

webMethods EntireX

EntireX z/VSE CICS® RPC Server

Innovation Release

Version 9.9

October 2015

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.

Copyright © 1997-2015 Software AG, Darmstadt, Germany and/or Software AG USA, Inc., Reston, VA, USA, and/or its subsidiaries and/or its affiliates and/or their licensors.

The name Software AG and all Software AG product names are either trademarks or registered trademarks of Software AG and/or Software AG USA, Inc. and/or its subsidiaries and/or its affiliates and/or their licensors. Other company and product names mentioned herein may be trademarks of their respective owners.

Detailed information on trademarks and patents owned by Software AG and/or its subsidiaries is located at <http://softwareag.com/licenses>.

Use of this software is subject to adherence to Software AG's licensing conditions and terms. These terms are part of the product documentation, located at <http://softwareag.com/licenses/> and/or in the root installation directory of the licensed product(s).

This software may include portions of third-party products. For third-party copyright notices, license terms, additional rights or restrictions, please refer to "License Texts, Copyright Notices and Disclaimers of Third-Party Products". For certain specific third-party license restrictions, please refer to section E of the Legal Notices available under "License Terms and Conditions for Use of Software AG Products / Copyright and Trademark Notices of Software AG Products". These documents are part of the product documentation, located at <http://softwareag.com/licenses> and/or in the root installation directory of the licensed product(s).

Use, reproduction, transfer, publication or disclosure is prohibited except as specifically provided for in your License Agreement with Software AG.

Document ID: EXX-CICSRPC-VSE-99-20171128

Table of Contents

| | |
|---|----|
| EntireX z/VSE CICS® RPC Server | v |
| 1 Introduction to the CICS RPC Server | 1 |
| Worker Models | 2 |
| Inbuilt Services | 4 |
| User Exit COBUEx02 | 6 |
| Impersonation | 7 |
| Usage of Server Mapping Files | 8 |
| Supported Interface Types | 9 |
| 2 Administering the EntireX RPC Server | 11 |
| Customizing the RPC Server | 12 |
| Configuring the RPC Server | 13 |
| Locating and Calling the Target Server | 22 |
| Using SSL/TLS with the RPC Server | 23 |
| User Exit COBUEx02 | 25 |
| Multiple RPC Servers in the same CICS | 28 |
| 3 RPC Online Maintenance Facility | 29 |
| Monitoring the RPC Server | 30 |
| Starting the RPC Server | 32 |
| Pinging the RPC Server | 33 |
| Stopping the RPC Server | 33 |
| Modifying Parameters of the RPC Server | 33 |
| Activating Tracing for the RPC Server | 34 |
| Console Commands for the RPC Server | 34 |
| 4 Deployment Service | 37 |
| Introduction | 38 |
| Scope | 39 |
| Enabling the Deployment Service | 39 |
| Disabling the Deployment Service | 40 |
| 5 Server-side Mapping Files | 41 |
| Server-side Mapping Files in the RPC Server | 42 |
| Deploying Server-side Mapping Files to the RPC Server | 43 |
| Undeploying Server-side Mapping Files from the RPC Server | 44 |
| Change Management of Server-side Mapping Files | 44 |
| List Deployed Server-side Mapping Files | 44 |
| Check if a Server-side Mapping File Revision has been Deployed | 45 |
| Access Control: Secure Server Mapping File Deployment | 45 |
| Ensure that Deployed Server-side Mapping Files are not Overwritten | 45 |
| Is There a Way to Smoothly Introduce Server-side Mapping Files? | 46 |
| 6 Scenarios and Programmer Information | 47 |
| COBOL Scenarios | 48 |
| Aborting RPC Server Customer Code and Returning Error to RPC Client | 49 |
| Automatic Syncpoint Handling | 49 |

EntireX z/VSE CICS® RPC Server

The EntireX z/VSE CICS® RPC Server allows standard RPC clients to communicate with RPC servers on the operating system z/VSE under CICS. It supports the programming language COBOL. It works together with the *COBOL Wrapper* and *IDL Extractor for COBOL*.

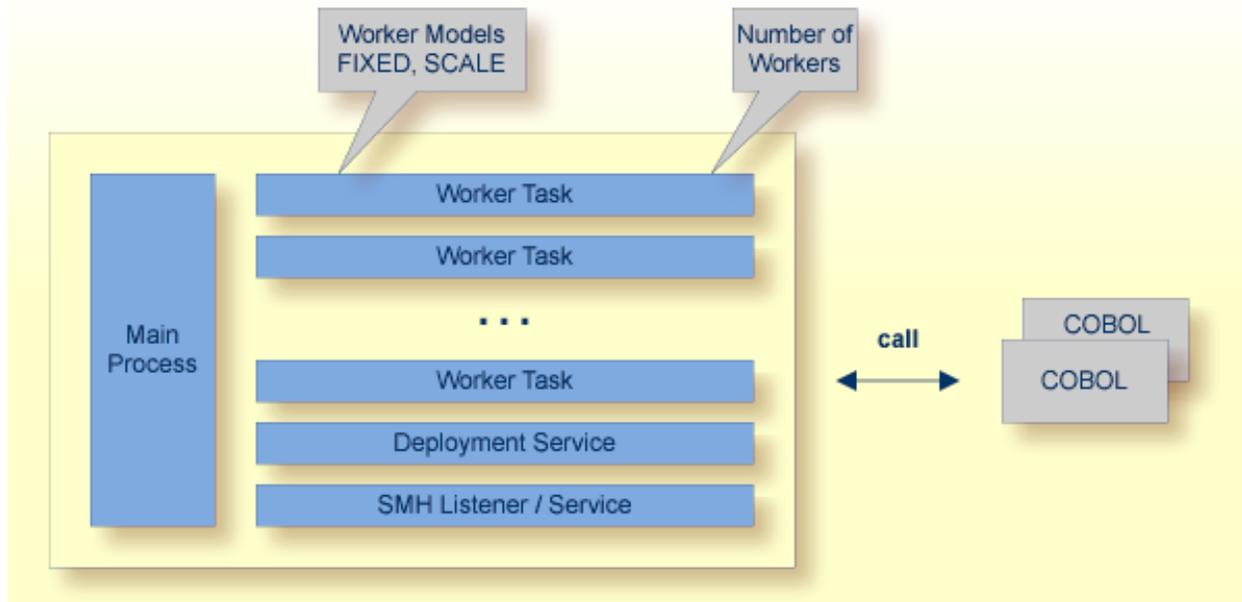
Supported compilers are listed under *z/VSE Prerequisites* in the *Release Notes*.

