonated user task, that is, the client must be know to CICS and the supplied password is validated against CICS. The worker model SCALE is forced; for details see *Impersonation*. **AUTO** Same as option YES above, except that no password validation is performed, that is, the client is treated as already authenticated. For this setting, make sure the RPC client is correctly authenticated; use either • a secure broker (validation must be against the correct mainframe security repository where the user IDs are defined) and option sameuser or • your own security implementation (option anyuser is supported for compatibility reasons if you need different broker and server user IDs - the customer-written security implementation must validate the RPC client using the *RPC client user ID*) **sameuser** The CICS RPC Server checks whether the *broker client user ID* matches the *RPC client user ID*. This is the default if AUTO is used. **anyuser** The *RPC client user ID* is used for impersonation. The *broker client user ID* is ignored. **Notes:** 1. EntireX supports two user ID/password pairs: a *broker client user ID/password* pair and an (optional) *RPC user ID/password* pair sent from RPC clients to the RPC server. 2. With EntireX Security, the *broker client user ID/password* pair is checked. The *RPC user ID/password* pair is designed to be checked by the target RPC server. Thus it is possible to use different user IDs in the broker and target RPC server. 3. RPC clients send the (optional) *RPC user ID/password* pair in the same way as specifying the Natural user ID/password pair for a Natural RPC Server. See for example *Using Natural Security* for applications under C | COBOL | PL/I | Web Services | SOAP/XML | Java. 4. If the RPC client does not specify the optional *RPC user ID/password pair*, the *broker client user ID* is inherited to the *RPC user ID* and thus used for impersonation by the CICS RPC Server. Example: IMPS=auto	O

ERXMAIN Macro Syntax	RPC Online Maintenance Facility Commands	Default	Values	Req/ Opt
LOGN	logon	YES	Execute broker functions LOGON/LOGOFF in worker threads. Must match the setting of the broker attribute AUTOLOGON. Reliable RPC requires logon set to YES. See *Reliable RPC*. **NO** No logon/logoff functions are executed. **YES** Logon/logoff functions are executed. Example: LOGN=no	O
n.a.	mapname		Alias for command *memory*.	O
n.a.	memory		Command to load an *ERXMAIN Control Block*. See *Modifying Parameters of the RPC Server*.	O
OPTS	runoption	0	This parameter is for special purposes. It provides the CICS RPC Server with additional information. The runoptions are normally set to meet the platform's requirements. Set this parameter only if a support representive provides you with an option and asks you to do so. Syntax: OPTS=(<option-list>) <option-list> = [<option-list>,] <option> Example: OPTS=(RUNOPT1,RUNOPT2)	O
PSWD	password		Password for broker logon. Case-sensitive, up to 32 characters. For more information see broker ACI control block field PASSWORD. Example: PSWD=MyPwd	O
PRELOAD	preload	YES	Enable to call CICS RPC Server with AMODE=24 **YES** Enable to call RPC server with AMODE 24 or 31. Internally the CICS RPC Server preloads the called RPC server before execution to check the AMODE and releases the RPC server after this. The disadvantage of this approach is the CICS USECOUNT of the called RPC server program is increased by 2 for every executed RPC call. **NO** The CICS RPC Server does not preload the called RPC server to check its AMODE. All RPC servers are called as running in AMODE 31. This option is useful for customers who require the CICS USECOUNT in their accounting (increased by 1 for every executed RPC call) but prevents usage of calling RPC Server with AMODE 24.	O
REPL	replicatename	ESRV	CICS transaction ID (uppercase, up to 4 characters) assigned to worker tasks and as default for user tasks if *Impersonation* is set. In the START-USER event of the user exit (see *User Exit COBUEX02*) the CICS transaction ID for user tasks can be overridden. See also *Inside the RPC Server*.	O
SIZE	n/a	32768	Size in bytes to hold work memory for worker tasks and impersonated user tasks if impersonation is used. Each task (worker and user) requires the same amount of memory. The following rules apply when calculating the ERXMAIN macro parameter MAXW: 1. The theoretical maximum number of tasks can be calculated using the formula: maximum = integer part of ((SIZE-2036)/864-1). 2. For tasks in intermediate states (starting or ending), the theoretical maximum number must be reduced. We recommend reserving at least 10% for this purpose. 3. If impersonation is used, the theoretical maximum number must be halved. This means: • For the default SIZE value of 32768, the theoretical maximum number of tasks (see rule 1 above) is 34 ((32768-2036)/864-1). • Reducing this value by at least 10% (see rule 2 above) gives 31 for *MAXW* if no impersonation is used. • If impersonation is used, MAXW should be no more than 15 (see rule 3 above).	O
SMH	smhport	0	The port where the server listens for commands from the System Management Hub (SMH). If this port is 0 (default), no port is used and management by the SMH is disabled. See *SMH Listener Service* for more information. Example: SMH=3001	O

