webMethods EntireX

EntireX z/OS CICS® RPC Server

Innovation Release

Version 9.9

October 2015

WEBMETHODS

This document applies to webMethods EntireX Version 9.9 and all subsequent releases.

Specifications contained herein are subject to change and these changes will be reported in subsequent release notes or new editions.

Document ID: EXX-CICSRPC-99-20171128

Table of Contents

EntireX z/OS 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.

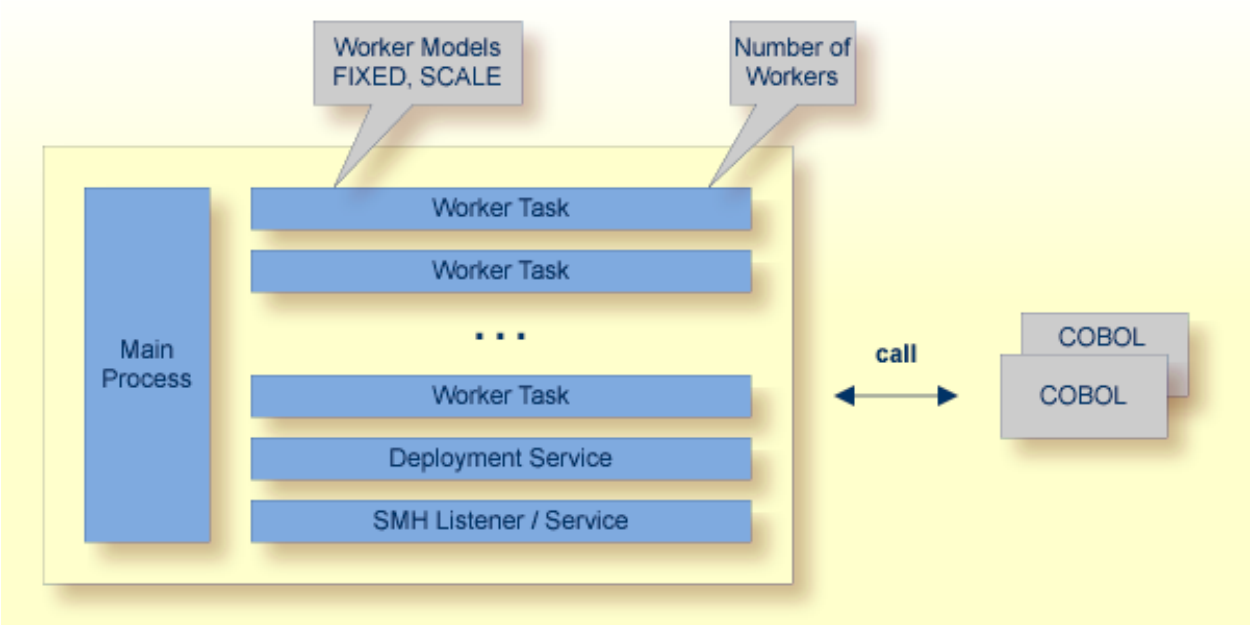
- For COBOL, it works together with the *COBOL Wrapper* and *IDL Extractor for COBOL*.
- For PL/I, it works together with the *PL/I Wrapper* and *IDL Extractor for PL/I*.

Supported compilers are listed under *z/OS Prerequisites* in the EntireX Release Notes in the EntireX Release Notes.

1 Introduction to 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.

Worker Models



RPC requests are worked off inside the RPC server in worker tasks, which are controlled by a main task. Every RPC request occupies during its processing a worker task. If you are using RPC conversations, each RPC conversation requires its own task during the lifetime of the conversation. The CICS RPC Server provides two worker models:

- `FIXED`
  The *fixed* model creates a fixed number of worker tasks. The number of worker tasks (defined with `ERXMAIN` **macro** parameter `MINW`) does not increase or decrease during the lifetime of an RPC server instance. It is configured by setting the `ERXMAIN` **macro** parameter `ENDW` to value `"NEVER"`. Example:

  ```
  ENDW=NEVER, MINW=4
  ```

- `SCALE`
  The *scale* model creates worker tasks depending on the incoming load of RPC requests.

  A maximum number (`ERXMAIN` **macro** parameter `MAXW`) of the worker tasks created can be set to restrict the system load. The minimum number (`ERXMAIN` **macro** parameter `MINW`), allows you to define a certain number of tasks - not used by the currently executing RPC request - to wait

for new RPC client requests to process. In this way the RPC server is ready to handle many RPC client requests arriving at the same time. It is configured by setting the ERXMAIN **macro** parameter ENDW to value "TIMEOUT" or "IMMEDIATE".

- With value IMMEDIATE, worker tasks shrink fast, that is, a worker task not used is stopped immediately as soon as it has finished its conversation, except for the number of workers specified as minimum being active.
- With value TIMEOUT, worker tasks shrink slowly, that is, all worker tasks not used are stopped in the time specified by the ERXMAIN **macro** parameter TOUT, except for the number of workers specified as minimum being active.

Example:

```
ENDW=IMMEDIATE, MINW=2,MAXW=6
```

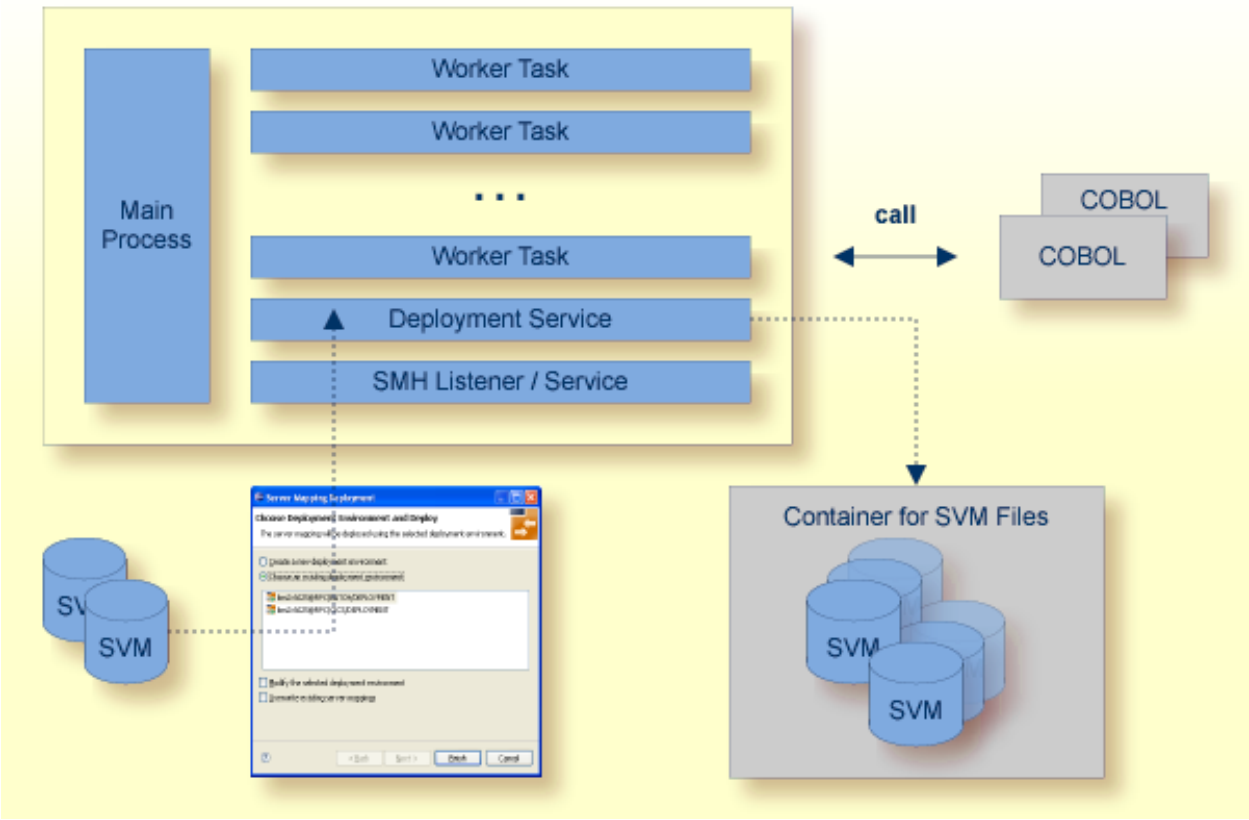
Inbuilt Services

CICS RPC Server provides the following services for ease-of-use:

- Deployment Service
- SMH Listener Service

Deployment Service

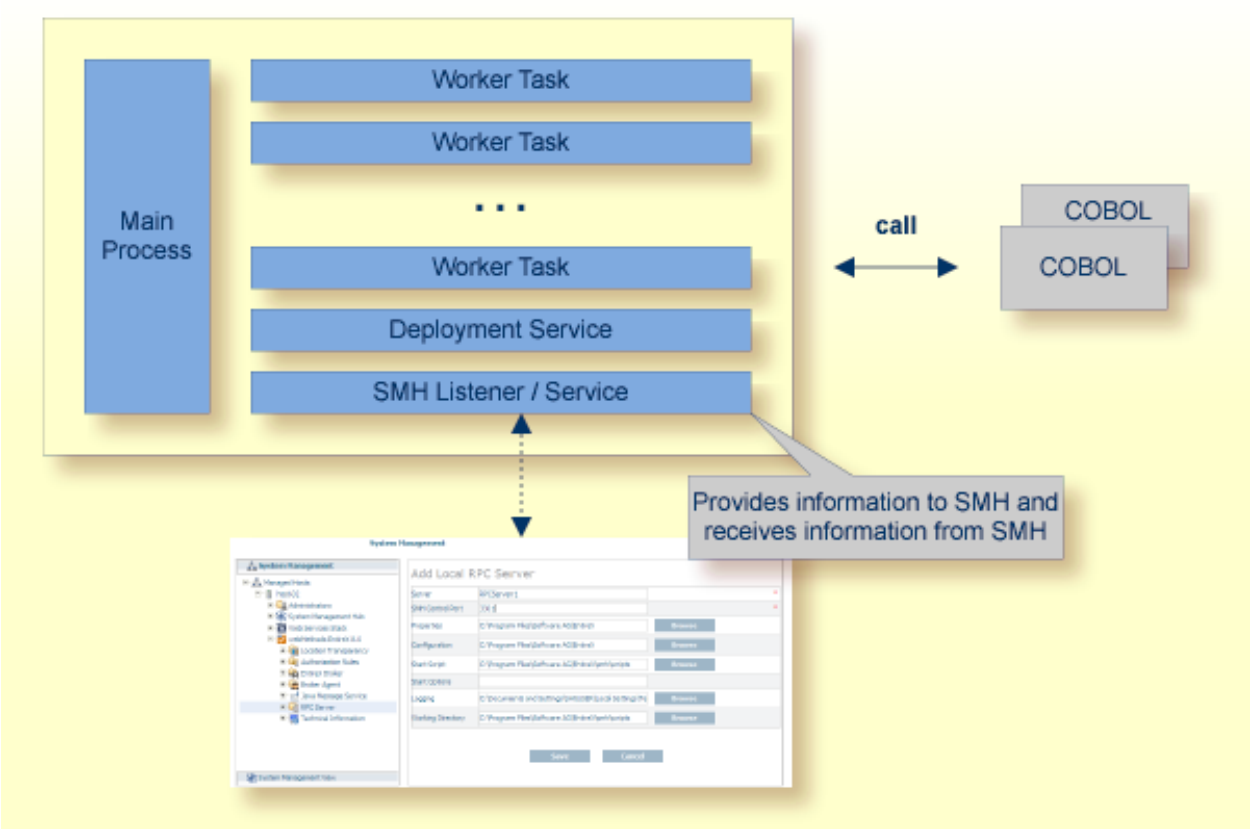
The Deployment Service allows you to deploy server-side mapping files (EntireX Workbench files with extension .svm) interactively using the *Server Mapping Deployment Wizard*. On the RPC server side, the server-side mapping files are stored in a server-side mapping container (VSAM file). See *Server-side Mapping Files in the RPC Server* and *Deployment Service* for configuration information.



SMH Listener Service

With the SMH Listener Service you use the System Management Hub to monitor the RPC server. See *Administering the EntireX RPC Servers using System Management Hub* in the UNIX and Windows administration documentation.

The SMH Listener Service is switched on if the SMH port number is set. See the `ERXMAIN` **macro** parameter `SMH` under *Configuring the RPC Server*.

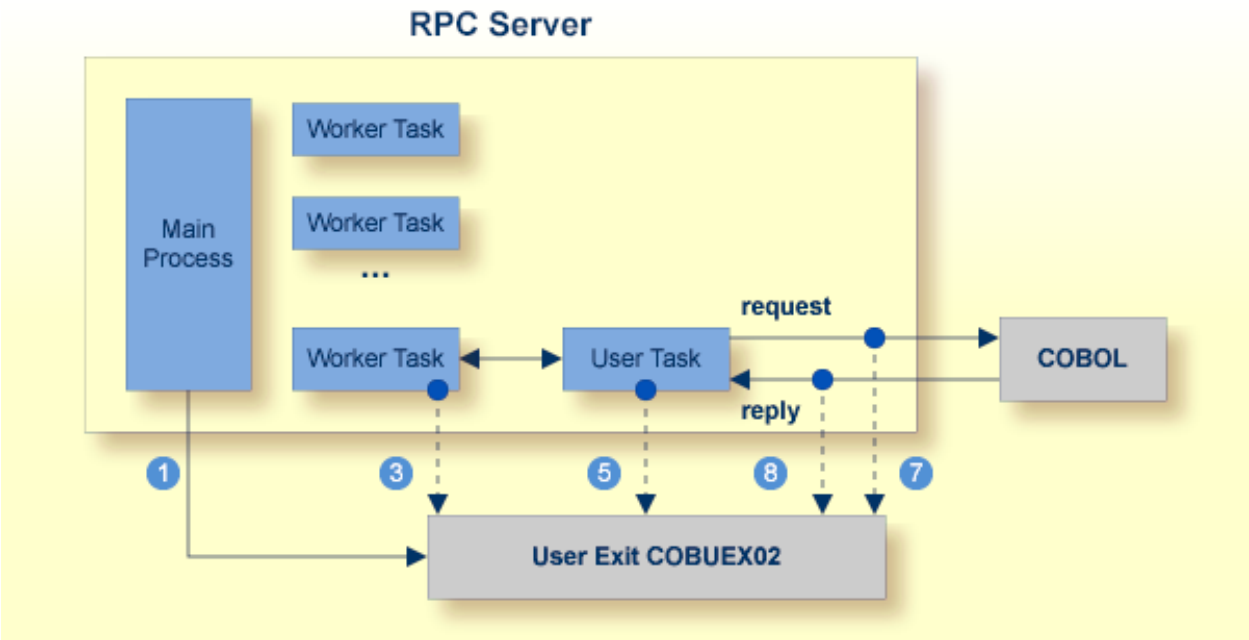


User Exit COBUEX02

The CICS RPC Server provides a user exit `COBUEX02`to influence/control the RPC logic. The exit is called on the events `START-WORKER`, `START-USER`, `CALL-START` and `CALL-END`. The following tasks can be performed:

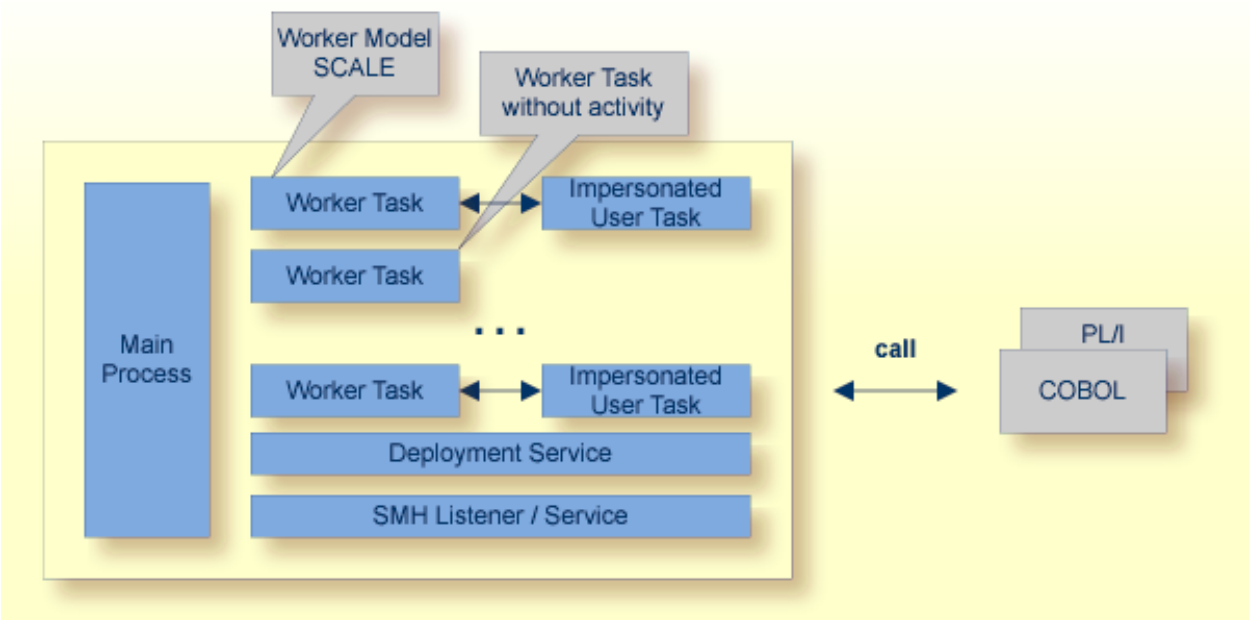
1. `WHICH-VERSION` event. Tells the CICS RPC Server the version to use.
3. `START-WORKER` event. Allows you to set the CICS transaction ID.
5. `START-USER` event. Apply user ID, CICS transaction ID and CICS terminal ID to impersonated user tasks. See ***Impersonation***.
7. `CALL-START` event. Inspect, modify or terminate the RPC request (payload) from the RPC client.
8. `CALL-END` event. Inspect or modify the RPC reply (payload) or give an error to the RPC client.

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



See also ***User Exit*** `COBUEX02` under *Administering the EntireX RPC Server*.

Impersonation



The CICS RPC Server can be configured to execute the RPC request impersonated under the RPC client user ID. For this, worker tasks start additional impersonated user tasks. This can be useful, for example for accounting. Impersonation is controlled by the `ERXMAIN` **macro** parameter `IMPS`.

- For `IMPS` value `AUTO`, the CICS RPC Server does not validate RPC passwords, so you have to take care the RPC client is correctly authenticated, either by using a secure EntireX Broker (validation must be against the correct mainframe security repository where CICS user IDs are defined) or with your own security implementation.
- For `IMPS` value `YES`, the CICS RPC Server uses the RPC user ID and password supplied by the RPC client for authentication and impersonation of the client. This means that the RPC server validates the password.

The picture above shows the configuration `IMPS=YES`.

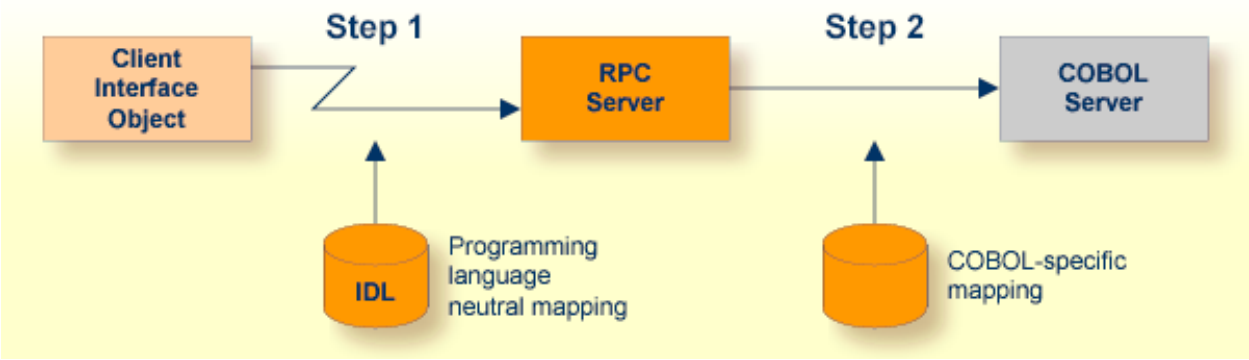
The lifetime of an impersonated user task starts when an open request for an RPC conversation or a non-conversational RPC request is received. It ends when the RPC conversation stops (after a commit operation or timeout) or when the non-conversational RPC request has been performed.

For worker tasks, the slow-shrinking worker model `SCALE` is used - value `TIMEOUT` is forced internally - any value given in the `ERXMAIN` **macro** parameter `ENDW` is ignored. The lifetime of worker tasks can be controlled with `ERXMAIN` **macro** parameter `TOUT` as well as the number of workers with macro parameters `MINW` and `MAXW`.

Usage of Server Mapping Files

There are many situations where the CICS RPC Server requires a server mapping file to correctly support special COBOL syntax such as `REDEFINES`, `SIGN LEADING` and `OCCURS DEPENDING ON` clauses, `LEVEL-88` fields, etc.

Server mapping files contain COBOL-specific mapping information that is not included in the IDL file, but is needed to successfully call the COBOL server program.



The RPC server marshals the data in a two-step process: the RPC request coming from the RPC client (Step 1) is completed with COBOL-specific mapping information taken from the server mapping file (Step 2). In this way the COBOL server can be called as expected.

The server mapping files are retrieved as a result of the *IDL Extractor for COBOL* extraction process and the *COBOL Wrapper* if a COBOL server is generated. See *When is a Server Mapping File Required?*.

There are *server*-side mapping files (*EntireX Workbench* files with extension .svm) and *client*-side mapping files (Workbench files with extension .cvm). See *Server Mapping Files for COBOL* and *How to Set the Type of Server Mapping Files*.

If you are using server-side mapping files, you need to customize the server-side mapping container with `ERXMAIN` **macro** parameter `SVM`. See ***Configuring the RPC Server***.

Note: Server mapping files are used for COBOL only.

Supported Interface Types

The interface types supported by the CICS RPC Server vary depending on the target programming language. See also ***Locating and Calling the Target Server***.

COBOL

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

PL/I

- *CICS with* `DFHCOMMAREA` *Calling Convention* (PL/I Wrapper | Extractor)

2 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.

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
- CICS Settings

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.
- the default name for the control block is ERXMAIN, but any meaningful name can be chosen. Using this name as input parameter *memory* for the *RPC Online Maintenance Facility* means that multiple CICS RPC Servers can be started and monitored in parallel. See *Installing Multiple EntireX RPC Servers in the same CICS (Optional)*.

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 EXP990.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.

CICS Settings

CICS Parameter	Description	Default	How to change?
`TWASIZE`	Transaction Work Area (TWA) size may be used by target RPC programs called by the CICS RPC server. If this is the case, the TWA size set for the CICS RPC server must match the largest TWA size required by all called target RPC programs.	`TWASIZE(28)` This corresponds to 'Adabas Parameter List' if transport method NET is used. See *Installing Adabas with TP Monitors for EntireX* in the z/OS installation documentation.	■ Use resource definition online (RDO), CICS transaction CEDA. ■ See member `DFHERX` in EXP990.SRCE data set. See *Update the CICS Tables* under *Installing the EntireX RPC Servers under z/OS* in the z/OS installation documentation.

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 `brokerid=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* in the RPC Programming documentation. Example: `BKRN=myhost.com:1971`	R

ERXMAIN Macro Syntax	RPC Online Maintenance Facility Commands	Default	Values	Req/ Opt
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* (DEFAULTS=SERVICE) under *Broker Attributes* in the platform-independent administration documentation). 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* (DEFAULTS=SERVICE) under *Broker Attributes* in the platform-independent administration documentation. 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* (DEFAULTS=SERVICE) under *Broker Attributes* in the platform-independent administration documentation. 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* under *Introduction to Internationalization*, the correct codepage (locale string) must be provided. This means it must:	O

ERXMAIN Macro Syntax	RPC Online Maintenance Facility Commands	Default	Values	Req/ Opt
			■ follow the rules described under *Locale String Mapping* in the internationalization documentation ■ 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`	

`COMP`	`compresslevel`	`N`	Enforce compression when data is transferred between broker and server. See *Data Compression in EntireX Broker* in the general administration documentation.
			