1 Introduction to the CICS RPC Server

| | |
|---------------------------------------|---|
| ▪ Worker Models | 2 |
| ▪ Inbuilt Services | 4 |
| ▪ User Exit COBUEX02 | 6 |
| ▪ Impersonation | 7 |
| ▪ Usage of Server Mapping Files | 8 |
| ▪ Supported Interface Types | 9 |

The EntireX z/VSE CICS® RPC Server allows standard RPC clients to communicate with RPC servers on the operating system z/VSE under CICS. It supports the programming language COBOL.

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
```

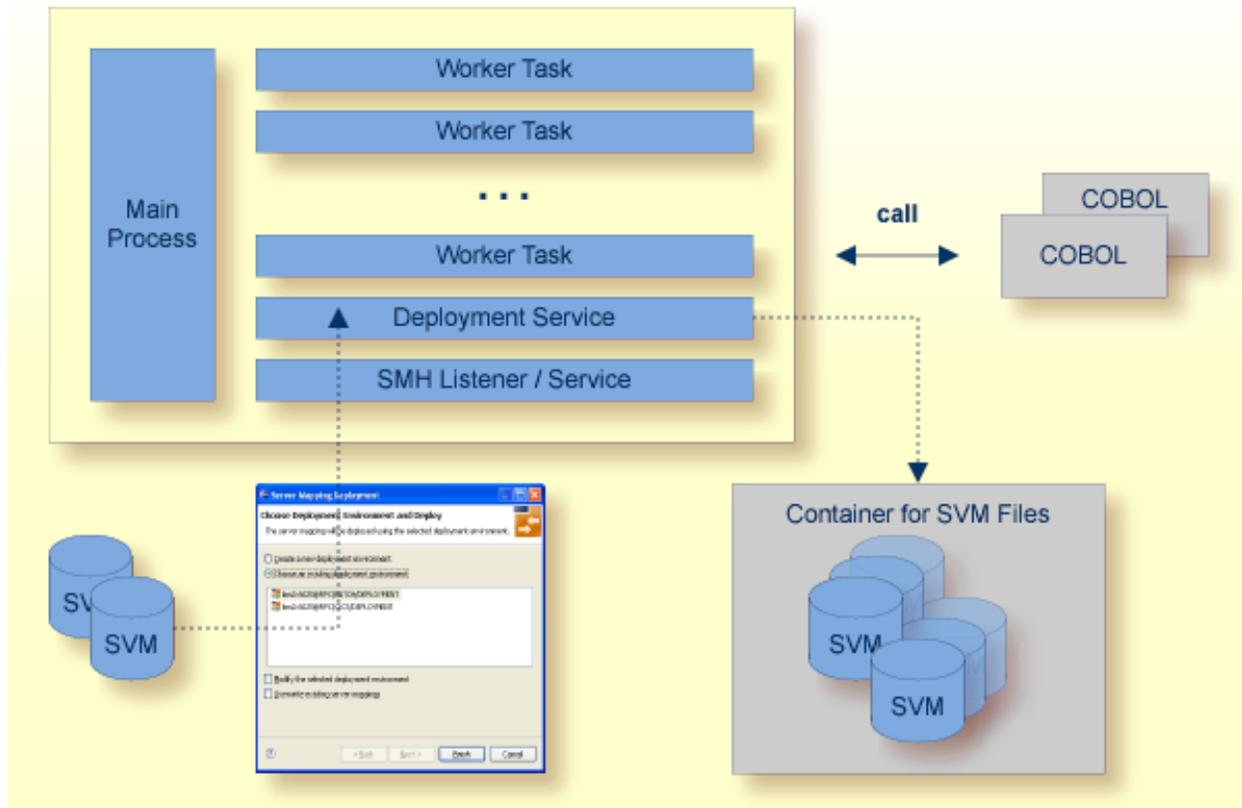
Inbuilt Services

CICS RPC Server provides several services for ease-of-use:

- Deployment Service
- SMH Listener Service

Deployment Service

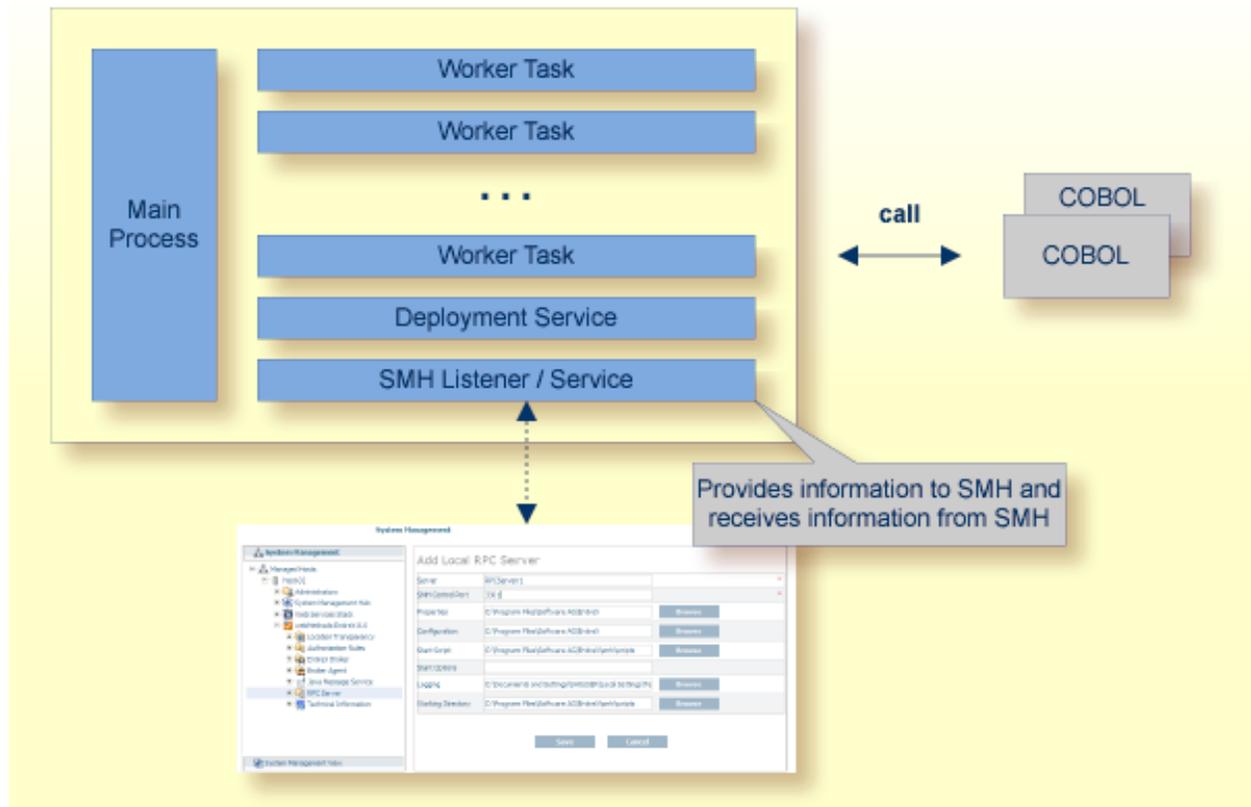
The Deployment Service allows you to deploy server-side mapping files (EntireX Workbench files with extension .svm) interactively using the *Server Mapping Deployment Wizard*. On the RPC server side, the server-side mapping files are stored in a server-side mapping container (VSAM file). See *Server-side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

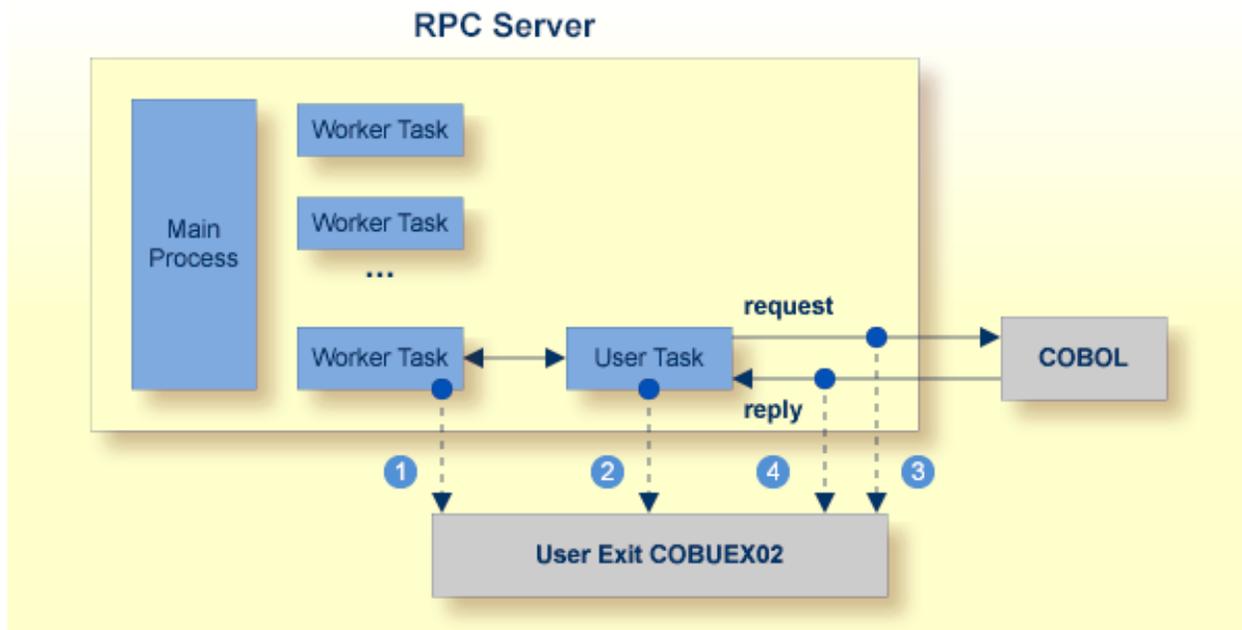
The SMH Listener Service is switched on if the SMH port number is set. See the [ERXMAIN macro](#) parameter [SMH](#) under *Configuring the RPC Server*.



User Exit COBUEX02

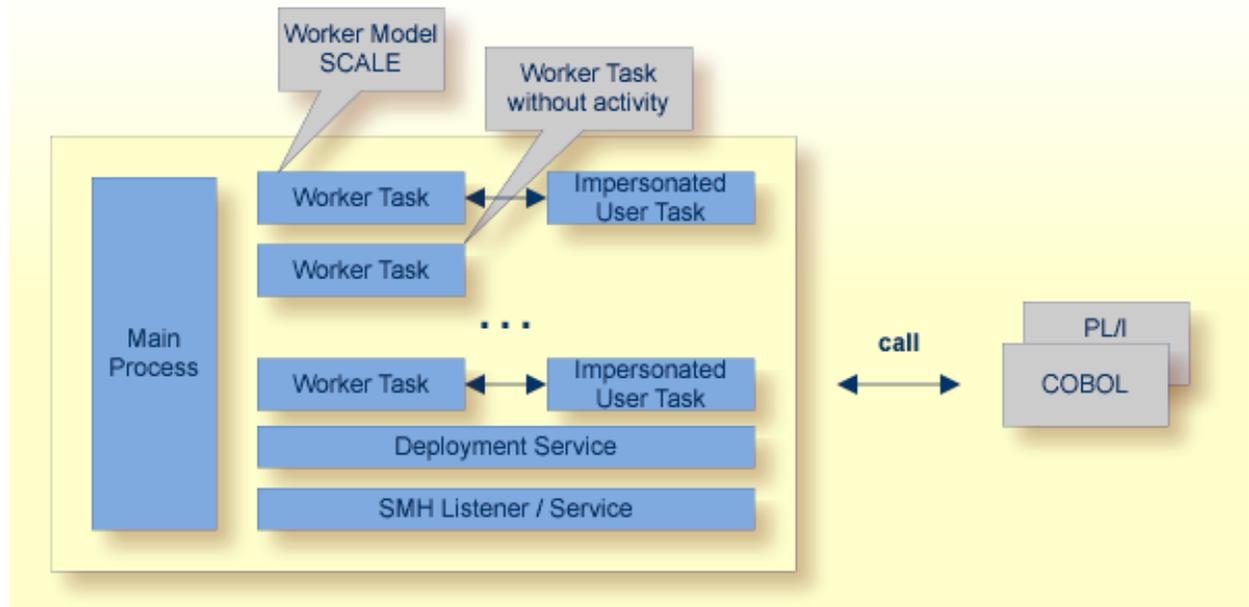
The CICS RPC Server provides a user exit `COBUEX02` to influence/control the RPC logic. The exit is called on the events `START-WORKER`, `START-USER`, `CALL-START` and `CALL-END`. The following tasks can be performed:

- 1 `START-WORKER` event before a CICS worker task is started. This allows you to programmatically set the CICS transaction ID.
- 2 `START-USER` event. Apply CICS transaction ID and user ID to impersonated user tasks. See [Impersonation](#).
- 3 `CALL-START` event. Inspect, modify or terminate the RPC request (payload) from the RPC client.
- 4 `CALL-END` event. Inspect or modify the RPC reply (payload) or give an error to the RPC client.



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**.

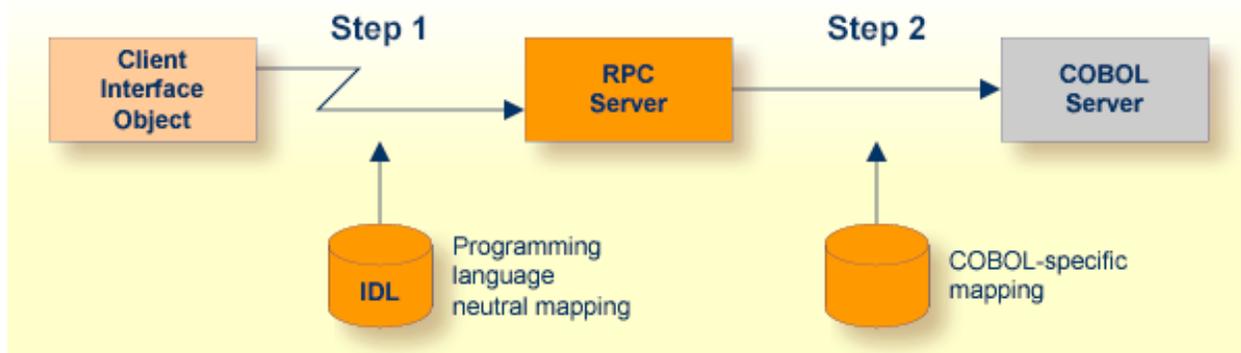
The lifetime of an impersonated user task starts when an open request for an RPC conversation or a non-conversational RPC request is received. It ends when the RPC conversation stops (after a commit operation or timeout) or when the non-conversational RPC request has been performed.

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 following interface types are supported by the CICS RPC Server:

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

See also [Locating and Calling the Target Server](#).

2 Administering the EntireX RPC Server

| | |
|--|----|
| ▪ Customizing the RPC Server | 12 |
| ▪ Configuring the RPC Server | 13 |
| ▪ Locating and Calling the Target Server | 22 |
| ▪ Using SSL/TLS with the RPC Server | 23 |
| ▪ User Exit COBUEX02 | 25 |
| ▪ Multiple RPC Servers in the same CICS | 28 |

The EntireX z/VSE CICS® RPC Server allows standard RPC clients to communicate with RPC servers on the operating system z/VSE under CICS. It supports the programming language COBOL.

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
- 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 sublibrary *EXP960*) - 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*.

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) | <ul style="list-style-type: none"> ■ Use resource definition online (RDO), CICS transaction CEDA. ■ Use job CICSDEF.J in sublibrary EXP960. See <i>Step 4: Update the CICS Tables</i>. |

