webMethods EntireX

EntireX z/OS CICS® RPC Server

Version 9.5 SP1

November 2013

This document applies to webMethods EntireX Version 9.5 SP1.

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

Copyright © 1997-2013 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://documentation.softwareag.com/legal/.

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://documentation.softwareag.com/legal/ 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 and license terms, please refer to "License Texts, Copyright Notices and Disclaimers of Third-Party Products". This document is part of the product documentation, located at http://documentation.softwareag.com/legal/ and/or in the root installation directory of the licensed product(s).

Document ID: EXX-CICSRPC-95SP1-20140628

Table of Contents

EntireX z/OS CICS® RPC Server v

1 Introduction to CICS RPC Server 1

- Inside the RPC Server 2
- Impersonation 9
- Usage of SVM Files 10
- Interface Types Supported by the RPC Server 11
- Automatic Syncpoint Handling 11
- Scenario I: Calling an Existing COBOL Server 11
- Scenario II: Writing a New COBOL Server 12
- Scenario III: Calling an Existing PL/I Server 12
- Scenario IV: Writing a New PL/I Server 13
- Returning Application Error Codes from a Server to a Client 13

2 Administering the CICS RPC Server 15

- Customizing the RPC Server 16
- Configuring the RPC Server 18
- Locating and Calling the Target Server 26
- Using SSL or TLS with the RPC Server 28
- RPC User Exits 28
- Autostart/Stop during CICS Start/Shutdown 33
- Multiple RPC Servers in the same CICS 34

3 RPC Online Maintenance Facility 35

- Monitoring the RPC Server 36
- Starting the RPC Server 38
- Pinging the RPC Server 39
- Stopping the RPC Server 39
- Modifying Parameters of the RPC Server 39
- Activating Tracing for the RPC Server 40
- Console Commands for the RPC Server 40

4 Deployment Service 43

- Introduction 44
- Scope 45
- Enabling the Deployment Service 45
- Disabling the Deployment Service 46

5 Handling SVM Files 47

- SVM Files in the EntireX Workbench 48
- SVM Files in the RPC Server 48
- Source Control of SVM Files 49
- Change Management of SVM Files 49
- Compare SVM Files 49
- List Deployed SVM Files 49
- Check if an SVM File Revision has been Deployed 50
- Access Control: Secure SVM File Deployment 50
- Ensure that Deployed SVM Files are not Overwritten 50

When is an SVM File Required? 51
Is There a Way to Smoothly Introduce SVM Files? 53

EntireX z/OS CICS® RPC Server

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

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

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

1 Introduction to CICS RPC Server

- Inside the RPC Server 2
- Impersonation 9
- Usage of SVM Files 10
- Interface Types Supported by the RPC Server 11
- Automatic Syncpoint Handling 11
- Scenario I: Calling an Existing COBOL Server 11
- Scenario II: Writing a New COBOL Server 12
- Scenario III: Calling an Existing PL/I Server 12
- Scenario IV: Writing a New PL/I Server 13
- Returning Application Error Codes from a Server to a Client 13

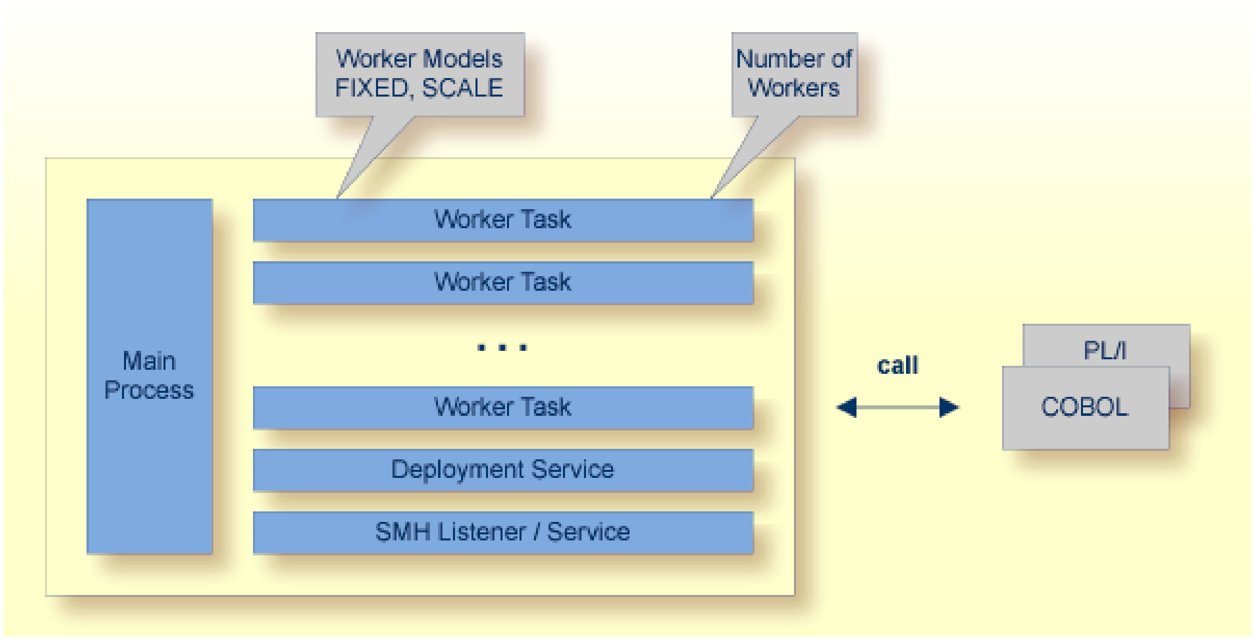
The EntireX z/OS CICS® RPC Server allows standard RPC clients to communicate with RPC servers on the operating system z/OS under CICS. It supports the programming languages COBOL and PL/I.