ERXMAIN Macro Syntax	RPC Online Maintenance Facility Commands	Default	Values	Req/ Opt
SVM	svmfile		Usage and location of server-side mapping files. See *Server-side Mapping Files in the RPC Server*. If no SVM parameter is given, the RPC server tries to open the server-side mapping container, using CICS file with name ERXSVM. If this CICS file is not available, no server-side mapping files are used. If you use server-side mapping files, the server-side mapping container must be installed and configured; see *Install the Server-side Mapping Container for a CICS RPC Server (Optional)* under *Installing the EntireX RPC Servers under z/OS*. There are also client-side mapping files that do not require configuration here; see *Server Mapping Files for COBOL*. Syntax: SVM=NO | *cicsname* ***cicsname*** The RPC server tries to open the server-side mapping container using the CICS file with name *cicsname*. **no** No server-side mapping files are used. Example: SVM=MYSVM See also *Usage of Server Mapping Files*.	O
TOUT	timeout	600	Timeout in seconds, used by the server to wait for broker requests. See broker ACI control block field WAIT for more information. Also influences restartcycles. See worker model SCALE to define the lifetime of worker threads in slow-shrinking worker model SCALE. Example: TOUT=300	O
TRC1	tracedestination	CSSL	Name of the destination for trace output. A valid CICS transient data queue.	O
TRLV	tracelevel	0	Trace level for the server. See also *Activating Tracing for the RPC Server*. Syntax: TRLV= None | Standard | Advanced | Support **None** No trace output. **Standard** For minimal trace output. **Advanced** For detailed trace output. **Support** This trace level is for support diagnostics and should only be switched on when requested by Software AG support. Example: TRLV=standard	O
TROP	traceoption	none	Additional trace option if trace is active. **None** No additional trace options. **STUBLOG** If tracelevel is Advanced or Support, the trace additionally activates the broker stub log. **NOTRUNC** Normally if a data buffer larger than 8 KB is traced, the buffer trace is truncated. Set this option to write the full amount of data without truncation. **Note:** This can increase the amount of trace output data dramatically if you transfer large data buffers. Example: TROP=(STUBLOG,NOTRUNC)	O
USER	userid	ERXSRV1	Used to identify the server to the broker. See broker ACI control block field USER-ID. Case-sensitive, up to 32 characters. Example: USER=MyUid	R

Locating and Calling the Target Server

The IDL library and IDL program names that come from RPC client are used to locate the RPC server. See `library-definition` and `program-definition`. This two-level concept (library and program) has to be mapped to the CICS RPC Server environment. Different mechanisms are used depending on the language:

- COBOL
- PL/I

COBOL

The approach used to derive the CICS program name for the RPC server depends on whether server mapping is used or not. See *Usage of Server Mapping Files* for an introduction.

1. If the RPC client sends a client-side type of server mapping with the RPC request, this server mapping is used first.
2. If no server mapping is available from step 1 above, and if server-side type of server mapping is used, the IDL library and IDL program names are used to form a key to locate the server mapping in the server-side mapping container. If a server mapping is found, this is then used.
3. If a server mapping is available from step 1 or 2 above, the CICS program name of the RPC server is derived from this mapping. In this case the IDL program name can be different to the CICS program name if it is renamed during wrapping process (see *Customize Automatically Generated Server Names*) or during the extraction process in the *COBOL Mapping Editor*.
4. If no server mapping is used at all, the IDL program name is used as the CICS program name of the RPC server (the IDL library name is ignored).