			`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 every 60 seconds. When the number of cycles is reached and a connection to the broker is not possible, the RPC server stops. Example: `CYCL=30`	O
`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:	O

ERXMAIN Macro Syntax	RPC Online Maintenance Facility Commands	Default	Values	Req/ Opt
			DPLY=YES	
ENCR	encryptionlevel	0	Deprecated. For encrypted transport we strongly recommend using the Secure Sockets Layer/Transport Layer Security protocol. See *SSL/TLS and Certificates with EntireX*.	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

ERXMAIN Macro Syntax	RPC Online Maintenance Facility Commands	Default	Values	Req/ Opt
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
ETBL	etblnk	CICSETB	Define the broker stub to be used. See *Administering Broker Stubs* in the z/OS administration documentation 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 COBUEXO2***.	O
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]	O

ERXMAIN Macro Syntax	RPC Online Maintenance Facility Commands	Default	Values			Req/ Opt
			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	

ERXMAIN Macro Syntax	RPC Online Maintenance Facility Commands	Default	Values	Req/ Opt
			broker client user ID is ignored. **Note:** 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* in the respective section of the documentation. 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`	
`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	O

ERXMAIN Macro Syntax	RPC Online Maintenance Facility Commands	Default	Values	Req/ Opt
			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)	
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 COBUEXO2***) the CICS transaction ID for user tasks can be overridden. See also ***Introduction to the CICS RPC Server***.	O
SIZE	n/a	32768	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.	O

ERXMAIN Macro Syntax	RPC Online Maintenance Facility Commands	Default	Values	Req/ Opt
			■ 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`	
`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

`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`.	O

ERXMAIN Macro Syntax	RPC Online Maintenance Facility Commands	Default	Values	Req/ Opt
			Example: `TOUT=300`	
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` under *Software AG IDL Grammar* in the *IDL Editor* documentation. 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

- 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/TLS with the RPC Server

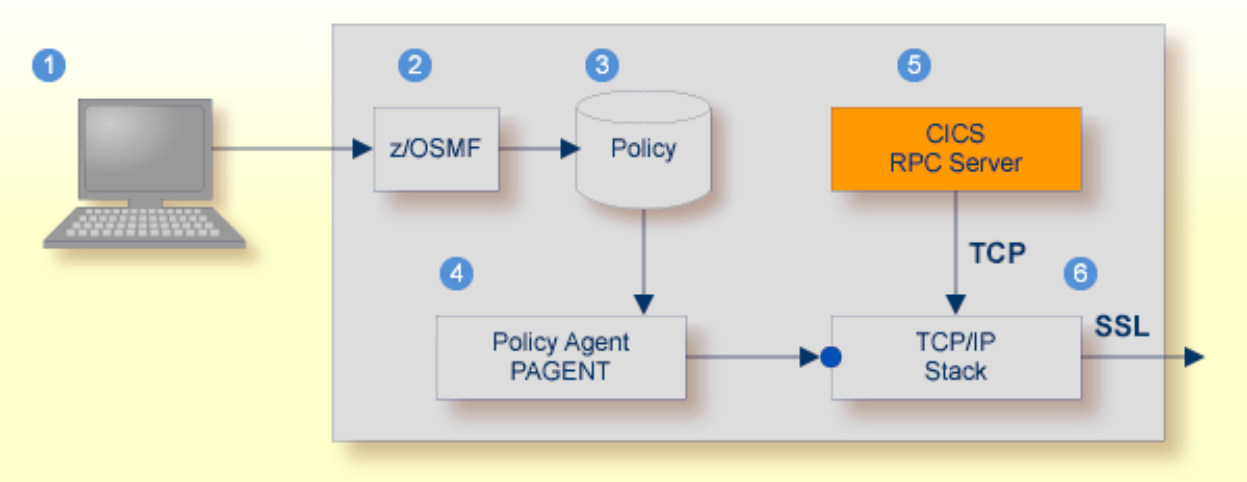
RPC servers can use Secure Sockets Layer/Transport Layer Security (SSL/TLS) as the transport medium. The term “SSL” in this section refers to both SSL and TLS. RPC-based servers are always SSL clients. The SSL server can be either the EntireX Broker, Broker SSL Agent, or Direct RPC in webMethods Integration Server (IS inbound). For an introduction see SSL/TLS and Certificates in the Security documentation.

SSL delivered on a z/OS mainframe will typically use the Resource Access Control Facility (RACF) as the certificate authority (CA). Certificates managed by RACF can only be accessed through the RACF keyring container. A keyring is a collection of certificates that identify a networking trust relationship (also called a trust policy). In an SSL client/server network environment, entities identify themselves using digital certificates called through a keyring. Server applications on z/OS that wish to establish network connections to other entities can use keyrings and their certificate contents to determine the trustworthiness of the client or peer entity. Note that certificates can belong to more than one keyring, and you can assign different users to the same keyring. Because of the way RACF internally references certificates, they must be uniquely identifiable by owner and label, and also unique by serial number plus data set name (DSN).

With the CICS RPC Server you can use IBM's Application Transparent Transport Layer Security (AT-TLS), where the establishment of the SSL connection is pushed down the stack into the TCP layer.

Using IBM's Application Transparent Transport Layer Security (AT-TLS)

Configure the AT-TLS rules for the policy agent (PAGENT) 4 using an appropriate client 1 and the z/OS Management Facility (z/OSMF) 2. Together with SSL parameters (to provide certificates stored in z/OS as RACF keyrings) define AT-TLS rules, for example by using the application 5 job name and remote TCP port number. If the rules match, the TCP connection is turned into an SSL connection 6. Refer to your IBM documentation for more information, for example the IBM Redbook *Communications Server for z/OS VxRy TCP/IP Implementation Volume 4: Security and Policy-Based Networking*.



1 Client to interact with z/OS Management Facility (z/OSMF).

2 AT-TLS rules are defined with z/OSMF policy management.

3 Policy Repository with AT-TLS rules stored as z/OS files.

4 Policy Agent, MVS task PAGENT, provides AT-TLS rules through a policy enforcement point (PEP) to TCP/IP stack.

5 Application using TCP connection.

6 If AT-TLS rules match, the TCP connection is turned into an SSL connection.

Notes:

1. The client 1 may vary per operating system, for example a Web browser for z/OS 2.1.
2. z/OSMF 2 includes other administration and management tasks in addition to policy management.
3. Policy Management 3 includes other rules, such as IP filtering, network address translation etc.

To set up SSL with AT-TLS

1 To operate with SSL, certificates need to be provided and maintained. Depending on the platform, Software AG provides default certificates, but we strongly recommend that you create your own. See *Default Certificates Delivered with EntireX*.