Inside the RPC Server

- Worker Models
- Inbuilt Services

- User Exits

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, worker tasks not used are 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
```

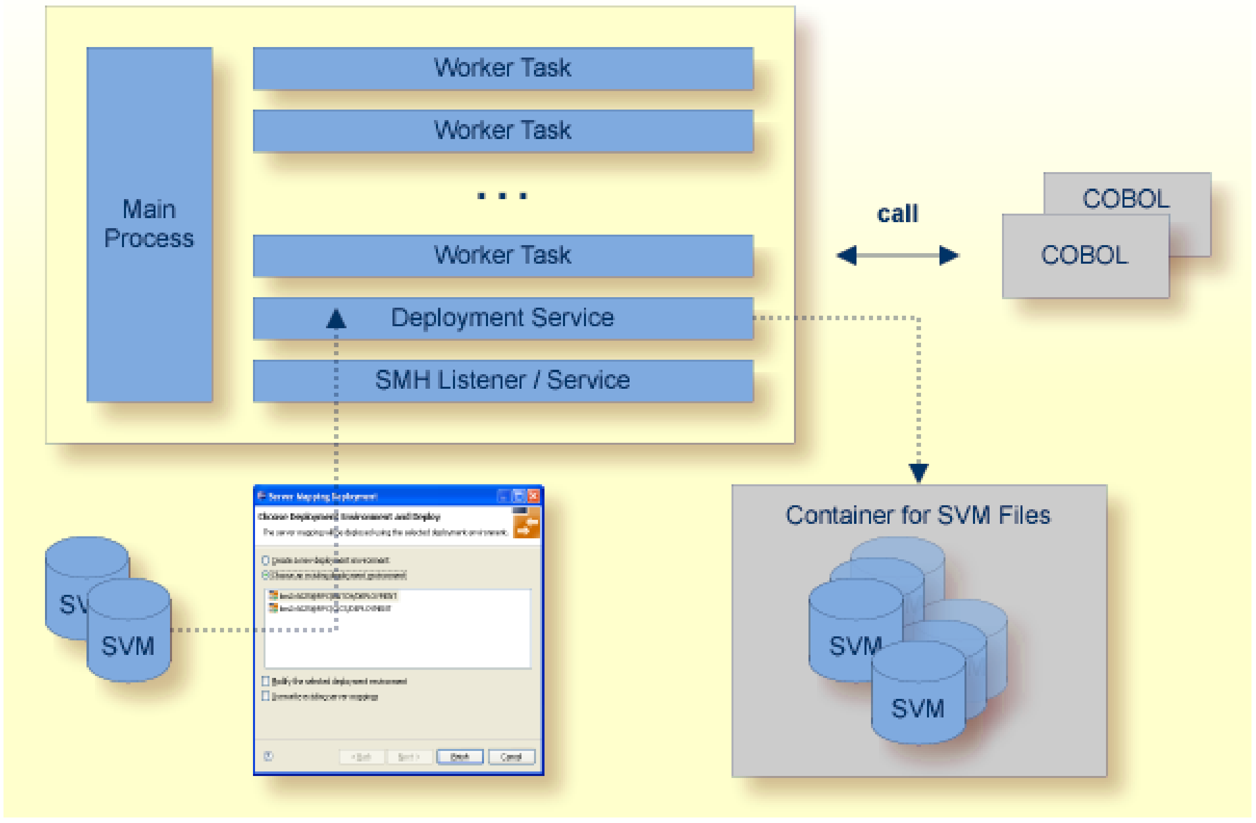
Inbuilt Services

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

- Deployment Service
- SMH Listener Service

Deployment Service

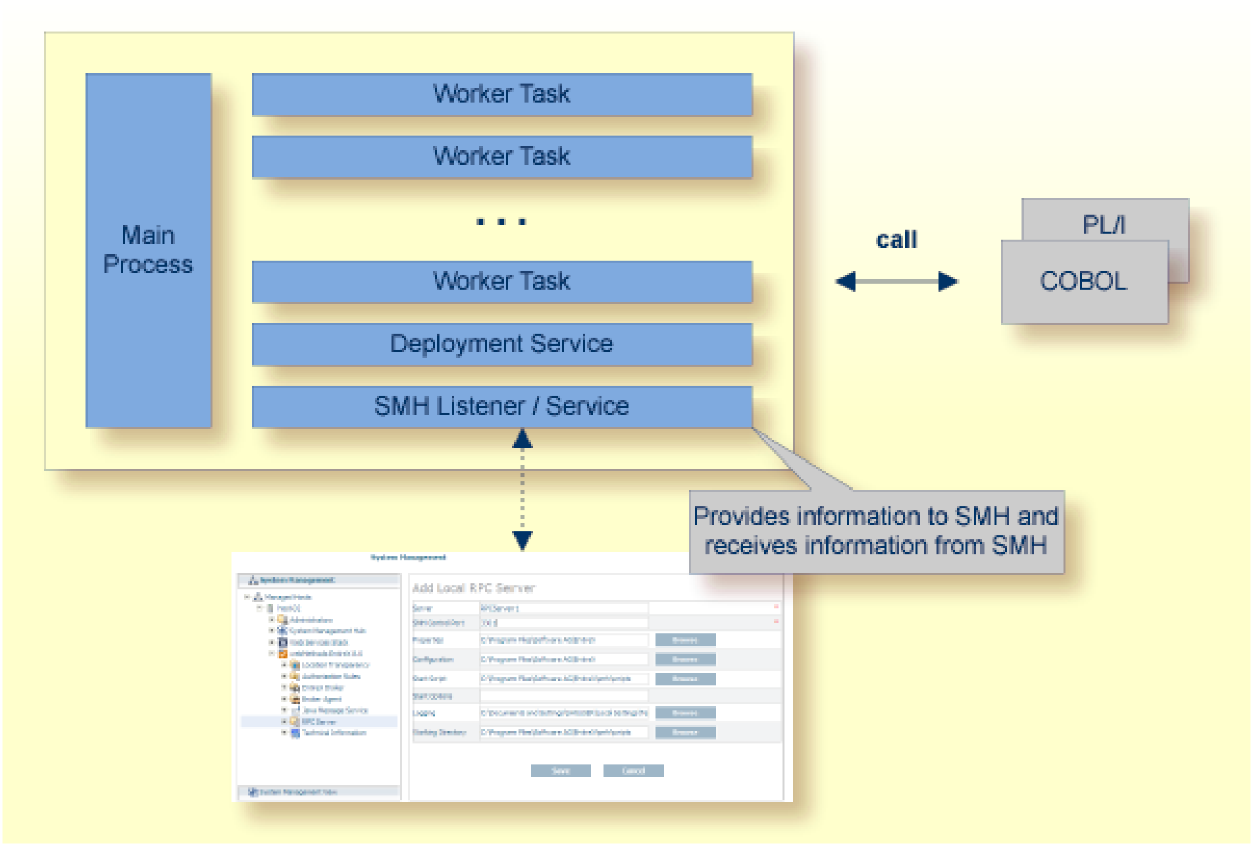
The Deployment Service allows you to deploy server mapping files (SVM files) interactively using the Deployment Wizard (see *Server Mapping Deployment*). On the RPC server side, the SVM files are stored in a VSAM file as the container. See ***Deployment Service*** for configuration information.



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 Exits

This section covers the following topics:

- User Exit COBUEX02
- User Exit RPCUEX01

User Exit COBUEX02

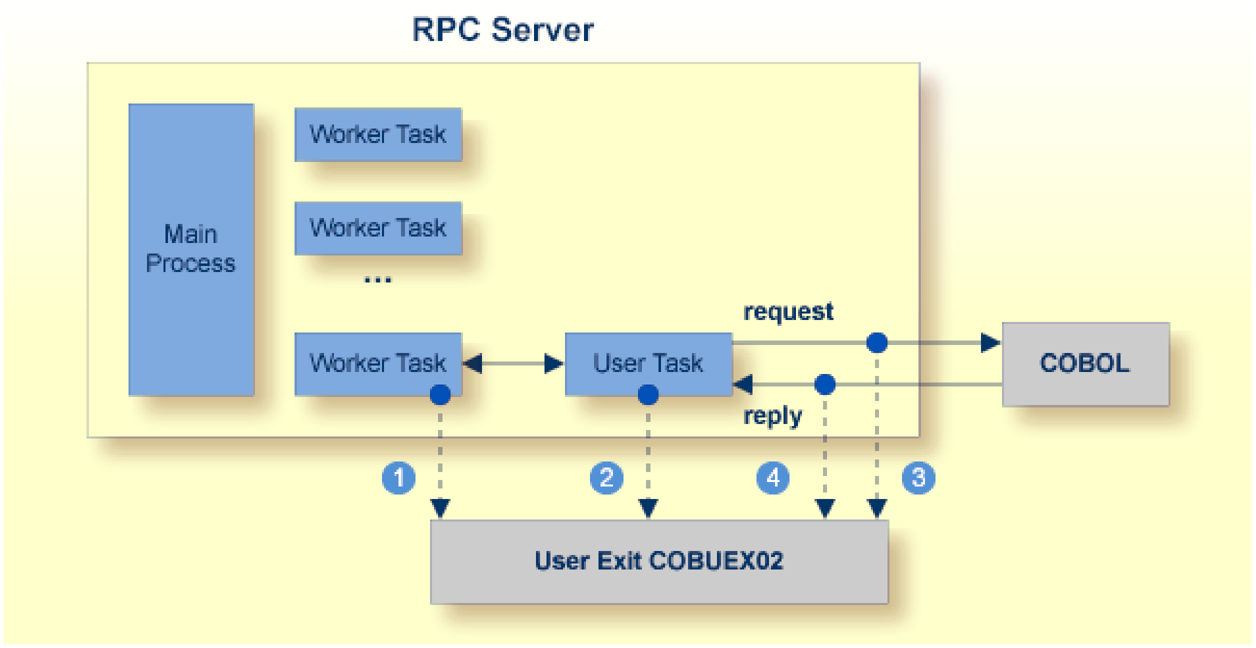
The CICS RPC Server provides a user exit to influence/control the RPC logic. The exit is called on the events `START-WORKER`, `START-USER`, `CALL-START` and `CALL-END`. The following tasks can be performed:

1 `START-WORKER` event before a CICS worker task is started. This allows you to programatically set the CICS transaction ID.

2 `START-USER` event. Apply CICS transaction ID and user ID to impersonated worker tasks. See *Impersonation*.

3 `CALL-START` event. Inspect, modify or terminate the RPC request (payload) from the RPC client.

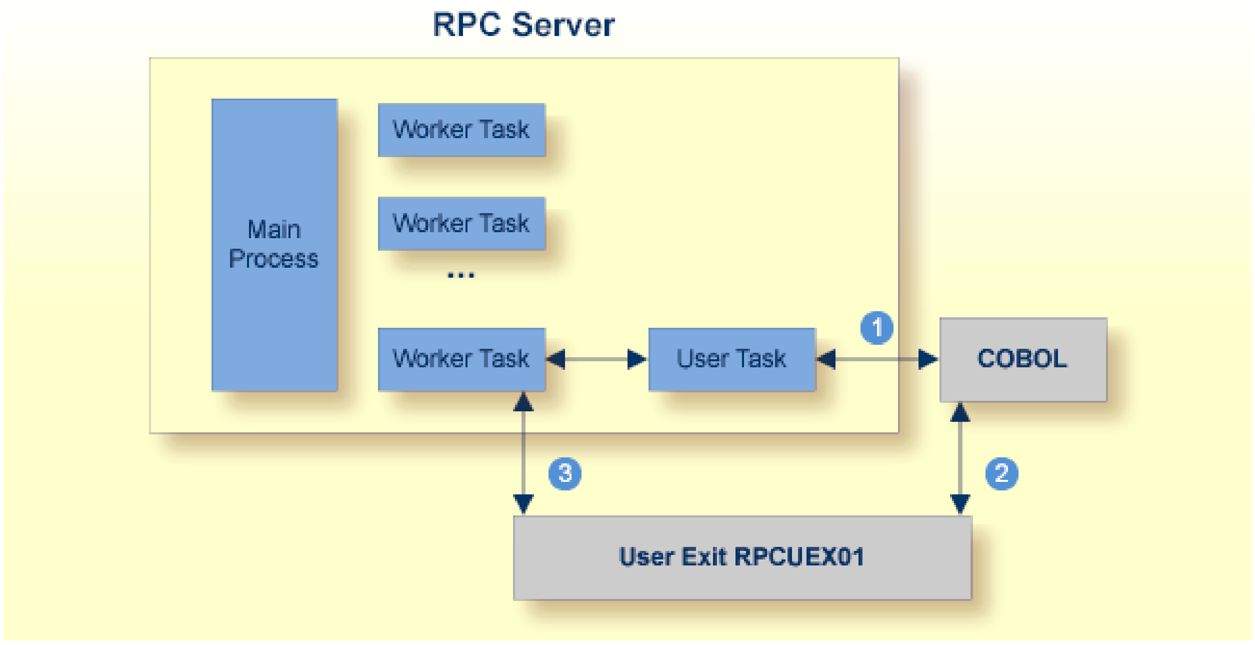
4 `CALL-END` event. Inspect or modify the RPC reply (payload) or give an error to the RPC client.



See also *Writing and Configuring User Exit COBUEX02*.

User Exit RPCUEX01

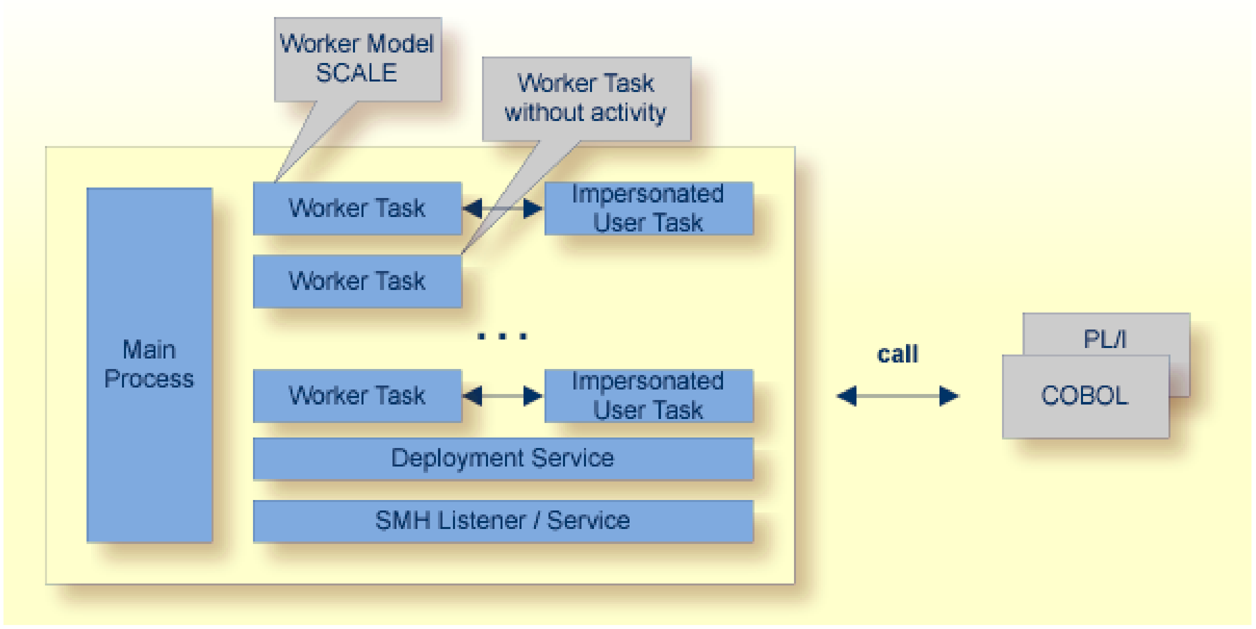
The server invokes the server program using `CICS LINK PROGRAM` and expects that the program returns with `CICS RETURN`. However, if the program uses `CICS ABEND CANCEL` to abort for particular error situations, the RPC server cannot trap the abort. If your server program uses `CICS ABEND CANCEL` you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



1. The server program is invoked within the user task.
2. The server program decides to abort using `CICS ABEND CANCEL` immediately before it calls the user exit. Then the server program can perform `CICS ABEND CANCEL` to abort. The `CICS ABEND CANCEL` terminates the user task.
3. The user exit 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 client.

See also *Using User Exit RPCUEX01*.

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.

The picture above shows the configuration `impersonation=yes`.

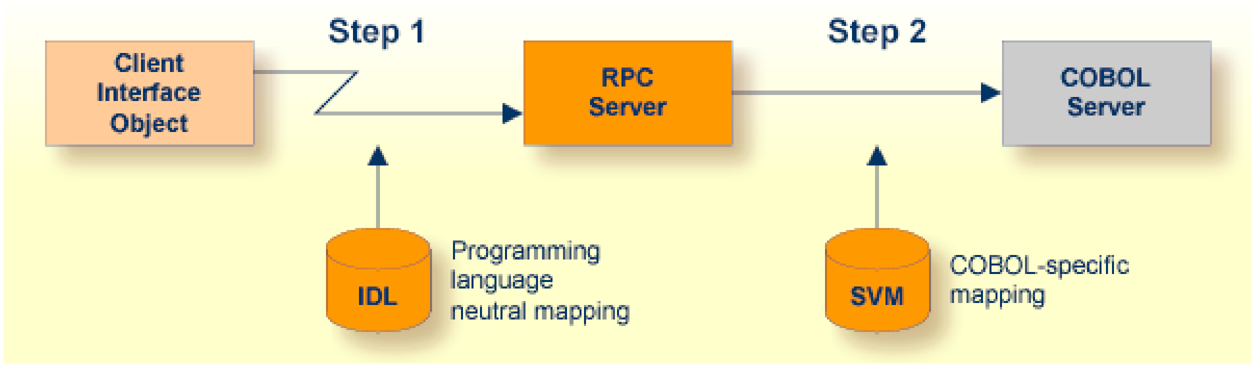
The lifetime of an impersonated user task starts when an open request for an RPC conversation or a non-conversational RPC request is received. It ends when the RPC conversation stops (after a commit operation or timeout) or when the non-conversational RPC request has been performed.

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 SVM Files

To correctly support special COBOL syntax such as `JUSTIFIED`, `SYNCHRONIZE` and `OCCURS DEPENDING ON` clauses, `LEVEL-88` fields, etc., the CICS RPC Server requires in many situations a server mapping file.

SVM files contain COBOL-specific mapping information that is not included in the IDL file and therefore *not* sent by an EntireX RPC client to the RPC server. See also ***When is an SVM File Required?*** under ***Handling SVM Files***.



The RPC server marshalls 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 SVM file (Step 2). In this way the COBOL server can be called as expected.