To use the CICS RPC Server with COBOL

1. Make sure that all CICS programs called as RPC servers
   - use an interface type supported by the CICS RPC Server for target language COBOL; see *Supported Interface Types*.
   - can be called with an `EXEC CICS LINK PROGRAM`
   - are accessible through the CICS RPL chain or accessible remotely using CICS DPL
2. Configure the `ERXMAIN` macro parameter `SVM` depending on whether server-side mapping files are used or not. See also *Usage of Server Mapping Files*.

See also *Scenario I: Calling an Existing COBOL Server* or *Scenario II: Writing a New COBOL Server*.

PL/I

There is a simple mechanism to derive the RPC server CICS program name:

- The IDL program name is used as the CICS program name.
- The IDL library name is not used.

To use the CICS RPC Server with PL/I

1. Make sure that all CICS programs called as RPC servers
   - use an interface type supported by the CICS RPC Server for target language PL/I; see *Supported Interface Types*.
   - can be called with an `EXEC CICS LINK PROGRAM`
   - are accessible through the CICS RPL chain or accessible remotely using CICS DPL

See also *Scenario III: Calling an Existing PL/I Server* or *Scenario IV: Writing a New PL/I Server*.

Using SSL or TLS with the RPC Server

The CICS RPC Server does not have direct SSL or TLS support inside. For this purpose, use instead IBM's Application Transparent Transport Layer Security (AT-TLS), where the establishment of the SSL or TLS connection is pushed down the stack into the TCP layer.

See *SSL or TLS and Certificates with EntireX* for more information.

To set up SSL or TLS with AT-TLS

1. Set up the CICS RPC Server for a TCP/IP connection.
2. Configure the rules for the AT-TLS policy agent the CICS RPC Server matches, for example by using the CICS job name and remote port number the CICS RPC Server connects to. Used certificates are also defined with those rules. Refer to your IBM documentation for further information.
3. Make sure the target the CICS RPC Server connects to is prepared for SSL/TLS connections as well. See the following sections:
   - *Running Broker with SSL or TLS Transport* under z/OS | UNIX | Windows
   - *Settting up and Administering the Broker SSL Agent* under UNIX | Windows
   - Direct RPC in the EntireX Adapter documentation under http://documentation.softwareag.com > *webMethods Product Line*

