software AG

webMethods EntireX

EntireX RPC Server for z/OS CICS®

Version 10.7

October 2020

WEBMETHODS

This document applies to webMethods EntireX Version 10.7 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-107-20220422

Table of Contents

EntireX RPC Server for z/OS CICS®

The EntireX RPC Server for z/OS CICS® 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.

1 About this Documentation

Document Conventions

Convention	Description
Bold	Identifies elements on a screen.
`Monospace font`	Identifies service names and locations in the format *`folder.subfolder.service`*, APIs, Java classes, methods, properties.
`Italic`	Identifies: Variables for which you must supply values specific to your own situation or environment. New terms the first time they occur in the text. References to other documentation sources.
`Monospace font`	Identifies: Text you must type in. Messages displayed by the system. Program code.
{ }	Indicates a set of choices from which you must choose one. Type only the information inside the curly braces. Do not type the { } symbols.
|	Separates two mutually exclusive choices in a syntax line. Type one of these choices. Do not type the | symbol.
[ ]	Indicates one or more options. Type only the information inside the square brackets. Do not type the [ ] symbols.
...	Indicates that you can type multiple options of the same type. Type only the information. Do not type the ellipsis (...).

Online Information and Support

Product Documentation

You can find the product documentation on our documentation website at **https://documentation.softwareag.com**.

In addition, you can also access the cloud product documentation via **https://www.softwareag.cloud**. Navigate to the desired product and then, depending on your solution, go to “Developer Center”, “User Center” or “Documentation”.

Product Training

You can find helpful product training material on our Learning Portal at **https://knowledge.softwareag.com**.

Tech Community

You can collaborate with Software AG experts on our Tech Community website at **https://tech-community.softwareag.com**. From here you can, for example:

- Browse through our vast knowledge base.
- Ask questions and find answers in our discussion forums.
- Get the latest Software AG news and announcements.
- Explore our communities.
- Go to our public GitHub and Docker repositories at **https://github.com/softwareag** and **https://hub.docker.com/publishers/softwareag** and discover additional Software AG resources.

Product Support

Support for Software AG products is provided to licensed customers via our Empower Portal at **https://empower.softwareag.com**. Many services on this portal require that you have an account. If you do not yet have one, you can request it at **https://empower.softwareag.com/register**. Once you have an account, you can, for example:

- Download products, updates and fixes.
- Search the Knowledge Center for technical information and tips.
- Subscribe to early warnings and critical alerts.
- Open and update support incidents.
- Add product feature requests.

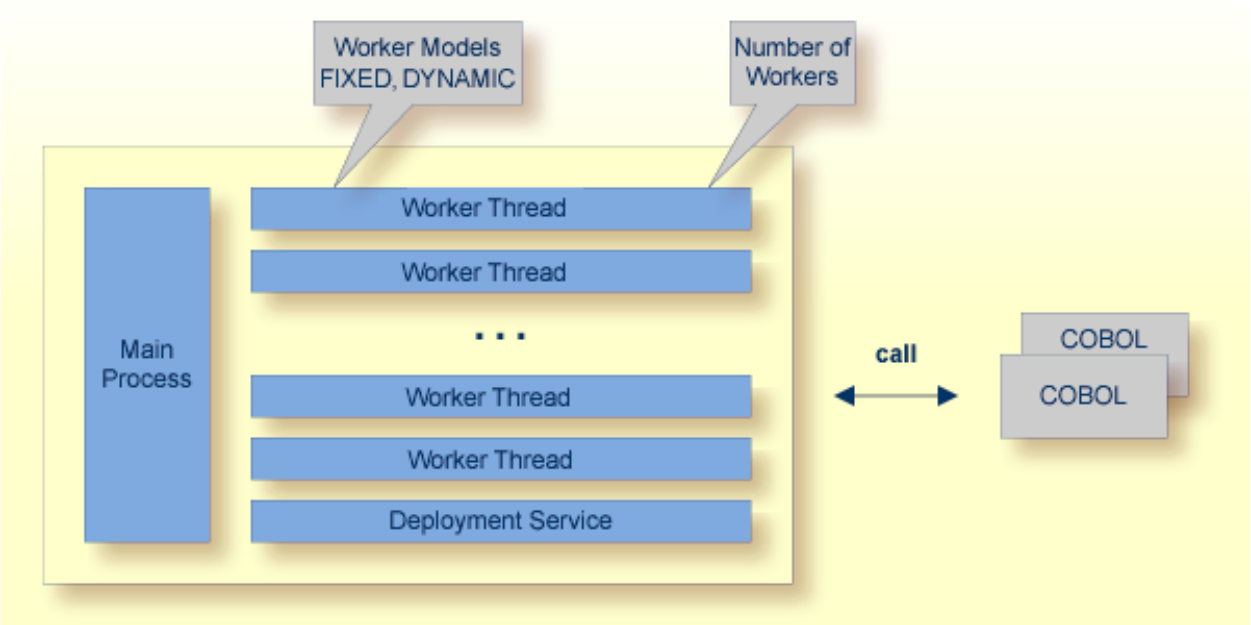
Data Protection

Software AG products provide functionality with respect to processing of personal data according to the EU General Data Protection Regulation (GDPR). Where applicable, appropriate steps are documented in the respective administration documentation.

2 Introduction to the RPC Server for CICS

The EntireX RPC Server for z/OS CICS® 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 threads, which are controlled by a main thread. Every RPC request occupies during its processing a worker thread. If you are using RPC conversations, each RPC conversation requires its own thread during the lifetime of the conversation. The RPC server provides two worker models:

- `FIXED`
  The *fixed* model creates a fixed number of worker threads. The number of worker threads does not increase or decrease during the lifetime of an RPC server instance.
- `DYNAMIC`
  The *dynamic* model creates worker threads depending on the incoming load of RPC requests.