The SVM files are retrieved as a result of the *IDL Extractor for COBOL* extraction process and the *COBOL Wrapper* if a COBOL server is generated.

You can customize the usage of the SVM file using the `ERXMAIN` **macro** parameter `SVM`. See ***Configuring the RPC Server***.

Note: SVM files are used for COBOL only.

Interface Types Supported by the RPC Server

Supported interface types 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)

Automatic Syncpoint Handling

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

- After a successful non-conversational request or an end-of-conversation, the server issues a `SYNCPOINT COMMIT` command. If you are running under CICS with impersonation, this `SYNCPOINT` command is not executed by the server, but by CICS when the user task is terminated. See ***Impersonation***.
- After abnormal termination of a non-conversational request or a conversation due to an error, the server performs a `SYNCPOINT ROLLBACK` command to back out any pending database modifications.

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 of the extraction, also an SVM file.
2. Build an EntireX RPC client using any EntireX wrapper. See *EntireX Wrappers*. For a quick test you can:
   - use the IDL Tester; see *EntireX IDL Tester* in the EntireX Workbench documentation

- generate an XML mapping file (XMM) and use the XML Tester for verification; see *EntireX XML Tester*

See *Client and Server Examples for z/OS CICS* 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 of the extraction, also an SVM file. Write your COBOL server and proceed as described under *Using the COBOL Wrapper for the Server Side*.
2. Build an EntireX RPC client using any EntireX wrapper. See *EntireX Wrappers*. For a quick test you can:
   - use the IDL Tester; see *EntireX IDL Tester* in the EntireX Workbench documentation
   - generate an XML mapping file (XMM) and use the XML Tester for verification; see *EntireX XML Tester*

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

Scenario III: Calling an Existing PL/I Server

To call an existing PL/I server

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

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

Scenario IV: Writing a New PL/I Server

To write a new PL/I server

1 Use the *PL/I Wrapper* to generate a PL/I server skeleton. Write your PL/I server and proceed as described under *Using the PL/I Wrapper for the Server Side*.

2 Build an EntireX RPC client using any EntireX wrapper. See *EntireX Wrappers*. For a quick test you can:

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

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

Returning Application Error Codes from a Server to a Client

See *Returning Application Errors from a Server under z/OS CICS to a Client* under *Writing Applications with the COBOL Wrapper*.

2 Administering the CICS RPC Server

- Customizing the RPC Server 16
- Configuring the RPC Server 18
- Locating and Calling the Target Server 26
- Using SSL or TLS with the RPC Server 28
- RPC User Exits 28
- Autostart/Stop during CICS Start/Shutdown 33
- Multiple RPC Servers in the same CICS 34

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

Customizing the RPC Server

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

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

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 mapping files, see `SVM`
  - scalability parameters such as `endworker`, `minworker` and `maxworker`, see `ENDW`, `MINW` and `MAXW`
  - etc.

ERXMAIN Macro

- creates an *`ERXMAIN` **Control Block***, a persistent setup of the CICS RPC Server