User Exit COBUEX02

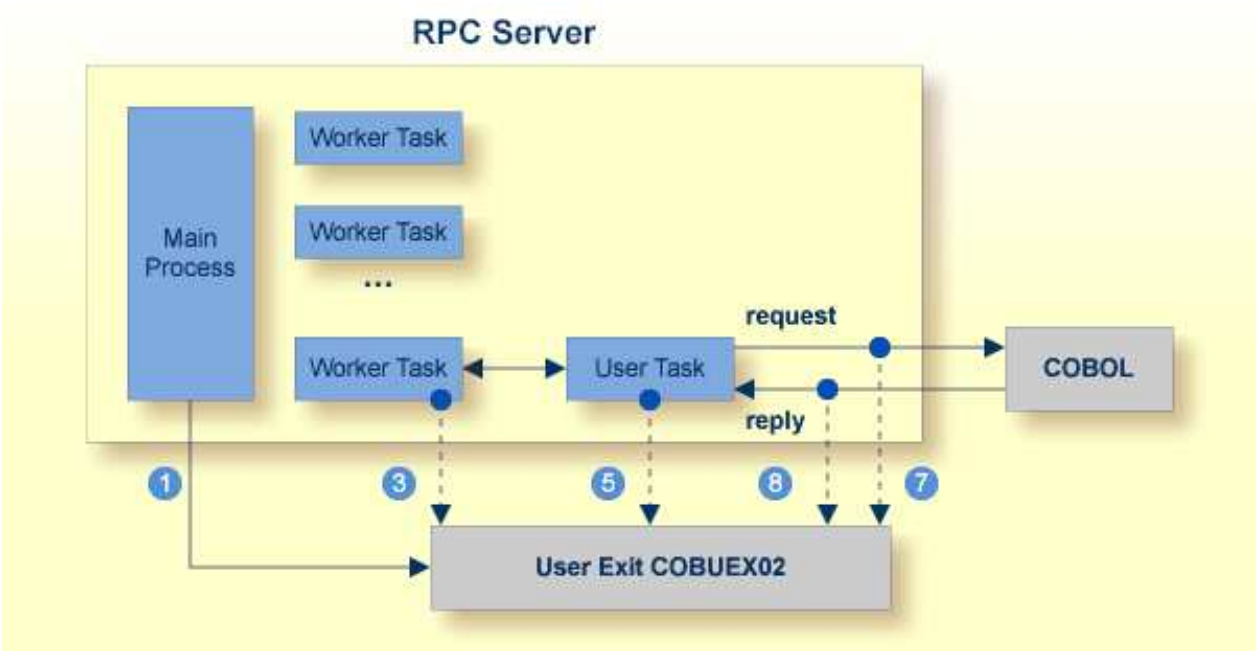
The CICS RPC Server provides a user exit `COBUEX02` to influence/control the RPC logic. This section covers the following topics:

- User Exit Events

- Writing the User Exit
- Configuring the User Exit

User Exit Events

The user exit is called on the following events:



The numbers in the graphic correspond to the event numbers in the user exit.

Step	Event	Called	Description	Note
1	`WHICH-VERSION`	During startup of RPC server.	The CICS RPC Server and the exit decide on the version to use. See field `VERSION`.	
3	`START-WORKER`	Before starting a worker task.	This allows you to set the CICS transaction ID of the worker task. See field `CICS-TRANSID`.	1
5	`START-USER`	Before starting a user task.	Requires *Impersonation*. Before an impersonated CICS transaction (user task) is started, the user exit may change the user ID, CICS transaction ID and CICS terminal ID of the new impersonated user task. See fields `USERID`, `CICS-TRANSID` and `CICS-TERMID`.	1
7	`CALL-START`	Before calling the target CICS program.	See field `RPC-SERVER`. You can inspect and modify the RPC request (payload data from the RPC client to the RPC server).	1
8	`CALL-END`	After calling the target CICS program.	See field `RPC-SERVER`. You can inspect and modify the RPC reply (payload data from the RPC server to the RPC client).	2

Notes:

1. An RPC request can be terminated if an error is given in the fields `ERROR-CODE` and `ERROR-TEXT`. The RPC request is already executed.
2. If an error is given in the fields `ERROR-CODE` and `ERROR-TEXT`, this error is returned to the RPC client. The RPC request is already executed.

Writing the User Exit

The Developer's Kit provides the following resources for COBOL:

- User exit skeleton `COBUEX02` in data set EXP970.SRCE. Copy this skeleton so you have your own user exit source for modifications. The user exit program must comply with the `EXEC CICS LINK PROGRAM COMMAREA` conventions.

- Copybook `COBUEX02` in data set EXP970.INCL. Please add EXP970.INCL to your COBOL compiler `SYSLIB DD` chain. The copybook also contains further description and usage comments.

The parameters of `COBUEX02` are described below.

Parameter	Format	I/O	Description
`EXIT-FUNCTION`	`PIC 9(4) BINARY`	I	Signals the event. See *User Exit Events*.
`VERSION`	`PIC 9(4) BINARY`	I/O	For event `WHICH-VERSION`, see *User Exit Events*. On input, the CICS RPC Server provides the maximum supported version; on output, the exit returns the version to be used. Values 1-2 may be supplied. Some of the fields require a minimum version.
`TRACE-LEVEL`	`PIC 9(4) BINARY`	I	Informational. Value is `0-`*n*. Corresponds to parameter `tracelevel`.