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 | <code>brokerid</code> | ETB001 | <p>Broker ID used by the server. See <i>Using the Broker ID in Applications</i> in the RPC Programming documentation.</p> <p>Example: BKRN=myhost.com:1971</p> | R |
| CLZN | <code>class</code> | RPC | <p>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 <i>Service-specific Attributes (DEFAULTS=SERVICE)</i> under <i>Broker Attributes</i> in the platform-independent administration documentation). Case-sensitive, up to 32 characters. Corresponds to CLASS attribute of the broker attribute file.</p> <p>Example: CLZN=MyRPC</p> | R |
| SRVN | <code>servername</code> | SRV1 | <p>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 <i>Service-specific Attributes (DEFAULTS=SERVICE)</i> under <i>Broker Attributes</i> in the platform-independent administration documentation. Case-sensitive, up to 32 characters. Corresponds to SERVER of the broker attribute file.</p> <p>Example: SRVN=mySrv</p> | R |
| SVCN | <code>service</code> | CALLNAT | <p>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 <i>Service-specific Attributes (DEFAULTS=SERVICE)</i> under <i>Broker Attributes</i> in the platform-independent administration documentation. Case-sensitive, up to 32 characters. Corresponds to SERVICE attribute of the broker attribute file.</p> <p>Example: SVCN=MYSERVICE</p> | R |
| CODE | <code>codepage</code> | no codepage transferred | <p>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 <i>What is the Best Internationalization Approach to use?</i> under <i>Internationalization with EntireX</i> for information on which</p> | O |

| ERXMAIN Macro Syntax | RPC Online Maintenance Facility Commands | Default | Values | Req/ Opt |
|----------------------------|--|---------|---|-------------|
| | | | <p>internationalization approach requires a codepage (locale string).</p> <p>By default, no codepage is transferred to the broker. For the most popular internationalization approach, <i>ICU Conversion</i> under <i>Introduction to Internationalization</i>, the correct codepage (locale string) must be provided. This means it must:</p> <ul style="list-style-type: none"> ■ follow the rules described under <i>Locale String Mapping</i> 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. <p>Example: CODE=ibm-273</p> | |
| COMP | <u>compresslevel</u> | N | <p>Enforce compression when data is transferred between broker and server. See <i>Data Compression in EntireX Broker</i> in the general administration documentation.</p> <p>compresslevel= 0 1 2 3 4 5 6 7 8 9 Y <u>N</u></p> <p>0-9 0=no compression 9=max. compression <u>N</u> No compression. Y Compression level 6.</p> <p>Example: COMP=6</p> | O |
| CYCL | <u>restartcycles</u> | 15 | <p>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.</p> <p>When the number of cycles is reached and a connection to the broker is not possible, the RPC server stops.</p> <p>Example: CYCL=30</p> | O |
| DPLY | <u>deployment</u> | NO | <p>Activates the deployment service, see Deployment Service. Required to use the Server Mapping Deployment Wizard. See <i>Server Mapping Deployment Wizard</i> in the EntireX Workbench documentation.</p> | O |

| ERXMAIN Macro Syntax | RPC Online Maintenance Facility Commands | Default | Values | Req/ Opt |
|----------------------------|--|---------|--|-------------|
| | | | <p>YES Activates the deployment service. The RPC server registers the deployment service in the broker.</p> <p>NO The deployment service is deactivated. The RPC server does not register the deployment service in the broker.</p> <p>Example: DPLY=YES</p> | |