- needs to be assembled to define a setup
- is defined in Assembler program `EMAINGEN` (in EXP951.SRCE) - use this for assembling; see *Build the ERXMAIN Control Block* under *Installing EntireX RPC Servers under CICS*

RPC Online Maintenance Facility

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

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

IBM LE Runtime Options

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

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

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

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

Configuring the RPC Server

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

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

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

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

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

ERXMAIN Macro Syntax	RPC Online Maintenance Facility Commands	Default	Values	Req/ Opt
`BKRN`	`brokerid`	`ETB001`	Broker ID used by the server. See *Using the Broker ID in Applications* in the RPC Programming documentation. Example: `BKRN=myhost.com:1971`	R
`CLZN`	`class`	`RPC`	Server class part of the server address used by the server. The server address must be defined as a service in the broker attribute file (see *Service-specific Attributes* (`DEFAULTS=SERVICE`) under *Broker Attributes* in the administration documentation). Case-sensitive, up to 32 characters. Corresponds to `CLASS` attribute of the broker attribute file. Example: `CLZN=MyRPC`	R
`SRVN`	`servername`	`SRV1`	Server name part of the server address used by the server. The server address must be defined as a service in the broker attribute file. See *Service-specific Attributes* (`DEFAULTS=SERVICE`) under *Broker Attributes* in the administration documentation. Case-sensitive, up to 32 characters. Corresponds to `SERVER` of the broker attribute file.	R

ERXMAIN Macro Syntax	RPC Online Maintenance Facility Commands	Default	Values	Req/ Opt
			Example: `SRVN=mySrv`	
`SVCN`	`service`	`CALLNAT`	Service part of the server address used by the server. The server address must be defined as a service in the broker attribute file. See *Service-specific Attributes* (`DEFAULTS=SERVICE`) under *Broker Attributes* in the administration documentation. Case-sensitive, up to 32 characters. Corresponds to `SERVICE` attribute of the broker attribute file. Example: `SVCN=MYSERVICE`	R
`CODE`	`codepage`	no codepage transferred	Depending on the internationalization approach, the codepage (locale string) where incoming data is provided to the COBOL server. Conversely, the COBOL server must provide outgoing data in the given codepage, otherwise unpredictable results occur. See *What is the Best Internationalization Approach to use?* under *Internationalization with EntireX* for information on which internationalization approach requires a codepage (locale string). By default, no codepage is transferred to the broker. For the most popular internationalization approach, *ICU Conversion* under *Introduction to Internationalization*, the correct codepage (locale string) must be provided. This means it must: ■ follow the rules described under *Locale String Mapping* in the internationalization documentation ■ be a codepage supported by the broker ■ be the codepage used in your environment for file and terminal IO, otherwise unpredictable results may occur. Example: `CODE=ibm-273`	O
`COMP`	`compresslevel`	`N`	Enforce compression when data is transferred between broker and server. See *Data Compression in EntireX Broker* in the general administration documentation. `compresslevel= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8| 9 | Y | N` `0-9` 0=no compression 9=max. compression	O