2 Set up the CICS RPC Server for a TCP/IP connection. On mainframe platforms, use *Transport-method-style Broker ID*. Example:

```
ETB024:1699:TCP
```

3 Configure AT-TLS to turn the TCP/IP connection to an SSL connection, see above.

4 Make sure the SSL server to which the CICS RPC Server connects is prepared for SSL connections as well. The SSL server can be EntireX Broker, Broker SSL Agent, or Direct RPC in webMethods Integration Server (IS inbound). See:

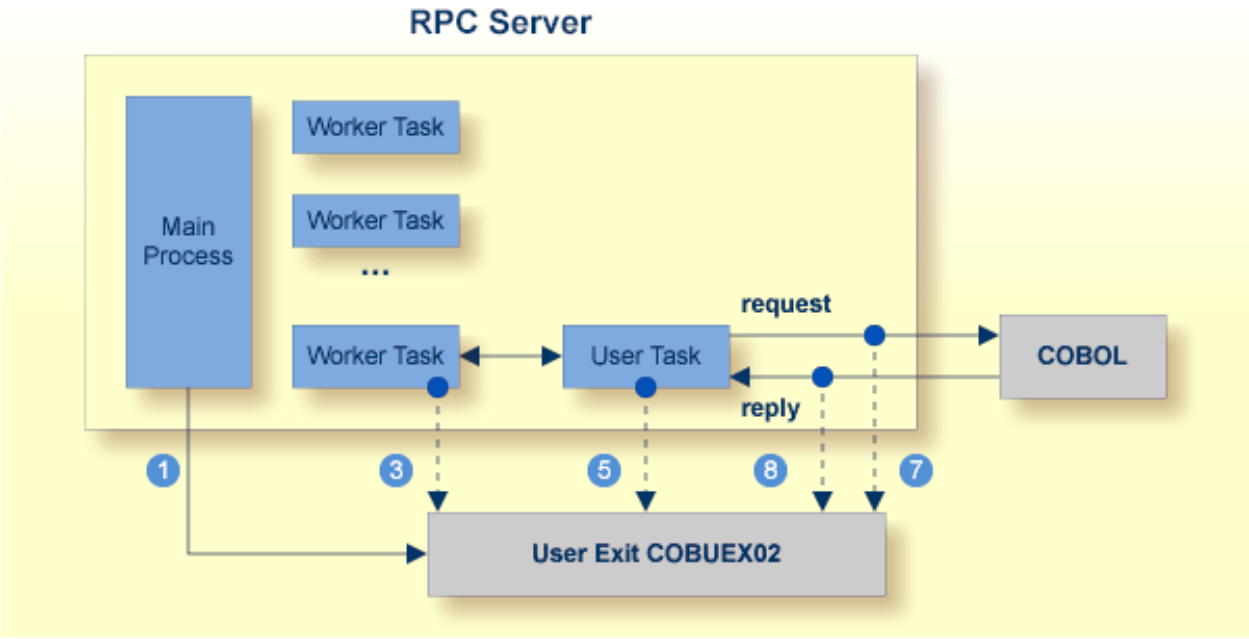
- *Running Broker with SSL/TLS Transport* in the platform-specific administration documentation
- *Settting up and Administering the EntireX Broker SSL Agent* in the UNIX and Windows administration documentation
- *Support for SSL/TLS* in the EntireX Adapter documentation (for Direct RPC)

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

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 EXP990.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 EXP990.INCL. Please add EXP990.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`.
`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* under *Error Messages and Codes*.
`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*.

<table>
<tr><th>Parameter</th><th>Format</th><th>I/O</th><th>Description</th></tr>
<tr><td>USERID</td><td>PIC X(32)</td><td>I/O</td><td>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.</td></tr>
<tr><td>RPC-LIBRARY</td><td>PIC X(128)</td><td>I</td><td>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.</td></tr>
<tr><td>RPC-PROGRAM</td><td>PIC X(128)</td><td>I</td><td>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.</td></tr>
<tr><td>INTERFACE-TYPE</td><td>PIC X</td><td>I</td><td>Type of interface. Informational. Available for events CALL-START and CALL-END. Possible values:

D DFHCOMMAREA
C Channel Container
W Large Buffer</td></tr>
<tr><td>RPC-SERVER</td><td>PIC X(8)</td><td>I</td><td>Target CICS program to call. Informational. Available for the following events:

CALL-START All supported field VERSIONs.
CALL-END All supported field VERSIONs.</td></tr>
<tr><td>CHANNEL-NAME</td><td>PIC X(16)</td><td>I</td><td>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.</td></tr>
<tr><td>CHAIN-COUNTER</td><td>PIC S9(9) BINARY</td><td>I</td><td>
<table>
<tr><th>Event</th><th>Interface Type</th><th>Chain Counter</th></tr>
<tr><td>CALL-START</td><td>DFHCOMMAREA</td><td>1</td></tr>
<tr><td>CALL-END</td><td>DFHCOMMAREA</td><td>1</td></tr>
<tr><td>CALL-START</td><td>Channel Container</td><td>number-input-containers</td></tr>
<tr><td>CALL-END</td><td>Channel Container</td><td>number-output-containers</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Parameter</th><th>Format</th><th>I/O</th><th>Description</th></tr>
<tr><td></td><td></td><td></td><td>
<table>
<tr><th>Event</th><th>Interface Type</th><th>Chain Counter</th></tr>
<tr><td>CALL-START</td><td>Large Buffer</td><td>1</td></tr>
<tr><td>CALL-END</td><td>Large Buffer</td><td>1</td></tr>
</table>
</td></tr>
<tr><td>CHAIN-POINTER</td><td>POINTER</td><td>I</td><td>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.</td></tr>
<tr><td>CHAIN-COUNTER-OUT</td><td>PIC S9(9) BINARY</td><td>I</td><td>
<table>
<tr><th>Event</th><th>Interface Type</th><th>Chain Counter</th></tr>
<tr><td>CALL-START</td><td>DFHCOMMAREA</td><td>1</td></tr>
<tr><td>CALL-START</td><td>Channel Container</td><td>number-output-containers</td></tr>
<tr><td>CALL-START</td><td>Large Buffer</td><td>1</td></tr>
</table>
</td></tr>
<tr><td>CHAIN-POINTER-OUT</td><td>POINTER</td><td>I</td><td>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 ...</td></tr>
<tr><td>RPC-RETCODE</td><td>PIC 9(9) BINARY</td><td>I</td><td>RPC error code. Informational.

1001 0045 CICS abend code.
1002 nnnn User-definable server message.
...

See Error Messages and Codes.</td></tr>
<tr><td>CICS-ABCODE</td><td>PIC X(4)</td><td>I</td><td>CICS abend code. Informational.</td></tr>
<tr><td>DATA-ENTRY</td><td></td><td>I</td><td>Structure. Informational. Consists of: DATA-NAME, DATA-LENGTH, DATA-POINTER.</td></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>
</table>
</td></tr>
</table>

Parameter	Format	I/O	Description
			Event / **Interface Type** / **Container Name** CALL-START / Channel Container / *input-container-name* or *output-container-name* CALL-END / Channel Container / *output-container-name* CALL-START / Large Buffer / 'WM-LCB-INPUT' CALL-END / Large Buffer / 'WM-LCB-OUTPUT'
DATA-LENGTH	PIC 9(9) BINARY	I	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.
DATA-POINTER	POINTER	I	Pointer to payload for this element.

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 EXP990.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* 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 EXP990.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* 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* under *Installing EntireX RPC Servers under CICS* in the z/OS installation documentation.

3 RPC Online Maintenance Facility

Monitoring the RPC Server

The parameters in the following screens are described under *Configuring the RPC Server*.

To call the RPC Online Maintenance Facility and display the RPC Broker Parameters

- Start the CICS transaction

```
ERXM [MEM=erxmain-control-block]
```

where *erxmain-control-block* is the name of the ERXMAIN control block. See **ERXMAIN Control Block** under *Customizing the RPC Server*.

The RPC Broker Parameter map is displayed:

```
11:56:56          --- ERX CICS Online utility  V990.0 ---          06/02/2014
                           RPC Broker Parameter


  Broker parameter
   Broker name    = ETB001
   Class name     = RPC
   Server name    = SRV1
   Service name   = CALLNAT
   User ID        = ERXSRV1
   Code page      =

   Logon          = No                    Server timeout =   600
                                          Encryptionlevel=   0
                                          Compression lvl=   N
   ETBLNK         = CICSETB




  COMMAND ===>
================================================================================
PF01=Help   03=Exit   04=Control       05=Broker parms      06=Server parms
                      08=Start server  09=Ping server       10=Stop server
```

Press **PF05** from any map to return to the RPC Broker Parameter map.