| ENCR | <u>encryptionlevel</u> | 0 | Deprecated. For encrypted transport we strongly recommend using the Secure Sockets Layer/Transport Layer Security protocol. See <i>SSL/TLS and Certificates with EntireX</i> . | O |
| ENDW | <u>endworker</u> | TIMEOUT | <p>NEVER Defines worker model FIXED with a fixed number of worker threads. The number of active workers is defined with ERXMAIN macro parameter MINW.</p> <p>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.</p> <p>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.</p> | O |

| ERXMAIN Macro Syntax | RPC Online Maintenance Facility Commands | Default | Values | Req/Opt |
|----------------------|--|---------|--|---------|
| | | | <p>This parameter is forced to value TIMEOUT if impersonation is switched on, see <i>Impersonation</i> and ERXMAIN macro parameter IMPS.</p> <p>Example: ENDW=IMMEDIATE , MINW=2 , MAXW=6</p> | |
| MINW | <u>minworker</u> | 1 | <p>Minimum limit of tasks active in parallel.</p> <ul style="list-style-type: none"> ■ 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. <p>See also ERXMAIN macro parameter ENDW.</p> <p>Example: MINW=2</p> | O |
| MAXW | <u>maxworker</u> | 10 | <p>Upper limit of tasks active in parallel.</p> <ul style="list-style-type: none"> ■ 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 <i>Impersonation</i>: 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. <p>Example: MAXW=2</p> | O |
| ETBL | <u>etblnk</u> | BKIMC | <p>Define the broker stub to be used. See <i>Administration of Broker Stubs under z/VSE</i> for available stubs.</p> <p>Example: ETBL=BKIMC</p> | O |
| EXIT | n/a | | <p>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 <i>User Exit COBUEX02</i>.</p> | O |
| IMPS | <u>impersonation</u> | NO | <p>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 <i>Impersonation</i>.</p> | O |

| ERXMAIN Macro Syntax | RPC Online Maintenance Facility Commands | Default | Values | | Req/ Opt |
|----------------------------|--|---------|---|--|-------------|
| | | | impersonation= <u>NO</u> YES AUTO [, <u>sameuser</u> , <u>anyuser</u>] | | |
| | | | 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 <i>RPC client user ID</i> . For execution of the RPC request, the CICS RPC Server 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 SCALE is forced; for details see Impersonation . | |
| | | | AUTO | <p>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</p> <ul style="list-style-type: none"> ■ 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 implementation must validate the RPC client using the <i>RPC client user ID</i>) | |
| | | | <code>sameuser</code> | The CICS RPC Server checks whether the <i>broker client user ID</i> matches the <i>RPC client user ID</i> . This is the default if AUTO is used. | |

| ERXMAIN Macro Syntax | RPC Online Maintenance Facility Commands | Default | Values | Req/ Opt | | | |
|----------------------------|--|---|--|-------------|---------|---|--|
| | | | <table border="1"> <tr> <td></td> <td>anyuser</td> <td>The <i>RPC client user ID</i> is used for impersonation. The <i>broker client user ID</i> is ignored.