Parameter	Format	I/O	Description
ERROR-CODE	PIC 9(4) BINARY	O	Must be set on output. Possible values: **0** Success. Continue this RPC request. **1..9999** Stop the RPC request. Reply with error class 1022 together with field ERROR-TEXT to the calling RPC client. Example: Error code value *nnnn* results in the following error message: 1022*nnnn error-text*. See *Message Class 1022 - CICS RPC Server User Exit Messages*.
ERROR-TEXT	PIC X(256)	O	Error text returned to RPC client if an error is supplied in field ERROR-CODE. Up to 256 characters.
CICS-TRANSID	PIC X(4)	I/O	Input defined by macro EMAINGEN. Default is ESRV. Can be modified on output. Available for the following events: **START-WORKER** Transaction ID to start the WORKER-TASK. **START-USER** Transaction ID to start the USER-TASK. Requires *Impersonation*.
CICS-TERMID	PIC X(4)	O	Available for the following event: **START-USER** If applied on output, starts the USER-TASK with the supplied terminal ID. Requires *Impersonation*.
USERID	PIC X(32)	I/O	Input supplied by RPC client. Available for the following event: **START-USER** If applied on output, starts the USER-TASK with the supplied user ID. Requires *Impersonation*.
RPC-LIBRARY	PIC X(128)	I	IDL library name. Informational. Availability depends on event and VERSION: **START-USER** If field VERSION >=2 has been set on WHICH-VERSION. **CALL-START** All supported field VERSIONs. **CALL-END** All supported field VERSIONs.
RPC-PROGRAM	PIC X(128)	I	IDL program name. Informational. Availability depends on event and VERSION: **START-USER** If field VERSION >=2 has been set on WHICH-VERSION. **CALL-START** All supported field VERSIONs. **CALL-END** All supported field VERSIONs.
INTERFACE-TYPE	PIC X	I	Type of interface. Informational. Available for events CALL-START and CALL-END. Possible values: **D** DFHCOMMAREA **C** Channel Container **W** Large Buffer
RPC-SERVER	PIC X(8)	I	Target CICS program to call. Informational. Available for the following events: **CALL-START** All supported field VERSIONs. **CALL-END** All supported field VERSIONs.
CHANNEL-NAME	PIC X(16)	I	Name of CICS Channel. Informational. Applicable if field INTERFACE-TYPE is 'C' (Channel Container). Available for the following events: **CALL-START** All supported field VERSIONs. **CALL-END** All supported field VERSIONs.

Parameter	Format	I/O	Description
CHAIN-COUNTER	PIC S9(9) BINARY	I	**Event** / **Interface Type** / **Chain Counter** CALL-START / DFHCOMMAREA / 1 CALL-END / DFHCOMMAREA / 1 CALL-START / Channel Container / *number-input-containers* CALL-END / Channel Container / *number-output-containers* CALL-START / Large Buffer / 1 CALL-END / Large Buffer / 1
CHAIN-POINTER	POINTER	I	Informational. Pointer to first element of a table (or chain of elements) describing payload data. See structure DATA-ENTRY. Available for the following events: **CALL-START** The table or chain of elements (structure DATA-ENTRY) contains descriptions of input payload data. **CALL-END** The table or chain of elements (structure DATA-ENTRY) contains descriptions of output payload data.
CHAIN-COUNTER-OUT	PIC S9(9) BINARY	I	**Event** / **Interface Type** / **Chain Counter** CALL-START / DFHCOMMAREA / 1 CALL-START / Channel Container / *number-output-containers* CALL-START / Large Buffer / 1
CHAIN-POINTER-OUT	POINTER	I	Similar to CHAIN-POINTER. Informational. See structure DATA-ENTRY. Available for the following event: **CALL-START** If field VERSION >=2 has been set on WHICH-VERSION. The table or chain of elements (structure DATA-ENTRY) contains descriptions of expected output payload data with maximum length for variable length data, for example OCCURS DEPENDING ON ...
RPC-RETCODE	PIC 9(9) BINARY	I	RPC error code. Informational. **1001 0045** CICS abend code. **1002** ***nnnn*** User-definable server message. **...** See *Error Messages and Codes*.
CICS-ABCODE	PIC X(4)	I	CICS abend code. Informational.
DATA-ENTRY		I	Structure. Informational. Consists of: DATA-NAME, DATA-LENGTH, DATA-POINTER.