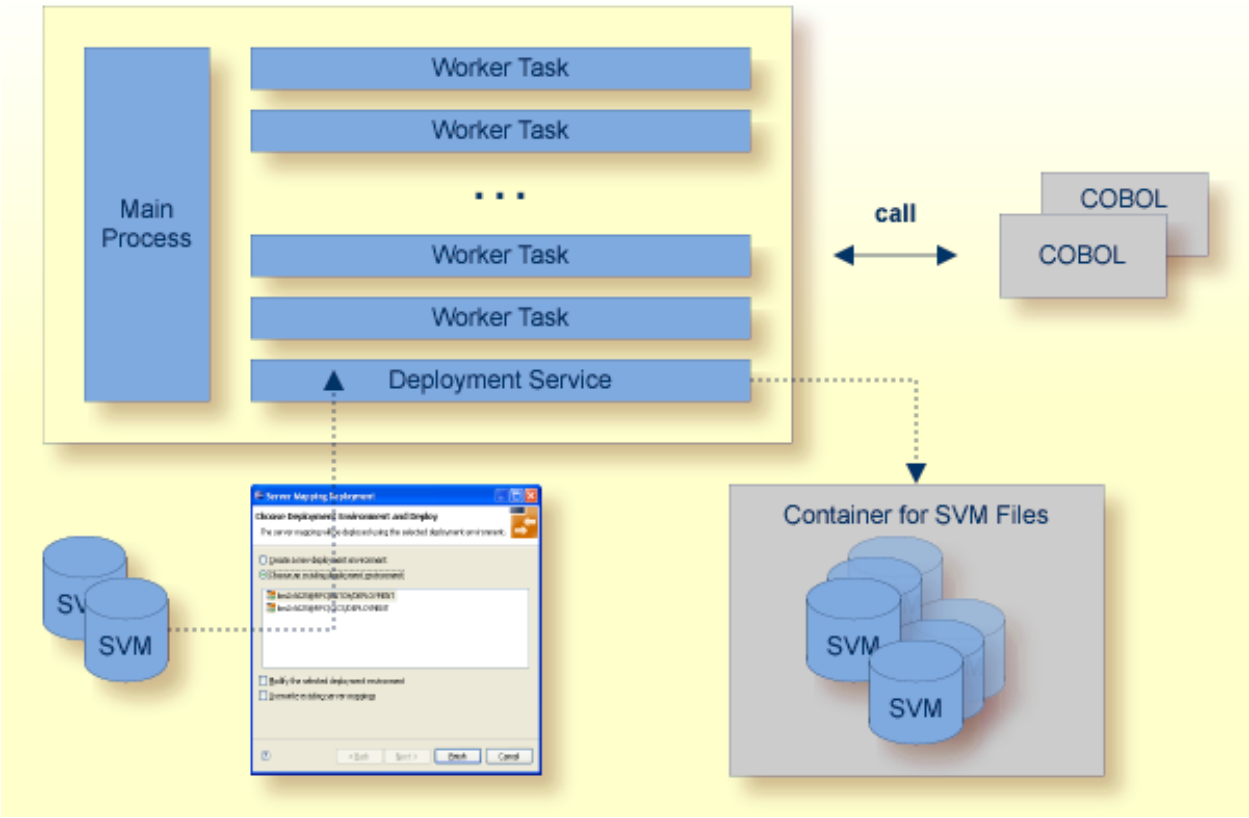
For configuration and technical details, see `ERXMAIN` **macro** parameter `ENDW` under *Administering the RPC Server for CICS*.

Inbuilt Services

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

Deployment Service

The Deployment Service allows you to synchronize server-side mapping files (Designer files with extension .svm) interactively using the *Server Mapping Deployment Wizard*. Synchronizing or un-deploying server mapping files from the RPC server is part of *Migrating Server Mapping Files*. 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.

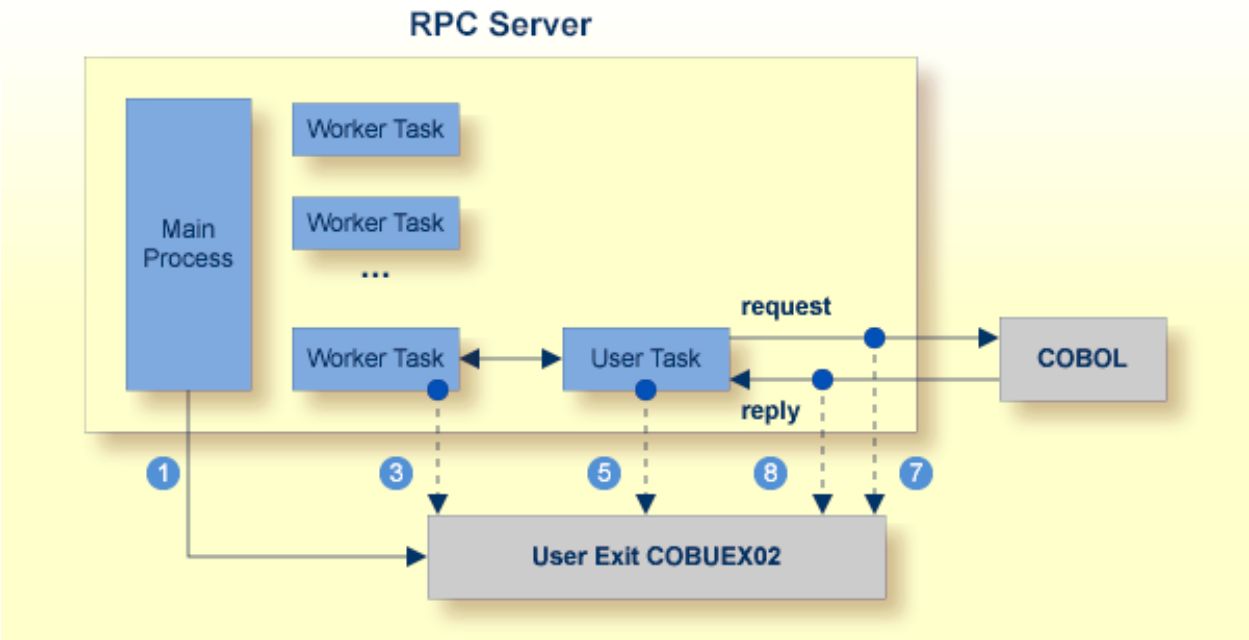


User Exit COBUEX02

The RPC Server for CICS 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:

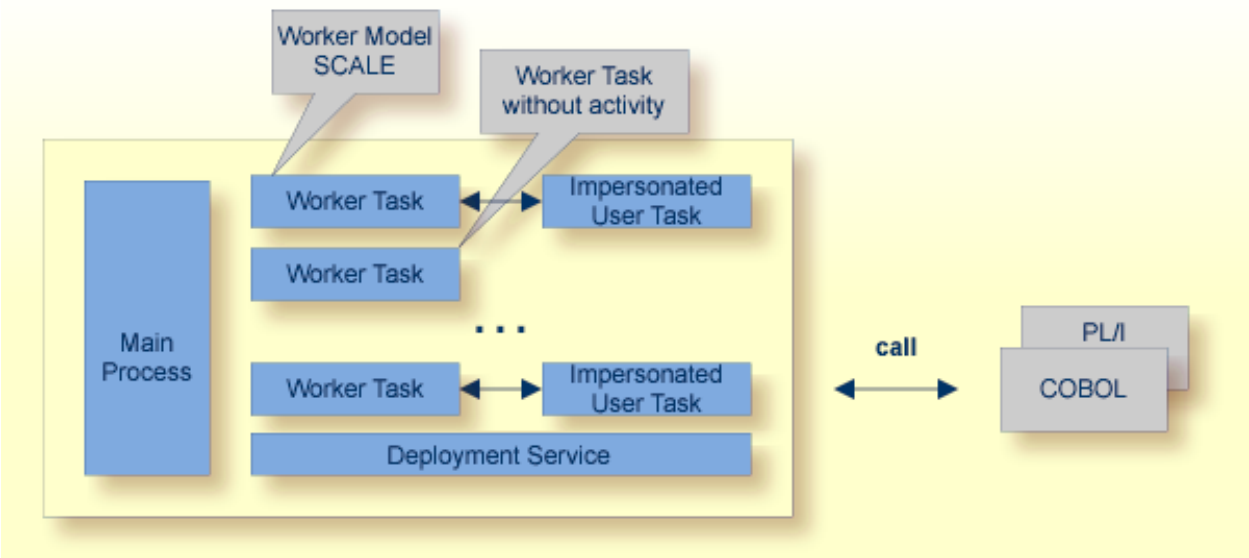
1. `WHICH-VERSION` event. Tells the RPC Server for CICS the version to use.
3. `START-WORKER` event. Allows you to set the CICS transaction ID.
5. `START-USER` event. Apply user ID, CICS transaction ID and CICS terminal ID to impersonated user tasks. See ***Impersonation***.
7. `CALL-START` event. Inspect, 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 RPC Server for CICS*.

Impersonation



The RPC Server for CICS 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 RPC Server for CICS 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 RPC Server for CICS uses the RPC user ID and RPC password sent by the calling RPC client for authentication and impersonation of the client. This means that the RPC server validates the RPC password or - if a long RPC password is sent - as a RACF password phrase.

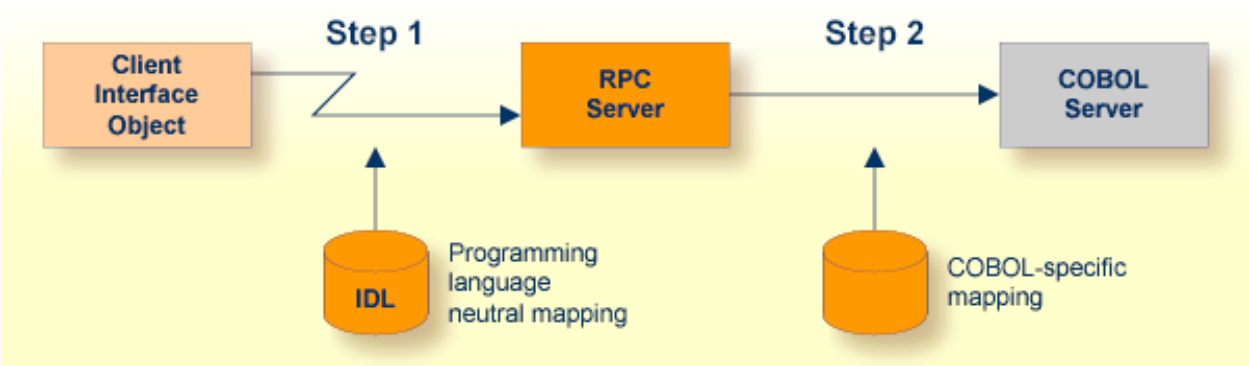
The picture above shows the configuration `IMPS=YES`.

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 threads, the slow-shrinking worker model `DYNAMIC` is used - value `TIMEOUT` is forced internally - any value given in the `ERXMAIN` **macro** parameter `ENDW` is ignored. The lifetime of worker threads 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

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. There are many situations where the RPC Server for CICS 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.



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 (Designer files with extension .cvm) may be retrieved as a result of the IDL Extractor for COBOL extraction process and the COBOL Wrapper if a COBOL server is generated. See *Server Mapping Files for COBOL* and *When is a Server Mapping File Required?*

Note: Server mapping files are used for COBOL only.

Supported Interface Types

The interface types supported by the RPC Server for CICS 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)

3 Administering the RPC Server for CICS

The EntireX RPC Server for z/OS CICS® 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

By default, the RPC Server for CICS runs as CICS transaction `ESRV`. This can be changed with parameter `REPL`. The following elements are used for setting up the RPC Server for CICS:

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

ERXMAIN Control Block

- defines a setup of the RPC Server for CICS 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`
  - 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 RPC Servers for CICS can be started and monitored in parallel. See *Installing Multiple EntireX RPC Servers in the same CICS (Optional)* under *RPC Servers* in the z/OS Installation documentation.

ERXMAIN Macro

- creates an *`ERXMAIN` Control Block*, a persistent setup of the RPC Server for CICS
- needs to be assembled to define a setup
- is defined in Assembler program `EMAINGEN` (in EXP107.SRCE) - use this for assembling; see *Build the ERXMAIN Control Block* in the z/OS installation documentation

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 RPC Server for CICS easily without the need to reassemble the ERXMAIN Control Block.
- runs as CICS transaction ERXM
- supports
  - starting
  - stopping
  - pinging
  - monitoring
  - activating trace

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

IBM LE Runtime Options

Depending on the feature the RPC Server for CICS 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 *IBM Z Publications Library Archive*.