</td> </tr> </table> <p>Note:</p> <ol style="list-style-type: none"> EntireX supports two user ID/password pairs: a <i>broker client user ID/password</i> pair and an (optional) <i>RPC user ID/password</i> pair sent from RPC clients to the RPC server. With EntireX Security, the <i>broker client user ID/password</i> pair is checked. The <i>RPC user ID/password</i> 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. RPC clients send the (optional) <i>RPC user ID/password</i> pair in the same way as specifying the Natural user ID/password pair for a Natural RPC Server. See for example <i>Using Natural Security</i> in the respective section of the documentation. If the RPC client does not specify the optional <i>RPC user ID/password</i> pair, the <i>broker client user ID</i> is inherited to the <i>RPC user ID</i> and thus used for impersonation by the CICS RPC Server. <p>Example: IMPS=auto</p> | | anyuser | The <i>RPC client user ID</i> is used for impersonation. The <i>broker client user ID</i> is ignored. | |
| | anyuser | The <i>RPC client user ID</i> is used for impersonation. The <i>broker client user ID</i> is ignored. | | | | | |
| LOGN | <u>logon</u> | YES | <p>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 <i>Reliable RPC</i>.</p> <p>NO No logon/logoff functions are executed. <u>YES</u> Logon/logoff functions are executed.</p> <p>Example: LOGN=no</p> | O | | | |
| n/a | <u>mapname</u> | | Alias for command <i>memory</i> . | O | | | |
| n/a | <u>memory</u> | | Command to load an <i>ERXMAIN Control Block</i> . See <i>Modifying Parameters of the RPC Server</i> . | O | | | |

| ERXMAIN Macro Syntax | RPC Online Maintenance Facility Commands | Default | Values | Req/Opt |
|----------------------|--|---------|---|---------|
| OPTS | <u>runoption</u> | 0 | <p>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 representative provides you with an option and asks you to do so.</p> <p>Syntax: OPTS=(<u><option-list></u>) <u><option-list></u> = [<u><option-list></u>,] <u><option></u></p> <p>Example: OPTS=(RUNOPT1,RUNOPT2)</p> | O |
| PSWD | <u>password</u> | | <p>Password for broker logon. Case-sensitive, up to 32 characters. For more information see broker ACI control block field PASSWORD.</p> <p>Example: PSWD=MyPwd</p> | O |
| PRELOAD | <u>preload</u> | YES | <p>Enable to call CICS RPC Server with AMODE=24</p> <p>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.</p> <p>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.</p> | O |
| REPL | <u>replicatename</u> | ESRV | <p>CICS transaction ID (uppercase, up to 4 characters) assigned to worker tasks and as default for user tasks if <i>Impersonation</i> is set. In the START-USER event of the user exit (see <i>User Exit COBUEX02</i>) the CICS transaction ID for user tasks can be overridden. See also <i>Introduction to the CICS RPC Server</i>.</p> | O |
| SIZE | n/a | 32768 | <p>Upper limit of tasks active in parallel.</p> <ul style="list-style-type: none"> ■ For worker model SCALE: workers active in parallel. Do not set a value higher than 31 without adjusting | O |

| ERXMAIN Macro Syntax | RPC Online Maintenance Facility Commands | Default | Values | Req/Opt |
|----------------------|--|---------|--|---------|
| | | | <p>ERXMAIN macro parameter SIZE. See also ERXMAIN macro parameter ENDW.</p> <ul style="list-style-type: none"> For <i>Impersonation</i>: 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. <p>Example: MAXW=2</p> | |
| SMH | <code>smhport</code> | 0 | <p>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.</p> <p>See <i>SMH Listener Service</i> for more information.</p> <p>Example: SMH=3001</p> | O |