ERXMAIN Macro Syntax	RPC Online Maintenance Facility Commands	Default	Values	Req/ Opt
			`N` No compression. `Y` Compression level 6. Example: `COMP=6`	
`CYCL`	`restartcycles`	`15`	Number of restart attempts if the broker is not available. This can be used to keep the CICS RPC Server running while the broker is down for a short time. A restart cycle will be repeated at an interval which is calculated as follows: `timeout` + `ETB_TIMEOUT` + 60 seconds where `timeout` is the RPC server parameter (see this table), and *`ETB_TIMEOUT`* is the environment variable (see *Environment Variables in EntireX* in the general administration documentation) When the number of cycles is reached and a connection to the broker is not possible, the RPC server stops. Example: `CYCL=30`	O
`DPLY`	`deployment`	`NO`	Activates the deployment service, see ***Deployment Service***. Required to use the deployment wizard. See *Server Mapping Deployment Wizard* in the COBOL Wrapper documentation. `YES` Activates the deployment service. The RPC server registers the deployment service in the broker. `NO` The deployment service is deactivated. The RPC server does not register the deployment service in the broker. Example: `DPLY=YES`	O
`ENCR`	`encryptionlevel`	`0`	Enforce encryption when data is transferred between client and server. Requires EntireX Security. See `ENCRYPTION-LEVEL` under *Broker ACI Fields*. `0` Encryption is enforced.	O

ERXMAIN Macro Syntax	RPC Online Maintenance Facility Commands	Default	Values	Req/ Opt
			1 Encryption is enforced between server and broker kernel. 2 Encryption is enforced between server and broker kernel, and also between client and broker. Example: `ENCR=2`	
`ENDW`	`endworker`	`TIMEOUT`	`NEVER` Defines worker model `FIXED` with a fixed number of worker threads. The number of active workers is defined with `ERXMAIN` **macro** parameter `MINW`. `TIMEOUT` Defines slow-shrinking worker model `SCALE`, where the number of worker threads is adjusted to the current number of client requests. With value `TIMEOUT`, all worker threads not used are stopped in the time specified by the `ERXMAIN` **macro** parameter `TOUT`, except for the minimum number of active workers specified with `ERXMAIN` **macro** parameter `MINW`. The upper limit of workers parallel active is restricted with `ERXMAIN` **macro** parameter `MAXW`. `IMMEDIATE` Defines fast-shrinking worker model `SCALE`, where the number of worker threads is adjusted to the current number of client requests. With value `IMMEDIATE`, worker threads not used are stopped immediately as soon as they have finished their conversation, except for the minumum number of active workers defined with `ERXMAIN` **macro** parameter `MINW`. The upper limit of workers active in parallel is restricted with `ERXMAIN` **macro** parameter `MAXW`. This parameter is forced to value `TIMEOUT` if impersonation is switched on, see ***Impersonation*** and `ERXMAIN` **macro** parameter `IMPS`. Example: `ENDW=IMMEDIATE,MINW=2,MAXW=6`	O

ERXMAIN Macro Syntax	RPC Online Maintenance Facility Commands	Default	Values	Req/ Opt
`MINW`	`minworker`	`1`	Minimum number of workers active in parallel with worker model `SCALE` or number of workers in worker model `FIXED`. See also `ERXMAIN` **macro** parameter `ENDW`. Example: `MINW=2`	O
`MAXW`	`maxworker`	`10`	Upper limit of workers active in parallel with worker model `SCALE`. See also `ERXMAIN` **macro** parameter `ENDW`. Example: `MAXW=2`	O