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 RPC Server for CICS using application-specific `CSECT CEEUOPT`. The options can be changed if `CEEUOPT` is replaced on RPC Server for CICS 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 RPC Server for CICS. If this is the case, the TWA size set for the RPC Server for CICS 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*.	■ Use resource definition online (RDO), CICS transaction CEDA. ■ See member `DFHERX` in EXP107.SRCE data set. See *Updating the CICS Tables*.

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, that is, 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
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*). 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*. 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*. Case-sensitive, up to 32 characters. Corresponds to SERVICE attribute of the broker attribute file. Example: SVCN=MYSERVICE	R
CODE	codepage	no codepage transferred	The codepage tells the broker the encoding of the data. The application must ensure the encoding of the data matches the codepage. The RPC server itself does not convert your application data. The application's data is shipped and received as given. Often, the codepage must also match the encoding used in the RPC server environment for file and terminal IO, otherwise unpredictable results may occur. By default, no codepage is transferred to the broker. It is assumed the broker's locale string defaults match. See *Locale String Mapping* If they do not match, provide the codepage here. Example: CODE=ibm-273 Enable character conversion in the broker by setting the service-specific attribute CONVERSION to "SAGTRPC". See also	O

ERXMAIN Macro Syntax	RPC Online Maintenance Facility Commands	Default	Values	Req/ Opt
			Configuring ICU Conversion under *Configuring Broker for Internationalization* in the platform-specific Administration documentation. More information can be found under *Internationalization with EntireX*.	
COMP	compresslevel	N	Enforce compression when data is transferred between broker and server. See *Data Compression in EntireX Broker* in the platform-independent 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 RPC Server for CICS running while the broker is down for a short time. A restart cycle will be repeated every 60 seconds. **Note:** Internally, the server waits in periods of 10 seconds (performing six times more cycles), which you can see in the server output. When the number of specified cycles is reached and a connection to the broker is not possible, the RPC Server for CICS stops. Example: CYCL=30 The server waits up to 30 minutes (30*6*10 seconds) before it terminates due to a missing broker connection.	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 Designer 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.	O

ERXMAIN Macro Syntax	RPC Online Maintenance Facility Commands	Default	Values	Req/ Opt
			Example: DPLY=YES	
ENDW	endworker	TIMEOUT	NEVER — Defines worker model FIXED with a fixed number of worker threads. The number of worker threads is defined with ERXMAIN **macro** parameter MINW. It does not increase or decrease during the lifetime of an RPC server instance. TIMEOUT — Defines slow-shrinking worker model DYNAMIC, 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 DYNAMIC, 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 worker threads. ■ For worker model DYNAMIC: minimum number of active worker threads, even if no RPC client requests have to be processed. This allows you to define a certain number of worker threads - not used by the currently executing RPC request - to wait for new RPC client requests to process.	O

ERXMAIN Macro Syntax	RPC Online Maintenance Facility Commands	Default	Values	Req/ Opt
			In this way the RPC server is ready to handle many RPC client requests arriving at the same time. Do not set a value higher than `ERXMAIN` **macro** parameter `MAXW`. ■ For worker model `FIXED`: number of active worker threads. Do not set a value higher than 31 without adjusting `ERXMAIN` **macro** parameter `SIZE`. See also `ERXMAIN` **macro** parameter `ENDW`. Example: `MINW=2`	
`MAXW`	`maxworker`	`10`	Upper limit of worker threads and impersonated user tasks. ■ For worker model `DYNAMIC`: used to restrict the system load. Do not set a value higher than 31 without adjusting `ERXMAIN` **macro** parameter `SIZE`. See also `ERXMAIN` **macro** parameter `ENDW`. ■ For ***Impersonation***: worker threads 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* for available stubs. Example: `ETBL=CICSETB`	O
`EXIT`	n/a		At startup, the RPC Server for CICS will call the user exit to synchronize its version. If successful, the RPC Server for CICS will continue and call the user exit for the implemented events. See ***User Exit*** `COBUEX02`.	O
`IMPS`	`impersonation`	`NO`	Defines if RPC requests are executed under the RPC user ID of the calling RPC client. ■ For how to send the RPC user ID/password pair from an RPC client, see *Using the Broker and RPC User ID/Password* (.NET Wrapper | Java Wrapper | C Wrapper | PL/I Wrapper | DCOM Wrapper | Web Services Wrapper | IDL Tester | Listener for XML/SOAP | Listener for IBM MQ).	O

<table>
<tr><th>ERXMAIN Macro Syntax</th><th>RPC Online Maintenance Facility Commands</th><th>Default</th><th>Values</th><th>Req/ Opt</th></tr>
<tr><td></td><td></td><td></td><td>

For the COBOL Wrapper, refer to Using Broker Logon and Logoff and Using RPC Authentication (Natural Security, Impersonation, Integration Server).
For non-RPC clients, see Using the Broker and RPC User ID/Password under EntireX XML Tester in the XML/SOAP Wrapper documentation.

Depending on settings, different levels of checks are done prior to RPC server execution. See also Impersonation under Introduction to the RPC Server for CICS.

<code>impersonation=NO|YES|AUTO[,sameuser|,anyuser]</code>
<table>
<tr><td><code>NO</code></td><td>The RPC request is executed anonymously, which means the user ID of the calling RPC client is not used. RPC requests are executed under the user ID of the RPC server.</td></tr>
<tr><td><code>YES</code></td><td>The RPC request runs impersonated under the supplied RPC client user ID. For execution of the RPC request, the RPC Server for CICS starts a separate impersonated user task, that is, the client must be known to CICS and the supplied password is validated against CICS. The worker model <code>DYNAMIC</code> is forced; for details see Impersonation.</td></tr>
<tr><td><code>AUTO</code></td><td>Same as option <code>YES</code> above, except that no password validation is performed, that is, the calling RPC 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 <code>sameuser</code>
or
your own security implementation (option <code>anyuser</code> is supported for compatibility reasons if you need different broker and server user IDs - the customer-written security