| SVM | <code>svmfile</code> | | <p>Usage and location of server-side mapping files. See <i>Server-side Mapping Files in the RPC Server</i>. 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 <i>Step 1: Define a Server-side Mapping Container - VSAMDEF.J (Optional)</i> under <i>Installing the z/VSE EntireX RPC Servers</i>. There are also client-side mapping files that do not require configuration here; see <i>Server Mapping Files for COBOL</i>.</p> <p>Syntax: SVM=NO <i>cicsname</i></p> <p><i>cicsname</i> The RPC server tries to open the server-side mapping container using the CICS file with name <i>cicsname</i>.</p> <p>no No server-side mapping files are used.</p> <p>Example: SVM=MYSVM</p> <p>See also <i>Usage of Server Mapping Files</i>.</p> | O |

| ERXMAIN Macro Syntax | RPC Online Maintenance Facility Commands | Default | Values | Req/Opt |
|----------------------|--|---------|---|---------|
| TOUT | <code>timeout</code> | 600 | <p>Timeout in seconds, used by the server to wait for broker requests. See broker ACI control block field <code>WAIT</code> for more information. Also influences <code>restartcycles</code>.</p> <p>See worker model <code>SCALE</code> to define the lifetime of worker threads in slow-shrinking worker model <code>SCALE</code>.</p> <p>Example: TOUT=300</p> | O |
| TRC1 | <code>tracedestination</code> | CSSL | Name of the destination for trace output. A valid CICS transient data queue. | O |
| TRLV | <code>tracelevel</code> | 0 | <p>Trace level for the server. See also Activating Tracing for the RPC Server.</p> <p>Syntax: TRLV= <code>None</code> <code>Standard</code> <code>Advanced</code> <code>Support</code></p> <p><code>None</code> No trace output. <code>Standard</code> For minimal trace output. <code>Advanced</code> For detailed trace output. <code>Support</code> This trace level is for support diagnostics and should only be switched on when requested by Software AG support.</p> <p>Example: TRLV=standard</p> | O |
| USER | <code>userid</code> | ERXSRV1 | <p>Used to identify the server to the broker. See broker ACI control block field <code>USER-ID</code>. Case-sensitive, up to 32 characters.</p> <p>Example: USER=MyUid</p> | 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.

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*.

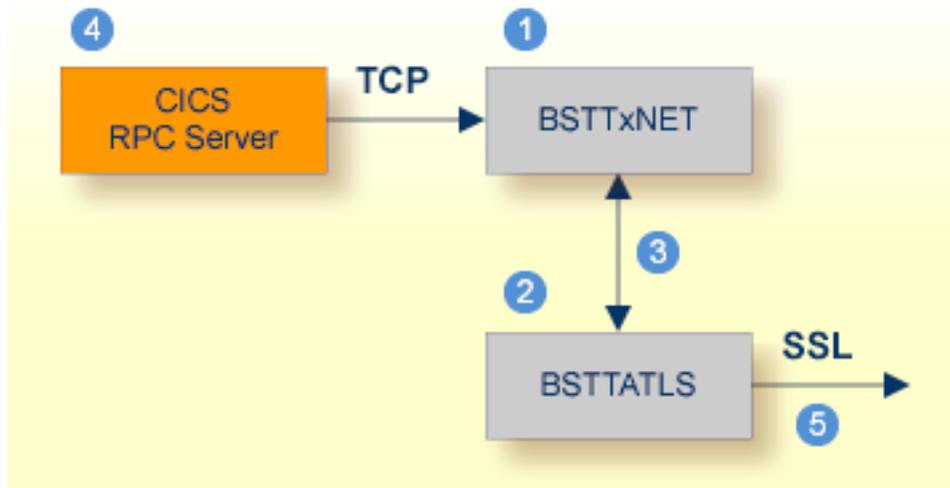
Using SSL/TLS with the RPC Server

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.

Establishing an SSL connection on z/VSE requires BSI's Automatic Transport Layer Security (ATLS). This facility is similar to z/OS Application Transparent - Transport Layer Security (AT-TLS). ATLS is supported by the BSI stack only.

Using BSI's Automatic Transport Layer Security (ATLS)

Together with SSL parameters (to provide certificates), define ATLS rules for socket interception in the ATLS daemon startup job `BSTTATLS` ②. If the rules match, the socket connection is turned into an SSL connection ⑤. Refer to your IBM documentation for further information. For an overview, refer to the IBM Redbook *Enhanced Networking on IBM z/VSE*; for a more detailed description, refer to *BSI SSL Installation, Programming and User's Guide*.



- ① BSI TCP/IP Stack, either `BSTTINET` (IPv4) or `BSTT6NET` (IPv6).
- ② ATLS rules are defined manually. See *Sample ATLS Daemon Configuration* below.
- ③ `BSTTATLS` is associated with a TCP/IP stack.
- ④ Application using TCP connection.
- ⑤ `BSTTATLS` intercepts outbound TCP connection and converts it to SSL connection. For inbound, SSL connections can also be intercepted and converted to TCP connections.

➤ To set up SSL with ATLS

- 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 ATLS 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
 - *Setting 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)

Sample ATLS Daemon Configuration