`ETBL`	`etblnk`	`CICSETB`	Define the broker stub to be used. See *Administration of Broker Stubs under z/OS* in the z/OS administration documentation for available stubs. Example: `ETBL=CICSETB`	O
`EXIT`	n.a.		At startup, the CICS RPC Server will call the user exit to synchronize its version. If successful, the CICS RPC Server will continue and call the user exit for the implemented events. See ***User Exits***.	O
`IMPS`	`impersonation`	`NO`	Defines if RPC requests are executed under the user ID of the RPC client. Depending on settings, different levels of checks are done prior to RPC server execution. See also ***Impersonation***. `impersonation= NO | AUTO [, sameuser | , anyuser ]` `NO`: The RPC request is executed anonymously, which means the user ID of the RPC client is not used. RPC requests are executed under the user ID of the RPC server. `AUTO`: The RPC request runs impersonated under the supplied *RPC client user ID*. For execution of the RPC request, the CICS RPC Server starts a separate impersonated user task with the *RPC client user ID*. This means the user ID must be known to CICS, but no password check is performed. The worker model `SCALE` is forced; for details see ***Impersonation***. For this setting you have to take care the RPC client is correctly authenticated: use	O

ERXMAIN Macro Syntax	RPC Online Maintenance Facility Commands	Default	Values	Req/ Opt
			either a secure EntireX Broker (validation must be against the correct mainframe security repository where the user IDs are defined) and option `sameuser`, or use your own security implementation (option `anyuser` is supported for compatibility reasons if you need different broker and server user IDs - the customer-written security implementation must validate the RPC client using the *RPC client user ID*).	
			`sameuser`: The CICS RPC Server checks whether the *broker client user ID* matches the *RPC client user ID*. This is the default if `AUTO` is used.	
			`anyuser`: The *RPC client user ID* is used for authentication. The *broker client user ID* is ignored.	
			Note: 1. EntireX supports two user ID/password pairs: a *broker client user ID/password* pair and an (optional) *RPC user ID/password* pair sent from RPC clients to the RPC server. 2. With EntireX Security, the *broker client user ID/password* pair is checked. The *RPC user ID/password* pair is designed to be checked by the target RPC server. Thus it is possible to use different user IDs in the broker and target RPC server. 3. RPC clients send the (optional) *RPC user ID/password* pair in the same way as specifying the Natural user ID/password pair for a Natural RPC Server. See for example *Using Natural Security* for applications in C | COBOL | *PL/I* | *Web Services* | SOAP/XML | Java. 4. If the RPC client does not specify the optional *RPC user ID/password pair*, the *broker client user ID/password* pair is inherited to the *RPC user ID/password* pair and thus used for impersonation by the CICS RPC Server.	

ERXMAIN Macro Syntax	RPC Online Maintenance Facility Commands	Default	Values	Req/ Opt
			Example: `IMPS=auto`	
`LOGN`	`logon`	`YES`	Execute broker functions `LOGON`/`LOGOFF` in worker threads. Must match the setting of the broker attribute `AUTOLOGON`. Reliable RPC requires logon set to `YES`. See *Reliable RPC*. `NO` No logon/logoff functions are executed. `YES` Logon/logoff functions are executed. Example: `LOGN=no`	O
`n.a.`	`mapname`		Alias for command *`memory`*.	O
`n.a.`	`memory`		Command to load an ***`ERXMAIN` Control Block***. See ***Modifying Parameters of the RPC Server***.	O