</td></tr>
</table>
</td><td></td></tr>
</table>

ERXMAIN Macro Syntax	RPC Online Maintenance Facility Commands	Default	Values	Req/ Opt
			implementation must validate the calling RPC client using the *RPC client user ID*) `sameuser` The RPC Server for CICS 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. 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 for CICS*.	O

`OPTS`	`runoption`	`0`	This parameter is for special purposes. It provides the RPC Server for CICS 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

ERXMAIN Macro Syntax	RPC Online Maintenance Facility Commands	Default	Values	Req/ Opt
PSWD	password		The password for secured access to the broker. Example: PSWD=MyPwd	O
PRELOAD	preload	YES	Enable to call RPC Server for CICS with AMODE=24 YES Enable to call RPC server with AMODE 24 or 31. Internally the RPC Server for CICS 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 RPC Server for CICS 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 ***Introduction to the RPC Server for CICS***.	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:	O

ERXMAIN Macro Syntax	RPC Online Maintenance Facility Commands	Default	Values	Req/ Opt
			■ 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).	
SVM	svmfile		Usage and location of server-side mapping files (Designer files with extension .svm at runtime). See ***Server-side Mapping Files in the RPC Server***. If the SVM parameter is omitted, 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 at runtime, the server-side mapping container must be installed and configured; see *Installing the Server-side Mapping Container for an RPC Server for CICS (Optional)* under *Installing the EntireX RPC Servers under z/OS*. Server mapping files with extension .svm are no longer supported at design time by the Designer. You can still use them at runtime in a server-side mapping container. All special COBOL syntax and features supported by server mapping files with extension .svm are also covered by server mapping files with extension .cvm. See *When is a Server Mapping File Required?* We recommend migrating .svm files to .cvm files. See *Migrating Server Mapping Files* under *Server Mapping Files for COBOL* in the Designer documentation. 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

ERXMAIN Macro Syntax	RPC Online Maintenance Facility Commands	Default	Values	Req/ Opt
SYNC		Y	Determines whether a CICS SYNCPOINT COMMIT command is issued. YES Execute CICS SYNCPOINT COMMIT. If running *without Impersonation*, the server issues a SYNCPOINT COMMIT command after a successful non-conversational request or an end-of-conversation. NO Do not execute CICS SYNCPOINT COMMIT. If running *with Impersonation*, a SYNCPOINT COMMIT command is issued by CICS when the user task ends. This cannot be disabled. See also *Automatic Syncpoint Handling*.	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 and worker model DYNAMIC. Example: TOUT=300	O
TRC1	tracedestination	CSSL	Name of the destination for trace output. A valid CICS transient data queue. See also *Activating Tracing for the RPC Server for CICS*.	O
TRLV	tracelevel	0	Trace level for the server. See also *Activating Tracing for the RPC Server for CICS*. 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. Use only when requested by Software AG Support. Example: TRLV=standard	O
TROP	traceoption	none	Additional trace option if trace is active. See also *Activating Tracing for the RPC Server for CICS*. None No additional trace options. STUBLOG If tracelevel is Advanced or Support, the trace additionally activates the broker stublog.	O

ERXMAIN Macro Syntax	RPC Online Maintenance Facility Commands	Default	Values	Req/ Opt
			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)	
USER	userid	ERXSRV1	The user ID for access to the broker. The default ERXSRV1 will be used if this parameter is omitted or specified without a value: "USER=" Example: USER=MyUid	O

Locating and Calling the Target Server

The IDL library and IDL program names that come from the 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 RPC Server for CICS environment. Different mechanisms are used depending on the language:

- COBOL
- PL/I

COBOL

The CICS program name for the RPC server called is taken from the server mapping if one is available. See ***Usage of Server Mapping Files*** for an introduction. If no server mapping is used, the IDL program name is used as the CICS program name of the RPC server and the IDL library name is ignored.

To use the RPC Server for CICS with COBOL

- Make sure that all CICS programs called as RPC servers
  - use an interface type supported by the RPC Server for CICS 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

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 RPC Server for CICS with PL/I

- Make sure that all CICS programs called as RPC servers
  - use an interface type supported by the RPC Server for CICS 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, HTTP(S), and Certificates with EntireX* in the platform-independent Administration documentation.

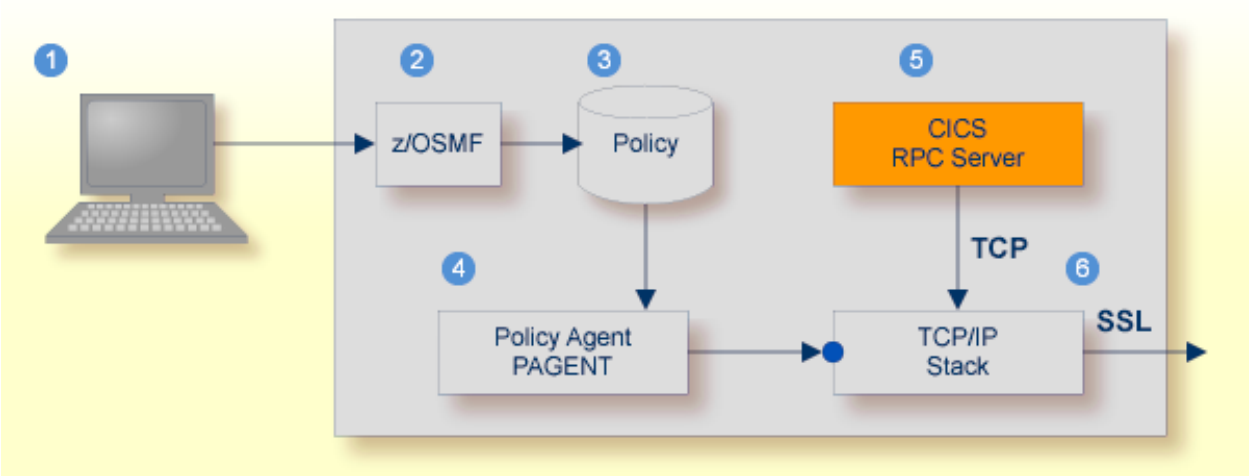
SSL delivered on a z/OS mainframe will typically use the Resource Access Control Facility (RACF) as the certificate authority (CA). Certificates managed by RACF can only be accessed through the RACF keyring container. A keyring is a collection of certificates that identify a networking trust relationship (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).