```
* Converting inbound EntireX Broker connection
* Converts listen port 1971 to SSL listen port 1972
OPTION SERVER
ATTLS 1971 AS 2071 SSL
*
* Converting outbound client connection
* Converts connect to 192.168.2.100:1972:TCP to 192.168.2.100:2072:SSL
OPTION CLIENT
ATTLS 1972 TO 192.168.2.100 AS 2072 SSL
```



Note: We recommend setting SETPARM value SUBTASK to a value greater than 0 in the ATLS daemon startup job (valid values 0-16, default=0). For example:

```
// SETPARM SUBTASK=8
```

See also *BSI SSL Installation, Programming and User's Guide*.

User Exit COBUEX02

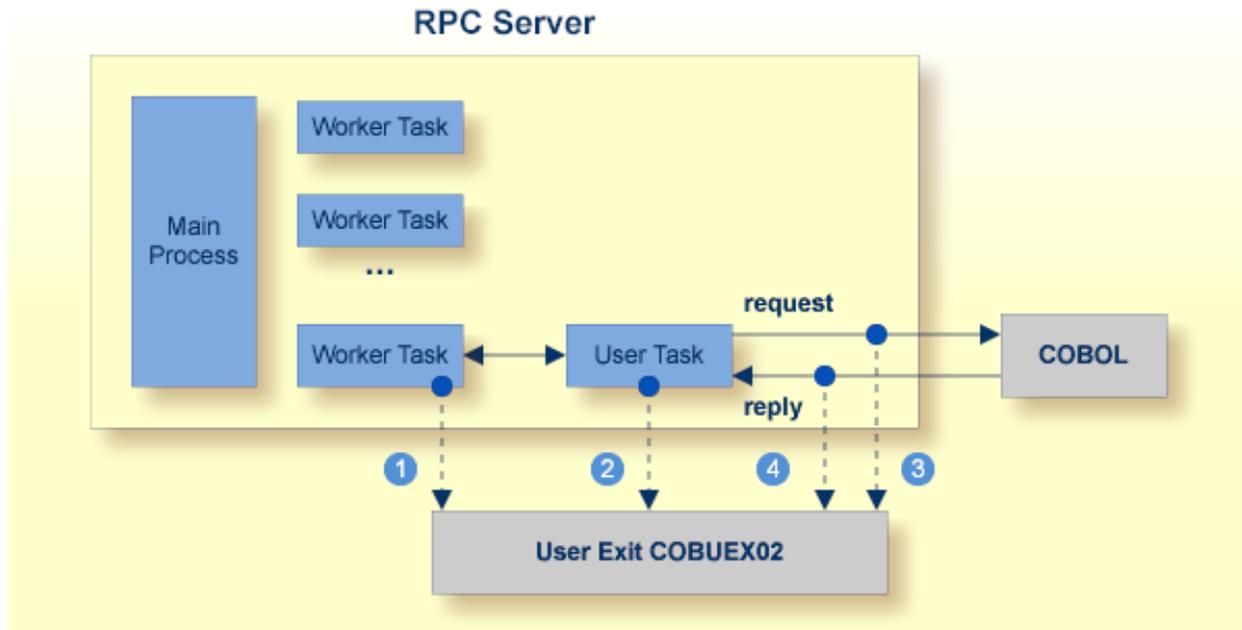
The CICS RPC Server provides a user exit COBUEX02 to influence/control the RPC logic.

- [User Exit Events](#)
- [Writing the User Exit](#)

- [Configuring the User Exit](#)

User Exit Events

The user exit is called on the following events:



- 1 **START-WORKER** event before a CICS worker task is started. This allows you to programmatically set the CICS transaction ID. You can terminate an RPC request by specifying an *ERROR-CODE* and optional *ERROR-TEXT*.
- 2 **START-USER** event. Before an impersonated CICS transaction (worker task) is started, the user exit may change the user ID and CICS transaction ID of the new impersonated worker. See *Impersonation*. You can terminate an RPC request by specifying an *ERROR-CODE* and optional *ERROR-TEXT*.
- 3 **CALL-START** event. The RPC request (payload data from the RPC client to the RPC server) may be inspected and modified. You can terminate an RPC request by specifying *ERROR-CODE* and optional *ERROR-TEXT*.
- 4 **CALL-END** event. The RPC reply (payload data from the RPC server to the RPC client) may be inspected and modified. If an *ERROR-CODE* and optional *ERROR-TEXT* is given in the API, this error is returned to the RPC client instead of the payload.

Writing the User Exit

The Developer's Kit RPC source data set EXP960.SRCE of the EntireX CICS installation provides the user exit skeleton COBUEX02 for COBOL. Copy this skeleton so you have your own user exit source for modifications.

Accordingly, a COBOL copybook COBUEX02 is provided in EXP960.INCL. Please add this library to your COBOL compiler SYSLIB DD chain.

The sublibrary EXP960 of the EntireX CICS installation provides the user exit skeleton COBUEX02.C for COBOL. Copy this skeleton so you have your own user exit source for modifications.

Accordingly, a COBOL copybook COBUEX02.CPY is provided in EXP960.

Hint: The copybook extension CPY might not be understood by the z/VSE COBOL compiler. In this case, copy COBUEX02.CPY into your copybook library and rename it to COBUEX02.C. Add this library to your COBOL compiler LIBDEF chain.

The most important API parameters of the user exit are described below. Other parameters are informational and are described in the source code. The user exit program must comply with the EXEC CICS LINK PROGRAM COMMAREA conventions.

| Parameter | Description |
|---------------|--|
| VERSION | Required for future changes. Do not change the skeleton code. |
| ERROR-CODE | You can terminate the current request: Any number between 1 and 9999 will cause the CICS RPC Server to stop execution of the current RPC request and pass back the given error code with message class 1022 to the RPC client. See <i>Message Class 1022 - CICS RPC Server User Exit Messages</i> under <i>Error Messages and Codes</i> . With error code 0000, the CICS RPC Server continues as normal. |
| ERROR-TEXT | If the error code is not zero, an error text of up to 256 characters may be applied. This is passed to the RPC client. |
| CICS-TRANSID | Can be applied in the event START-USER, otherwise it is informational. Apply the TRANSID that your business logic requires. |
| CICS-TERMINID | Can be applied in the event START-USER, otherwise it is informational. In some (rare) cases, RPC server routines require a terminal ID. Apply the TERMINID that your business logic requires. |
| USERID | Can be applied in the event START-USER otherwise it is informational. Under some circumstances, it might be necessary to change the original RPC-USERID from the calling RPC client. |
| DATA-POINTER | This pointer refers to the payload data for the events CALL-START and CALL-END. The payload to which this pointer is pointing may be inspected as well as modified. The pointer itself must not be changed. |

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.

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/VSE installation documentation.

3 RPC Online Maintenance Facility

- Monitoring the RPC Server 30
- Starting the RPC Server 32
- Pinging the RPC Server 33
- Stopping the RPC Server 33
- Modifying Parameters of the RPC Server 33
- Activating Tracing for the RPC Server 34
- Console Commands for the RPC Server 34

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  V960.0 ---          06/02/2014
                    RPC Broker Parameter

Broker parameter
Broker name      = ETB001:SVC045:NET
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          = BKIMC

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  V960.0 ---      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  V960.0 ---      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:SVC045:NET
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**.

Alternatively, you can use the `start` command from the console. See [Console Commands for the RPC Server](#).

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.

Alternatively, you can use the stop command from the console. See [Console Commands for the RPC Server](#).

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/VSE console using the syntax below:

- `task_id` is the name of the CICS task ID
- `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/VSE console

- Use the following z/VSE command:

```
task_id ERXM [MEM=erxmain-control-block,]CMD=START
```

➤ To ping the CICS RPC Server from a z/VSE console

- Use the following z/VSE command:

```
task_id ERXM [MEM=erxmain-control-block,]CMD=PING
```

➤ **To stop the CICS RPC Server from a z/VSE console**

- Use the following z/VSE command:

```
task_id ERXM [MEM=erxmain-control-block,]CMD=STOP
```

➤ **To switch on tracing for the CICS RPC Server from a z/VSE console**

- Use the following z/VSE command:

```
task_id ERXM [MEM=erxmain-control-block,]CMD=TRACELEVEL=tracelevel
```

For *tracelevel*, see [Activating Tracing for the RPC Server](#).

4 Deployment Service

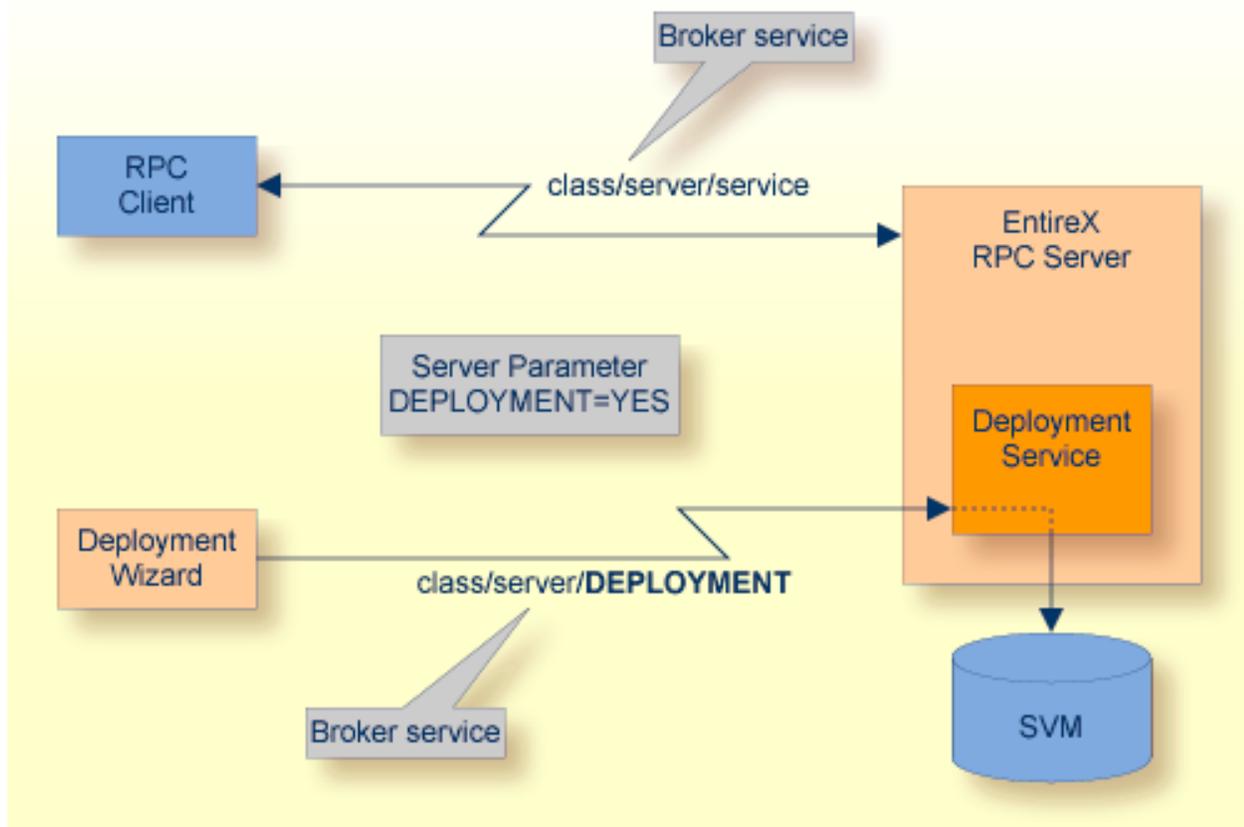
- Introduction 38
- Scope 39
- Enabling the Deployment Service 39
- Disabling the Deployment Service 40

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 *Step 1: Define a Server-side Mapping Container - VSAMDEF.J (Optional)* under *Installing the z/VSE EntireX RPC Servers*.
- 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-side Mapping Files in the RPC Server 42
- Deploying Server-side Mapping Files to the RPC Server 43
- Undeploying Server-side Mapping Files from the RPC Server 44
- Change Management of Server-side Mapping Files 44
- List Deployed Server-side Mapping Files 44
- Check if a Server-side Mapping File Revision has been Deployed 45
- Access Control: Secure Server Mapping File Deployment 45
- Ensure that Deployed Server-side Mapping Files are not Overwritten 45
- Is There a Way to Smoothly Introduce Server-side Mapping Files? 46

Server mapping enables the RPC server to correctly support special COBOL syntax such as `REDEFINES`, `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/VSE`, 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 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

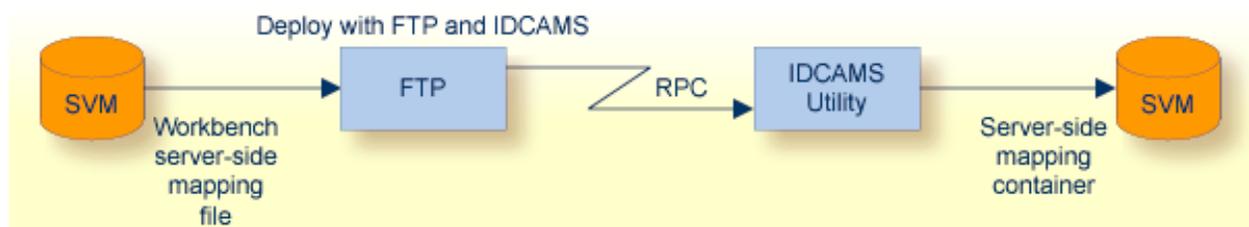
The following approaches are available to deploy a server-side mapping file (EntireX Workbench file with extension `.svm`; see *Server Mapping Files for COBOL*):

- 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 *Step 1: Define a Server-side Mapping Container - VSAMDEF.J (Optional)* under *Installing the z/VSE EntireX RPC Servers*.
- 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.

 **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/VSE, 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*.

```

* $$ JOB JNM=VSAMPRNT,CLASS=0,DISP=D
* $$ LST CLASS=A,DISP=K
/* ----- */
/* PRINT CONTENT OF AN SVM VSAM CLUSTER */
/* ----- */
// JOB VSAMPRNT
// DLBL ERXSVM,'ENTIREX.SVMDEV.KSDS',0,VSAM,CAT=VSESPUC
// EXEC IDCAMS,SIZE=AUTO
// PRINT INFILE(ERXSVM) CHAR
/*
/ &
* $$ EOJ

```

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 `YYYYMMDDHHIISS`. 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 48
- Aborting RPC Server Customer Code and Returning Error to RPC Client 49
- Automatic Syncpoint Handling 49

COBOL Scenarios

- [Scenario I: Calling an Existing COBOL Server](#)
- [Scenario II: Writing a New COBOL Server](#)

Scenario I: Calling an Existing COBOL Server

➤ To call an existing COBOL server

- 1 Use the *IDL Extractor for COBOL* to extract the Software AG IDL and, depending on the complexity, also a server mapping file. See *When is a Server Mapping File Required?* in the EntireX Workbench documentation.
- 2 Build an EntireX RPC client using any EntireX wrapper. 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/VSE 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/VSE CICS* in the COBOL Wrapper documentation for COBOL RPC Server examples.

Aborting RPC Server Customer Code and Returning Error to RPC Client

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.

Automatic Syncpoint Handling

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

- After a successful non-conversational request or an end-of-conversation, the server issues a `SYNCPPOINT COMMIT` command. If you are running under CICS with impersonation, this `SYNCPPOINT` 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 `SYNCPPOINT ROLLBACK` command to back out any pending database modifications.