`OPTS`	`runoption`	`0`	This parameter is for special purposes. It provides the CICS RPC Server with additional information. The runoptions are normally set to meet the platform's requirements. Set this parameter only if a support representive provides you with an option and asks you to do so.
			
			Syntax:
			`OPTS=(<option-list>)`
			`<option-list> = [<option-list>,] <option>`
			
			Example:
			`OPTS=(RUNOPT1,RUNOPT2)`
				O

`PSWD`	`password`		Password for broker logon. Case-sensitive, up to 32 characters. For more information see broker ACI control block field `PASSWORD`. Example: `PSWD=MyPwd`	O
`PRELOAD`	`preload`	`YES`	Enable to call CICS RPC Server with `AMODE=24` `YES` Enable to call RPC server with `AMODE 24` or `31`. Internally the CICS RPC Server preloads the called RPC server before execution to check the `AMODE` and releases the RPC server after this. The disadvantage of this approach is the CICS `USECOUNT` of the called RPC server program is increased by 2 for every executed RPC call.	O

ERXMAIN Macro Syntax	RPC Online Maintenance Facility Commands	Default	Values	Req/ Opt
			`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`.	
`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 Exits***) the CICS transaction ID for user tasks can be overridden. See also ***Inside the RPC Server***.	O
`SMH`	`smhport`	`0`	The port where the server listens for commands from the System Management Hub (SMH). If this port is 0 (default), no port is used and management by the SMH is disabled. See ***SMH Listener Service*** for more information. Example: `SMH=3001`	O

`SVM`	`svmfile`		Usage and location of SVM files. If no SVM parameter is given, the RPC server tries to open the SVM container using CICS file with name `ERXSVM`. If this CICS file is not available, no server mappings are used. For more information see ***Usage of SVM Files***.
			
			Syntax:
			`SVM=NO | cicsname`
			
			`cicsname` The RPC server tries to open the SVM container using the CICS file with name *`cicsname`*.
			
			`no` No server mappings are used.
			
			Example:
			`SVM=MYSVM`
			
			The server mapping file VSAM (container) must be installed and configured. See *Install the SVM File for a CICS RPC Server (Optional)* in the z/OS installation documentation.
				O

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

Locating and Calling the Target Server

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

- COBOL

- PL/I

COBOL

The approach used to derive the CICS program name for the RPC server depends on whether so-called server mapping files are used or not. See *Usage of SVM Files* for an introduction.

- If SVM files are used, the IDL library and IDL program names are used to form a key to locate the SVM entry in the SVM container. If an SVM entry is found, the CICS program name of the RPC server is derived from the SVM entry. 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 (see *The Software AG IDL Tree Pane*).
- If no SVM files are 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 *Interface Types Supported by the RPC Server*.
   - can be called with an `EXEC CICS LINK PROGRAM`
   - are accessible through the CICS RPL chain

2. Configure the `ERXMAIN` **macro** parameter `SVM` depending on whether SVM files are used or not.

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

PL/I

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

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

To use the CICS RPC Server with PL/I

- Make sure that all CICS programs called as RPC servers

  - use an interface type supported by the CICS RPC Server for target language PL/I; see *Interface Types Supported by the RPC Server*.