For establishing an SSL connection on z/OS, IBM's Application Transparent Transport Layer Security (AT-TLS) can be used, where the establishment of the SSL connection is pushed down the stack into the TCP layer.

With the RPC Server for CICS 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 *SSL/TLS Sample Certificates Delivered with EntireX* in the EntireX Security documentation.

2 Set up the RPC Server for CICS 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, using a client to interact with the z/OS Management Facility (z/OSMF). The outcome of this configuration is a Policy Repository with AT-TLS rules stored as z/OS files. This file is the configuration file for the Policy Agent, MVS task `PAGENT`.

4 Make sure the SSL server to which the RPC Server for CICS 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
- Broker SSL Agent in the UNIX | Windows Administration documentation
- *Support for SSL/TLS* in the EntireX Adapter documentation (for Direct RPC)

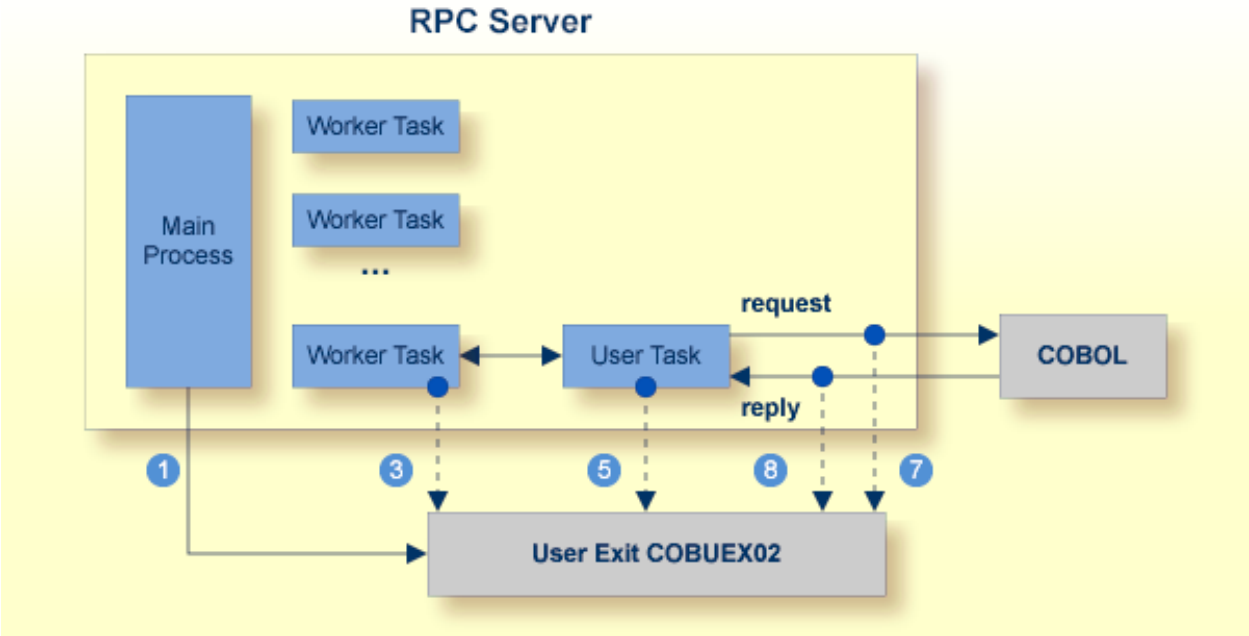
User Exit COBUEX02

The RPC Server for CICS 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 RPC Server for CICS 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

Step	Event	Called	Description	Note
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

EntireX provides the following resources for COBOL:

- User exit skeleton COBUEX02 in data set EXP107.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 EXP107.INCL. Add EXP107.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 RPC Server for CICS 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.

Parameter	Format	I/O	Description
			Example: Error code value *nnnn* results in the following error message: `1022nnnn error-text`. See *Message Class 1022 - RPC Server for CICS 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	RPC user ID supplied by calling RPC client. ■ For how to send the RPC user ID/password pair from an RPC client, see *Using the Broker and RPC User ID/Password* (.NET Wrapper | Java Wrapper | C Wrapper | PL/I Wrapper | DCOM Wrapper | Web Services Wrapper | IDL Tester | Listener for XML/SOAP | Listener for IBM MQ). ■ For the COBOL Wrapper, refer to *Using Broker Logon and Logoff* and *Using RPC Authentication (Natural Security, Impersonation, Integration Server)*. ■ For non-RPC clients, see *Using the Broker and RPC User ID/Password* under *EntireX XML Tester* in the XML/SOAP Wrapper documentation. 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 `VERSION`s. `CALL-END` All supported field `VERSION`s.

<table>
<tr><th>Parameter</th><th>Format</th><th>I/O</th><th>Description</th></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>
<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.</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>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><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>
</table>

Parameter	Format	I/O	Description
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 RPC Server for CICS 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 RPC Server for CICS will continue and call the user exit for the implemented events.

Autostart/Stop during CICS Start/Shutdown

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

To start the RPC Server for CICS during the initialization of CICS

1 If the COBOL source `ERXSTART` of the EntireX installation library EXP107.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 the RPC Server for CICS* in the z/OS Installation documentation.

To stop the RPC Server for CICS during the shutdown of CICS

1 If the COBOL source `ERXSTOP` of the EntireX installation library EXP107.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 the RPC Server for 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)* in the z/OS Installation documentation.

4 RPC Online Maintenance Facility

Monitoring the RPC Server for CICS

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  V107.0 ---          31/11/2020
                            RPC Broker Parameter

  Broker parameter
   Broker name    = ETB001
   Class name     = RPC
   Server name    = SRV2
   Service name   = CALLNAT

   User ID        = ERXSRV1
   Logon          = Yes

   Code page      =
   Server timeout = 60
   Compress level = 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  V107.0 ---          31/11/2020
                           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

    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  V107.0 ---          31/11/2020
                            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 for CICS

To start the RPC Server for CICS 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 for CICS*.
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 RPC Server for CICS 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**.

Alternatively, you can use the `start` command from the console. See *Console Commands for the RPC Server for CICS*.