To display the RPC Server Parameters

- Press **PF06** from any map and the RPC Server Parameters will be displayed:

```
12:03:05          --- ERX CICS Online utility  V990.0 ---          06/02/2014
                           RPC Server Parameter

   Server parameter
    # Min. Workers =     2             Trace Level    = 0
    # Max. Workers =     2             Trace Dest.(TD)= CSSL
    Ending Workers = Never
    Impersonation  = No
    Deployment     = Yes
    Restart Cycles =    3
    SMH Port       =

    Server options =  SVM      AutoSYNC
    Marshal options=

   CICS parameter                      Mapping file   = ERXSVM   (Prefered)
    Memory name    = ERXMAIN  (V900)    Dsn(ENTIREX.SVMDEV.KSDS)
    Transaction ID = ESRV               Opn Add Rea Upd Del


  COMMAND ===>
================================================================================
PF01=Help   03=Exit   04=Control       05=Broker parms      06=Server parms
                      08=Start server  09=Ping server       10=Stop server
```

To display the RPC Server Control map

- Press **PF04**.

```
 12:07:18          --- ERX CICS Online utility  V990.0 ---          06/02/2014
                              RPC Server Control

   MAIN Task
    Status          Running

   WORKER Tasks
    Registered             2
    Busy                   0
    Maximum busy           2

   USER Tasks
    Active                 0
    Max. active            0

   BrokerId in use:     ETB001
   Class in use:        RPC
   Server Name in use:  SRV1
   Service in use:      CALLNAT

   COMMAND ===>
```

```
================================================================================
PF01=Help   03=Exit   04=Control       05=Broker parms      06=Server parms
                      08=Start server  09=Ping server       10=Stop server
```

To display help for the RPC Online Maintenance Facility

- Enter `Help` or press **PF01**.

To stop the RPC Online Maintenance Facility

- Enter `Exit` or press **PF03**.

Starting the RPC Server

To start the CICS RPC Server using the RPC Online Maintenance Facility

1. Start the CICS transaction `ERXM` to call the RPC Online Maintenance Facility. See also *Monitoring the RPC Server*.
2. Start the server with the **PF08** key or with the command `start`.
   The status of the `MAIN` task (see RPC server control panel) changes to "is running". The defined number (see `ERXMAIN` **macro** parameter `MINW`) of worker tasks that are registered is displayed.

If an error occurred and the CICS RPC Server is not correctly registered in the broker, but the number of currently active worker tasks is not zero:

- Check with CICS command `CEMT INQUIRE TASK` whether server instances are already running. If yes, stop them using native CICS commands.
- Verify the server parameters matching your system requirements. See column 2 of table under *Configuring the RPC Server*.
- Then issue command `start` or use **PF08**.

Pinging the RPC Server

To ping the CICS RPC Server using the RPC Online Maintenance Facility

1. Start the CICS transaction `ERXM` to call the EntireX RPC Online Maintenance Facility. See *Monitoring the RPC Server*.
2. Issue the command `ping` or use **PF09**.

Alternative Method

- Use the `ping` command from the console. See *Console Commands for the RPC Server*.

Stopping the RPC Server

To stop the CICS RPC Server using the RPC Online Maintenance Facility

1. Start the CICS transaction `ERXM` to call the RPC Online Maintenance Facility. See *Monitoring the RPC Server*.
2. Issue the `stop` command or use **PF10**. This ensures correct deregistration from broker and all worker tasks are shut down.

Modifying Parameters of the RPC Server

With RPC Online Maintenance Facility commands, CICS RPC Server parameters can be temporarily modified. Modifications are lost if CICS is restarted. The purpose of the commands is to try out easily new configurations. For persistent modifications (setup) of the CICS RPC Server, reassemble the `ERXMAIN` **Control Block** using the `ERXMAIN` **Macro**.

To modify the CICS RPC Server parameters using the RPC Online Maintenance Facility

1. Start the CICS transaction `ERXM` to call the RPC Online Maintenance Facility. See *Monitoring the RPC Server*.
2. Use the appropriate RPC Online Maintenance Facility command to modify the parameters. See the column 2 of table under *Configuring the RPC Server*.

Activating Tracing for the RPC Server

To switch on tracing for the CICS RPC Server using the RPC Online Maintenance Facility

A prerequisite to switch on tracing is a valid defined trace destination. We recommend defining it permanently, see `ERXMAIN` **macro** parameter `TRC1`.

1. Start the CICS transaction `ERXM` to call the RPC Online Maintenance Facility. See *Monitoring the RPC Server*.
2. Use the command `tracelevel=`*`tracelevel`*, where *`tracelevel`* is one of `None`, `Standard`, `Advanced` or `Support`. See `ERXMAIN` **macro** parameter `TRLV`.

Example: `tracelevel=Standard`

To evaluate CICS RPC Server return codes, see *EntireX RPC Server Return Codes* under *Error Messages and Codes*.

Console Commands for the RPC Server

The RPC Online Maintenance Facility `ERXM` can be used directly from a z/OS console using the z/OS command `MODIFY /F`. In the command syntax below:

- *`cics-name`* is the name of the CICS job
- *`erxmain-control-block`* is the name of the `ERXMAIN` **Control Block**. It can be omitted if the default name `ERXMAIN` is used.
- No blanks are allowed in the string provided to `ERXM`, for example `MEM=`*`erxmain-control-block`*`,CMD=...`

To start the CICS RPC Server from a z/OS console

- Use the following z/OS modify command:

```
F cics-name,ERXM [MEM=erxmain-control-block,]CMD=START
```

To ping the CICS RPC Server from a z/OS console

- Use the following z/OS modify command:

```
F cics-name,ERXM [MEM=erxmain-control-block,]CMD=PING
```

To stop the CICS RPC Server from a z/OS console

- Use the following z/OS modify command:

```
F cics-name,ERXM [MEM=erxmain-control-block,]CMD=STOP
```

To switch on tracing for the CICS RPC Server from a z/OS console

- Use the following z/OS modify command:

```
F cics-name,ERXM [MEM=erxmain-control-block,]CMD=TRACELEVEl=tracelevel
```

For *tracelevel*, see ***Activating Tracing for the RPC Server***.