<table>
<tr><th>Parameter</th><th>Format</th><th>I/O</th><th>Description</th></tr>
<tr><td>DATA-NAME</td><td>PIC X(16)</td><td>I</td><td>
<table>
<tr><th>Event</th><th>Interface Type</th><th>Container Name</th></tr>
<tr><td>CALL-START</td><td>DFHCOMMAREA</td><td>'DFHCOMMAREA'</td></tr>
<tr><td>CALL-END</td><td>DFHCOMMAREA</td><td>'DFHCOMMAREA'</td></tr>
<tr><td>CALL-START</td><td>Channel Container</td><td>input-container-name or output-container-name</td></tr>
<tr><td>CALL-END</td><td>Channel Container</td><td>output-container-name</td></tr>
<tr><td>CALL-START</td><td>Large Buffer</td><td>'WM-LCB-INPUT'</td></tr>
<tr><td>CALL-END</td><td>Large Buffer</td><td>'WM-LCB-OUTPUT'</td></tr>
</table>
</td></tr>
<tr><td>DATA-LENGTH</td><td>PIC 9(9) BINARY</td><td>I</td><td>Length or maximum expected length:
With event CALL-START and CHAIN-POINTER-OUT: the maximum length of payload data.
For all other cases, the actual length of the payload data.</td></tr>
<tr><td>DATA-POINTER</td><td>POINTER</td><td>I</td><td>Pointer to payload for this element.</td></tr>
</table>

Configuring the User Exit

Apply the name of your exit routine to the EntireX RPC server ERXMAIN macro parameter EXIT. See *Configuring the RPC Server*.

At startup, the CICS RPC Server will call the named user exit to synchronize its version. If successful, the *RPC Online Maintenance Facility* will display the user exit as map field "parameter opts". See *To display the Server parameters* (PF06) under *RPC Online Maintenance Facility*. The CICS RPC Server will continue and call the user exit for the implemented events.

Autostart/Stop during CICS Start/Shutdown

The CICS RPC Server can be started and stopped automatically during start and stop of the CICS region. For manual start/stop, see *Starting the RPC Server* and *Stopping the RPC Server* under *RPC Online Maintenance Facility*.

To start the CICS RPC Server during the initialization of CICS

1. If the COBOL source ERXSTART of the EntireX installation library EXP970.SRCE has not been defined in the CICS CSD data sets by the installation job $INSTALL, define it.
2. Customize and compile ERXSTART if necessary.
3. Add the following entry to your CICS PLTPI table (second phase PLT program):

   ```
   DFHPLT TYPE=ENTRY,PROGRAM=ERXSTART
   ```

   See also *Starting the EntireX RPC Server Automatically on CICS Startup (Optional)* under *Installing EntireX RPC Servers under CICS* in the z/OS installation documentation.

To stop the CICS RPC Server during the shutdown of CICS

1. If the COBOL source ERXSTOP of the EntireX installation library EXP970.SRCE has not been defined in the CICS CSD data sets by the installation job $INSTALL, define it.
2. Customize and compile ERXSTOP if necessary.
3. Add the following entry to your CICS PLTSD table (first phase PLT program):

   ```
   DFHPLT TYPE=ENTRY,PROGRAM=ERXSTOP
   ```

   See also *Stopping the EntireX RPC Server Automatically on CICS Shutdown (Optional)* under *Installing EntireX RPC Servers under CICS* in the z/OS installation documentation.

Multiple RPC Servers in the same CICS

If you need to install multiple instances in the same CICS region, see *Installing Multiple EntireX RPC Servers in the same CICS (Optional)* under *Installing EntireX RPC Servers under CICS* in the z/OS installation documentation.