Pinging the RPC Server for CICS

To ping the RPC Server for CICS 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 for CICS*.
2. Issue the command `ping` or use **PF09**.

Alternatively you can use the `ping` command from the console. See *Console Commands for the RPC Server for CICS*.

Stopping the RPC Server for CICS

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

1 Start the CICS transaction `ERXM` to call the RPC Online Maintenance Facility. See *Monitoring the RPC Server for CICS*.

2 Issue the `stop` command or use **PF10**. This ensures correct deregistration from broker and all worker threads are shut down.

Alternatively, you can use the `stop` command from the console. See *Console Commands for the RPC Server for CICS*.

Modifying Parameters of the RPC Server for CICS

With RPC Online Maintenance Facility commands, RPC Server for CICS 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 RPC Server for CICS, reassemble the `ERXMAIN` **Control Block** using the `ERXMAIN` **Macro**.

To modify the RPC Server for CICS 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 for CICS*.

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 for CICS

To switch on tracing for the RPC Server for CICS 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 for CICS*.

2 Temporarily change the trace level with the command `tracelevel=tracelevel`, where *tracelevel* is one of `None`, `Standard`, `Advanced` or `Support`. See `ERXMAIN` **macro** parameter `TRLV`.

   Example: `tracelevel=Standard`

3 Dynamically change the trace option with the command `traceoption=traceoption`, where *traceoption* is one of `None`, `STUBLOG` or `NOTRUNC`. See `ERXMAIN` **macro** parameter `TROP`.

   Example: `traceoption=NOTRUNC`

To evaluate RPC Server for CICS return codes, see *EntireX RPC Server Return Codes*.

Alternatively, you can issue the command from the console. See ***Console Commands for the RPC Server for CICS***.

Console Commands for the RPC Server for CICS

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 RPC Server for CICS 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 RPC Server for CICS 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 RPC Server for CICS 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 RPC Server for CICS 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 for CICS***.

5 Deployment Service

Introduction

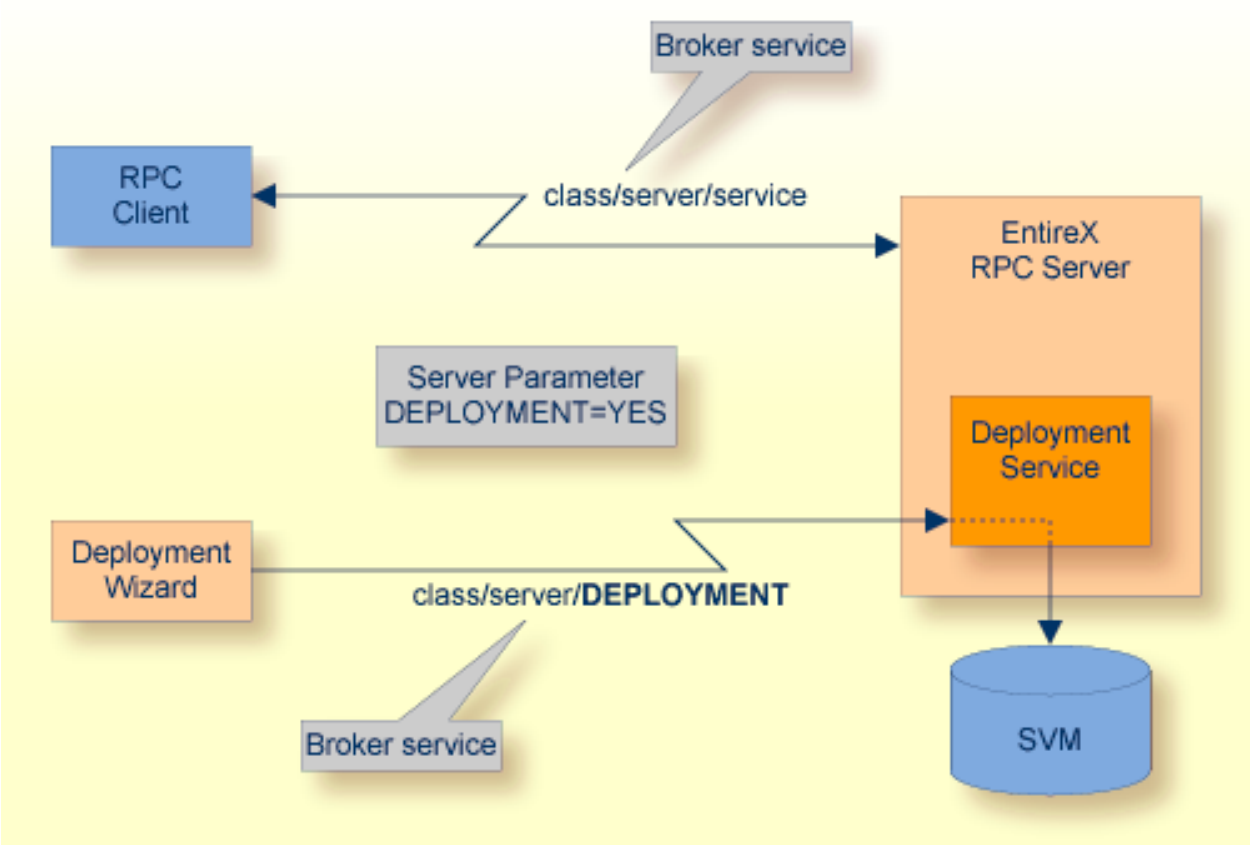
The deployment service is the (server-side) counterpart to the deployment wizard; see *Server Mapping Deployment Wizard*, the tool used when *Migrating Server Mapping Files*. It is a built-in service of the EntireX RPC server, which can be enabled/disabled by EntireX RPC server configuration settings.

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

You need to configure the deployment service only when server-side mapping files (Designer files with extension .svm) are used or you want to migrate them using the *Server Mapping Deployment Wizard*.

Server mapping files with extension .svm are no longer supported at design time by the Designer. You can still use them at runtime in a server-side mapping container. All special COBOL syntax and features supported by server mapping files with extension .svm are also covered by server mapping files with extension .cvm. See *When is a Server Mapping File Required?* We recommend migrating .svm files to .cvm files. See *Migrating Server Mapping Files* under *Server Mapping Files for COBOL* in the Designer documentation.