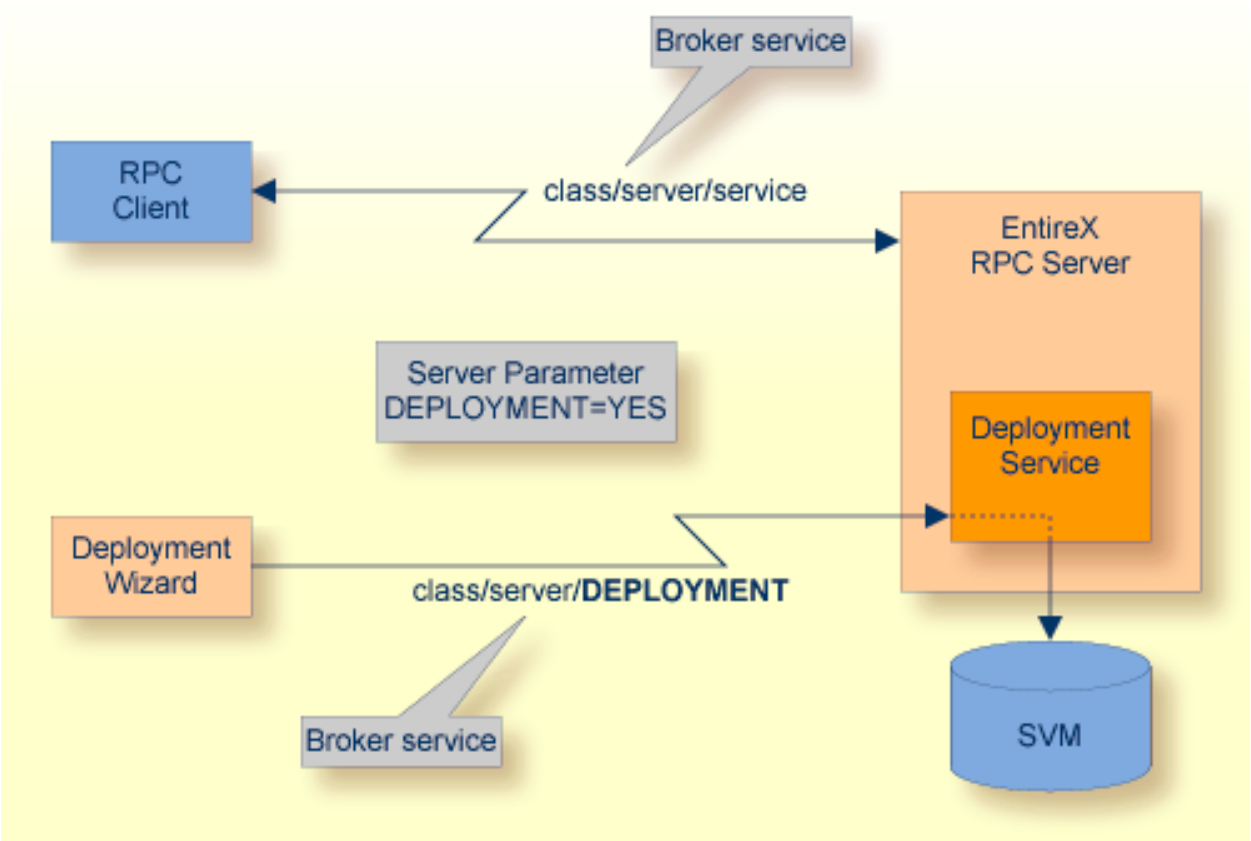
4 Deployment Service

Introduction

The deployment service is the (server-side) counterpart to the deployment wizard; see *Server Mapping Deployment Wizard*. It is a built-in service of the EntireX RPC server, which can be enabled/disabled by EntireX RPC server configuration settings.

Usage can be restricted to certain users or group of users, using EntireX Security; see *Authorization of Client and Server* under *Overview of EntireX Security* in the EntireX Security documentation.

You need to configure the deployment service only when server-side mapping files are used. There are also client-side server mapping files that do not need configuration here; see *Server Mapping Files for COBOL* in the EntireX Workbench documentation.



Scope

The deployment service is used in conjunction with the

- IDL Extractor for COBOL to deploy server-side mapping files with the deployment wizard;
- COBOL Wrapper for RPC server generation to deploy server-side mapping files with the deployment wizard.

See also *Deploying Server-side Mapping Files to the RPC Server*.

The deployment service uses the same class and server names as defined for the EntireX RPC server, and `DEPLOYMENT` as the service name, resulting in *`class`*`/`*`server`*`/DEPLOYMENT` as the broker service. Please note `DEPLOYMENT` is a service name reserved by Software AG. See broker attribute `SERVICE`.

Enabling the Deployment Service

To enable the deployment service

1 For a CICS RPC Server, the server-side mapping container (VSAM file) 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*.

2 Set *`ERXMAIN`* ***Macro*** parameter `DPLY=YES`. See `DPLY` under ***Configuring the RPC Server***.

3 Define in the broker attribute file, under the RPC service, an additional broker service with `DEPLOYMENT` as the service name and values for class and server identical to those used for the RPC service. For example, if your RPC service is named

```
CLASS = RPC    SERVER = SRV1    SERVICE = CALLNAT
```

the deployment service requires the following additional service definition in the broker attribute file:

```
CLASS = RPC    SERVER = SRV1    SERVICE = DEPLOYMENT
```

4 Optional. If you need to restrict the use of the deployment service to a selected group of users, use EntireX Security and define security rules for the *`class`*`/`*`server`*`/DEPLOYMENT` broker service. The service name `DEPLOYMENT` is a constant.

- For a z/OS broker, see *Resource Profiles in EntireX Security* in the EntireX Security documentation.

- For a UNIX or Windows broker, see *Administering Authorization Rules using System Management Hub* in the UNIX and Windows administration documentation.
- Not applicable to a BS2000/OSD or z/VSE broker.

Disabling the Deployment Service

To disable the deployment service

- Set *ERXMAIN Macro* parameter DPLY=NO. See ERXMAIN **macro** parameter DPLY.

  The CICS RPC Server will not register the deployment service in the broker.

5 Server-side Mapping Files

Server mapping enables the RPC server to correctly support special COBOL syntax such as `REDEFINE`s, `SIGN LEADING` and `OCCURS DEPENDING ON` clauses, `LEVEL-88` fields, etc. If one of these elements is used, the IDL Extractor for COBOL automatically extracts a server mapping file in addition to the IDL file (interface definition language). Also, the COBOL Wrapper may generate a server mapping file for RPC server generation. The server mapping is used at runtime to marshal and unmarshal the RPC data stream. There are client-side mapping files (EntireX Workbench files with extension .cvm) and server-side mapping files (Workbench files with extension .svm). If you have not used server-side mapping, we recommend you use client-side mapping. See *Server Mapping Files for COBOL* in the EntireX Workbench documentation.

See also *Source Control of Server Mapping Files* | *Comparing Server Mapping Files* | *When is a Server Mapping File Required?* | *Migrating Server Mapping Files* in the EntireX Workbench documentation.

Server-side Mapping Files in the RPC Server

Under z/OS, server-side mapping corresponds to lines of EntireX Workbench files with extension .svm. See *Server Mapping Files for COBOL*. The mapping information is stored as records within one VSAM file, the server-side mapping container. This container contains all server-side mapping entries from all EntireX Workbench files with extension .svm. The unique key of the VSAM file file consists of the first 255 bytes of the record: for the type (1 byte), for the IDL library (127 bytes) and for the IDL program (127 bytes).

If *one* server requires a server-side mapping file, you need to provide this to the RPC server:

- Development environments: to deploy new server-side mapping files, see ***Deploying Server-side Mapping Files to the RPC Server***.
- Production environments: provide a server-side mapping container (VSAM file) containing all required server-side mapping files to the RPC server. See `ERXMAIN` macro parameter `SVM`.

If *no* server requires server-side mapping, you can execute the RPC server without server mapping files:

- Development environments: you can disable the deployment service. See ***Disabling the Deployment Service***.
- Production environments: there is no need to provide a server-side mapping container (VSAM file) to the RPC server. See `ERXMAIN` macro parameter `SVM`.

Deploying Server-side Mapping Files to the RPC Server

The following approaches are available to deploy a server-side mapping file (EntireX Workbench file with extension .svm; see *Server Mapping Files for COBOL*):

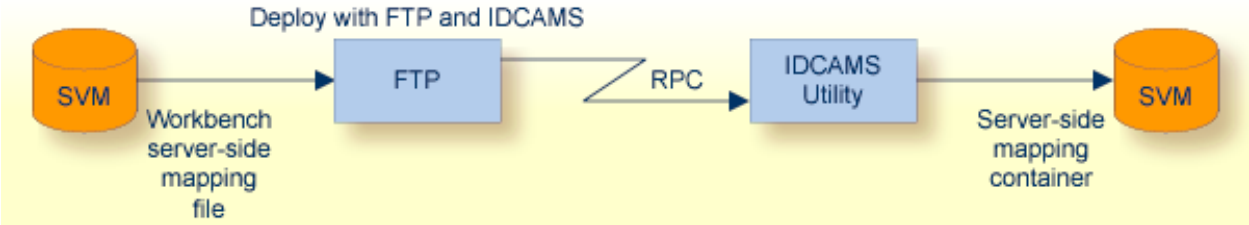
- Server Mapping Deployment Wizard
- FTP and IDCAMS

To deploy a server-side mapping file with the Server Mapping Deployment Wizard

1 Make sure your RPC server is active and that the Deployment Service of the RPC server is properly configured. See ***Deployment Service***.

2 From the context menu of your IDL file, choose **COBOL > Deploy/Synchronize Server Mapping** and call the Server Mapping Deployment Wizard. See *Server Mapping Deployment Wizard* in the EntireX Workbench documentation.

To deploy a server-side mapping file using FTP and IDCAMS



1 Make sure the server-side mapping container (VSAM file) is installed. See *Install the Server-side Mapping Container for a CICS RPC Server (Optional)* under *Installing the EntireX RPC Servers under z/OS*.

2 Allocate a target sequential file on your mainframe.

3 Allow write access to the VSAM file mentioned above and usage of IDCAMS tools.

4 Transfer the server-side mapping file to the target host, using FTP. You have to switch to text mode and the codepage of the FTP service must be the same as the codepage (locale string) of the RPC server used.