When migration is finished, disable the deployment service. See also *Server Mapping Files for COBOL* in the Designer documentation.



Scope

The deployment service is used to synchronize or undeploy server mapping files when *Migrating Server Mapping Files*.

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.

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 an RPC Server for CICS, the server-side mapping container (VSAM file) must be installed and configured. See *Installing the Server-side Mapping Container for an RPC Server for CICS (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*.
- For a UNIX or Windows broker, see *Authorization Rules* in the platform-independent Administration documentation.
- Not applicable to a BS2000 broker.

Disabling the Deployment Service

To disable the deployment service

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

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

6 Server-side Mapping Files

Server-side mapping files have the extension .svm.

Server mapping files with extension .svm are no longer supported at design time by the Designer. You can still use them at runtime in a server-side mapping container. All special COBOL syntax and features supported by server mapping files with extension .svm are also covered by server mapping files with extension .cvm. See *When is a Server Mapping File Required?* We recommend migrating .svm files to .cvm files. See *Migrating Server Mapping Files* under *Server Mapping Files for COBOL* in the Designer 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 Designer documentation.

Server-side Mapping Files in the RPC Server

Under z/OS, server-side mapping corresponds to lines of Designer 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 Designer 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 (.svm), you need to provide the server-side mapping container to the RPC server. See `ERXMAIN` macro parameter `SVM`.
- If *no* server requires server-side mapping (.svm) or you use server mapping files with extension .cvm only, you can execute the RPC server without the server-side mapping container.

Undeploying Server-side Mapping Files from the RPC Server

Use the Server Mapping Deployment Wizard to undeploy a server-side mapping file (Designer 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 a Designer directory (folder) without the related server-side mapping file (.svm) or renamed to extension .cvm if you are *Migrating Server Mapping Files*.

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 Designer documentation. Because there is no related server-side mapping file

(.svm) in the Designer, all server mapping information related to the IDL file in the server-side mapping container of 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 Designer server-side mapping files (.svm) added since the last backup should be available. See *Server Mapping Files for COBOL* in the Designer 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 Designer file with extension .svm. See *Server Mapping Files for COBOL* in the Designer 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 Usage of Server Mapping Deployment Wizard

To control the usage of the *Server Mapping Deployment Wizard*, use EntireX Security if the broker is running on platforms z/OS, UNIX or Windows. See ***Enabling the Deployment Service***.

7 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?* under *Server Mapping Files for COBOL* in the Designer 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 Designer 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?* under *Server Mapping Files for COBOL* in the Designer 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 Designer 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 Designer 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 Designer 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.

Returning Application Errors

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 RPC Server for CICS 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

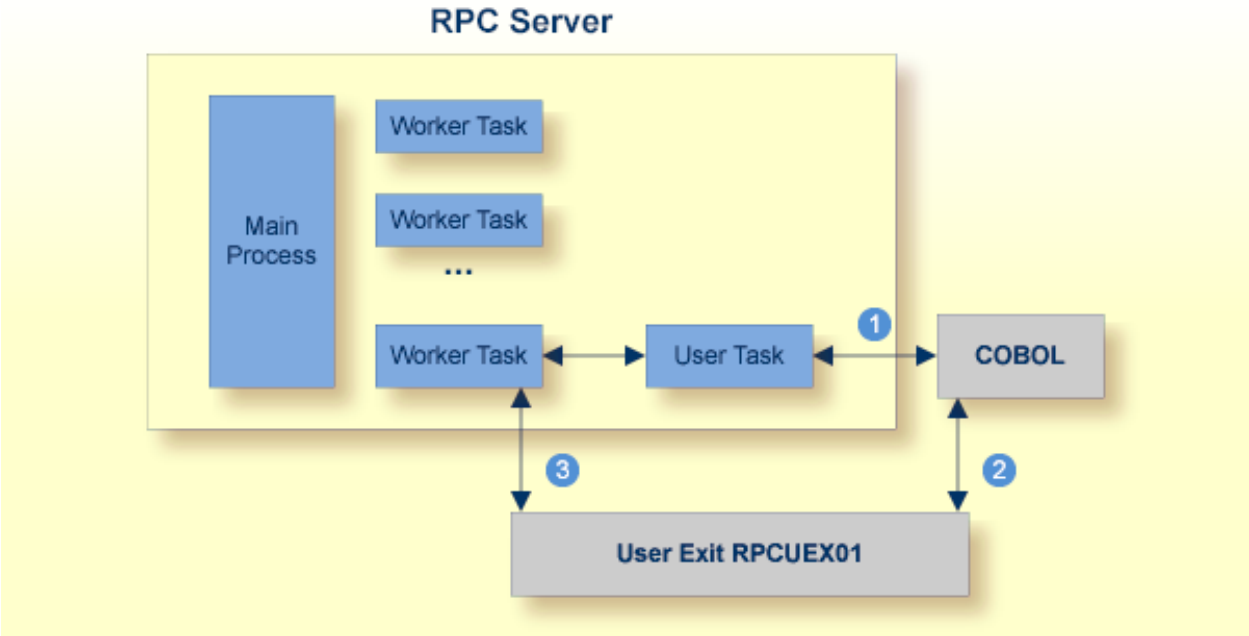
This approach applies to all CICS scenarios (all programming languages and all interface types) if impersonation is used (`YES|AUTO`). See ***Supported Interface Types*** and ***Impersonation***. If impersonation is not set, `EXEC CICS ABEND CANCEL` cannot be used.

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 RPC Server for CICS 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 RPC Server for CICS. Under these circumstances, IBM's Language Environment for C provides the application return code to EIBRESP2, where it can be detected by the RPC Server for CICS.

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 RPC Server for CICS. This module is available as source in the source data set EXP107.SRCE as well as precompiled in the load data set EXP107.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 EXP107.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 - 2020 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 RPC Server for CICS. 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 RPC Server for CICS. 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 EXP107.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 RPC Server for CICS issues a SYNCPOINT command under the following circumstances:

- If you are running under CICS *without* ***Impersonation***, the server issues a SYNCPOINT COMMIT command after a successful non-conversational request or an end-of-conversation. This can be disabled with the SYNC parameter.
- 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.
- 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.