5 Install the server mapping contained in the server-side mapping file into the server-side mapping container (VSAM file) with an appropriate IDCAMS job.

```
//EXPSVMR  JOB (,,,999),ENTIREX,NOTIFY=&SYSUID,MSGLEVEL=(1,1),
//             CLASS=K,MSGCLASS=X,REGION=0M,COND=(0,LT)
//*-------------------------------------------------------*
//* FILL THE SVM VSAM CLUSTER                             *
//*-------------------------------------------------------*
//IMPORT   EXEC PGM=IDCAMS
//RECORDS  DD DISP=SHR,DSN=EXP.SVM.TARGET.SEQ.RECORDS
//SVM      DD DISP=SHR,DSN=EXP.SVM.KSDS
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
  REPRO -
    REPLACE -
    INFILE(RECORDS) -
    OUTFILE(SVM)
```

Note: If you omit the keyword `REPLACE` or define `NOREPLACE` in the `SYSIN` data stream of IDCAMS instead, existing server mapping information is not overwritten. This protects server-side mapping records from being overwritten by duplicates.

Undeploying Server-side Mapping Files from the RPC Server

Use the Server Mapping Deployment Wizard to undeploy a server-side mapping file (Workbench file with extension .svm). See *Server Mapping Files for COBOL*.

To undeploy a server-side mapping file with the Server Mapping Deployment Wizard

1 Make sure your RPC server is active and that the Deployment Service of the RPC server is properly configured. See ***Deployment Service***.

2 Make sure your IDL file is within an EntireX Workbench directory (folder) without the related server-side mapping file (.svm).

3 From the context menu of your IDL file, choose **COBOL > Deploy/Synchronize Server Mapping** and call the Server Mapping Deployment Wizard. See *Server Mapping Deployment Wizard* in the EntireX Workbench documentation. Because there is no related server-side mapping file in the Workbench, all server mapping information related to the IDL file in the RPC server will be removed.

Change Management of Server-side Mapping Files

Under z/OS, change management for a VSAM file (server-side mapping container, see ***Server-side Mapping Files in the RPC Server***) is similar to change management for a database. The complete VSAM file can be backed up at any time, for example by using IDCAMS. All updates to the VSAM file done after a backup must be kept.

All EntireX Workbench server-side mapping files (.svm) added since the last backup should be available. See *Server Mapping Files for COBOL* in the EntireX Workbench documentation.

List Deployed Server-side Mapping Files

Use IDCAMS to list the contents of the server-side mapping container. See ***Server-side Mapping Files in the RPC Server***.

```
//EXXPRINT JOB (,,,999),ENTIREX,NOTIFY=&SYSUID,MSGLEVEL=(1,1),
//             CLASS=K,MSGCLASS=X,REGION=0M
//*-------------------------------------------------------------*
//* PRINT CONTENTS OF AN SVM VSAM CLUSTER                       *
//*-------------------------------------------------------------*
//SVMPRINT EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//IN       DD DISP=SHR,DSN=ETS.SVM.KSDS
//OUT      DD SYSOUT=*
//SYSIN    DD *
  PRINT  -
    INFILE(IN) -
    DUMP | HEX | CHAR -
    OUTFILE(OUT)
/*
//
```

Use `DUMP` or `CHAR` format to print the server-side mapping records of the VSAM file.

Check if a Server-side Mapping File Revision has been Deployed

Server-side mapping records in the server-side mapping container correspond to lines of EntireX Workbench files with extension .svm. See *Server Mapping Files for COBOL* in the EntireX Workbench documentation. The records contain a creation timestamp at offset 276 (decimal) in the format `YYYYMMDDHHIISST`. Precision is 1/10 of a second. The creation timestamp can be checked.

The timestamp can be found on the same offset in the records in the server-side mapping container (VSAM file). See ***Server-side Mapping Files in the RPC Server***.

Access Control: Secure Server Mapping File Deployment

For deployment with the *Server Mapping Deployment Wizard*, use EntireX Security if the broker is running on platforms z/OS, UNIX , Windows or z/VSE. See ***Enabling the Deployment Service***.

For IBM deployment tool `IDCAMS`, use RACF to secure deployment.

Ensure that Deployed Server-side Mapping Files are not Overwritten

For IDCAMS, use the `NOREPLACE` option to disallow overwriting of duplicate server-side mapping records in the server-side mapping container (VSAM file); see ***Server-side Mapping Files in the RPC Server***. See also ***Deploying Server-side Mapping Files to the RPC Server***.

Is There a Way to Smoothly Introduce Server-side Mapping Files?

All EntireX RPC servers can be executed without server-side mapping files. See ***Server-side Mapping Files in the RPC Server***. There is no need to install the server-side mapping container if the following conditions are met:

- You do not use features that require server mapping; see *When is a Server Mapping File Required?*
- Server-side type of COBOL mapping is switched on in the EntireX Workbench. If you have not used server-side mapping, we recommend you use client-side mapping. See *Server Mapping Files for COBOL*.

You can also call COBOL servers generated or extracted with previous versions of EntireX mixed with a COBOL server that requires server-side mapping. All EntireX RPC servers are backward compatible.

6 Scenarios and Programmer Information

COBOL Scenarios

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. 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. 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

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. 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.

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. 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

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

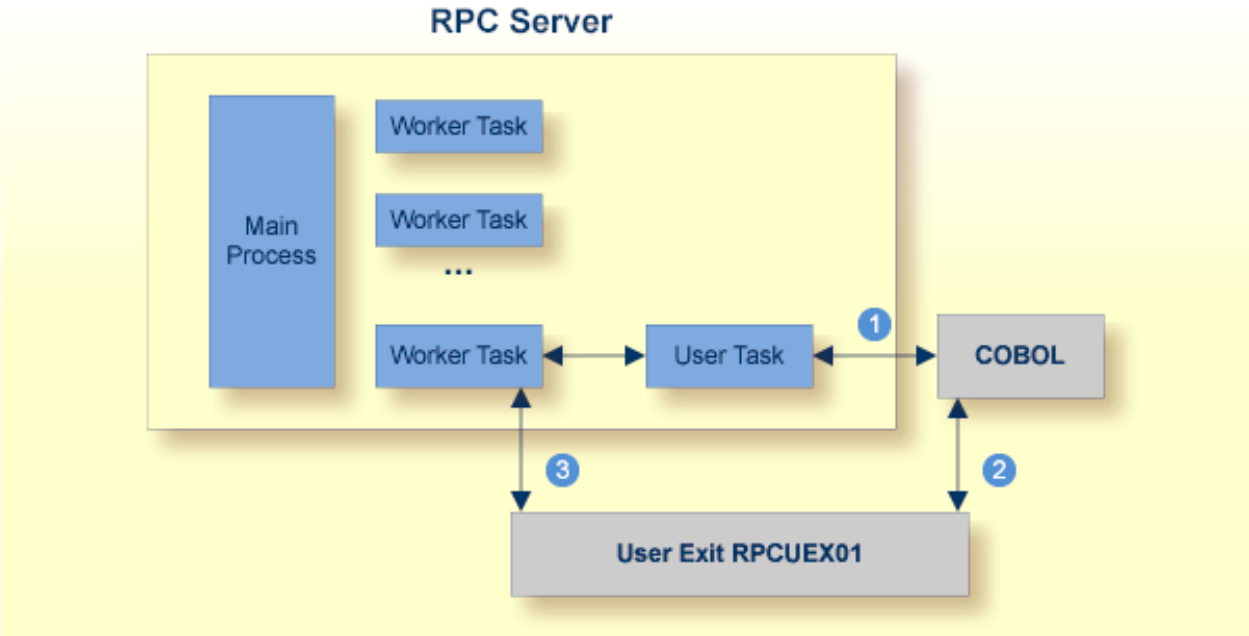
This approach applies to all CICS scenarios (all programming languages and all interface types) if impersonation is used (`YES|AUTO`). See ***Supported Interface Types*** and ***Impersonation***. If impersonation is not set, `EXEC CICS ABEND CANCEL` cannot be used.

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`***.

- 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 EXP990.SRCE as well as precompiled in the load data set EXP990.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 EXP990.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 EXP990.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.