- can be called with an `EXEC CICS LINK PROGRAM`
- are accessible through the CICS RPL chain

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

Using SSL or TLS with the RPC Server

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

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

To set up SSL or TLS with AT-TLS

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

RPC User Exits

This section covers the following topics:

- Writing and Configuring User Exit COBUEX02

- Using User Exit RPCUEX01

Writing and Configuring User Exit COBUEX02

- Writing User Exit COBUEX02
- User Exit Events
- Configuring User Exit COBUEX02

Writing User Exit COBUEX02

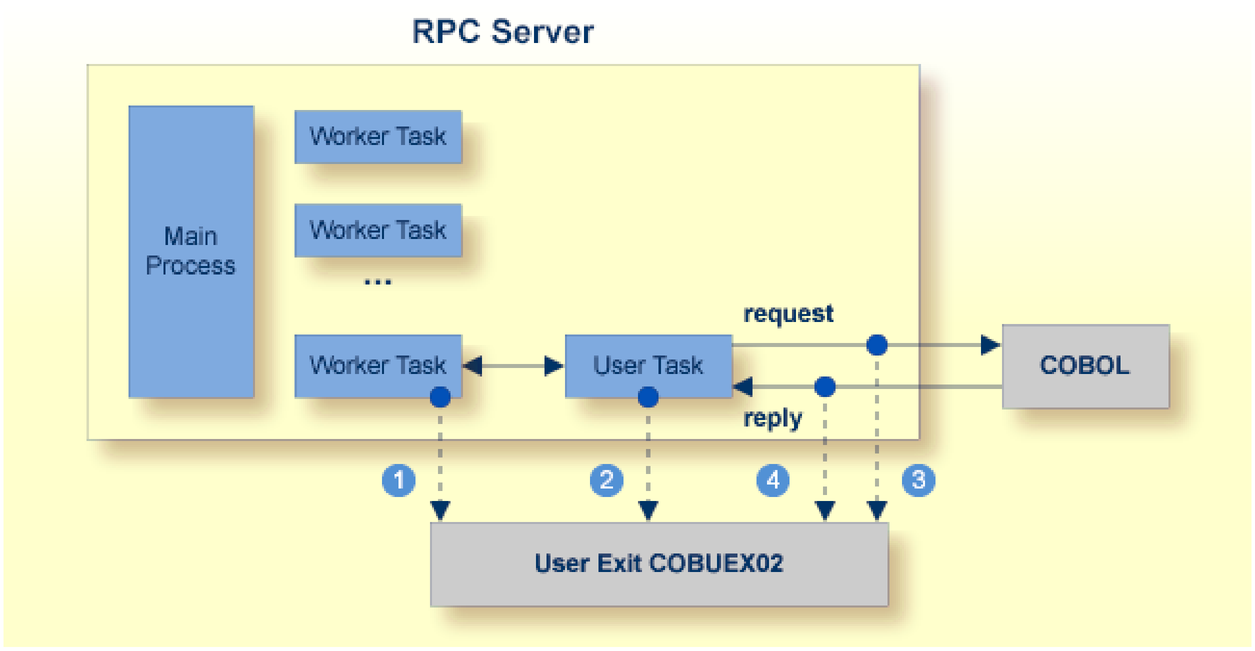
The Developer's Kit RPC source data set EXP951.SRCE of the EntireX z/OS installation provides the user exit skeleton `COBUEX02` for COBOL. Copy this skeleton so you have your own user exit source for modifications.

Accordingly, a COBOL copybook `COBUEX02` is provided in EXP951.INCL. Please add this library to your COBOL compiler `SYSLIB DD` 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 *Message Class 1022 - CICS RPC Server User Exit Messages* under *Error Messages and Codes*. 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-TERMID`	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 `TERMID` 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.

User Exit Events



1. `START-WORKER` event before a CICS worker task is started. This allows you to programatically 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.

Configuring User Exit COBUEX02

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.

Using User Exit RPCUEX01

The user exit `RPCUEX01` aborts an RPC out of a server program of the user application without returning to the RPC server. If your server program decides to abort with `CICS ABEND CANCEL`, it needs to invoke `RPCUEX01` directly before informing the RPC server about the imminent abort. If the server program aborts without calling `RPCUEX01`, the client gets a Broker timeout without any further information about the abort situation at the server.

This section covers the following topics:

- Installation
- Restrictions
- Process Flow
- Usage

Installation

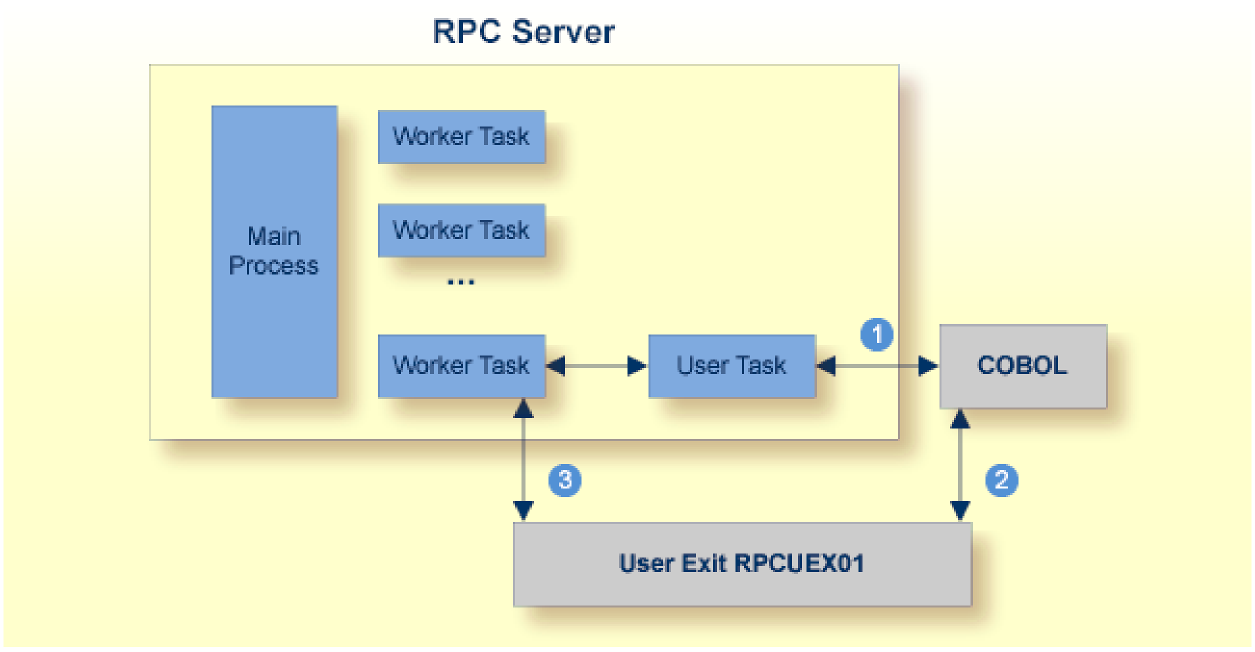
The program `RPCUEX01` must reside in the CICS load library concatenation. The following PPT entry is required:

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

Restrictions

This user exit is only for CICS and if impersonation is used. It is used only if your server program aborts its execution with `CICS ABEND CANCEL`.

Process Flow



1 The server program is invoked within the user task.

2 The server program decides to abort using `CICS ABEND CANCEL` immediately before it calls the user exit. Then the server program can perform `CICS ABEND CANCEL` to abort. The `CICS ABEND CANCEL` terminates the user task.

3 The user exit 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 client.

Usage

The server program calls the exit 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
```

Autostart/Stop during CICS Start/Shutdown

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

To start the CICS RPC Server during the initialization of CICS

1. If the COBOL source `ERXSTART` of the EntireX installation library EXP951.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)* in the z/OS installation documentation under *Installing EntireX RPC Servers under CICS*.

To stop the CICS RPC Server during the shutdown of CICS

1. If the COBOL source `ERXSTOP` of the EntireX installation library EXP951.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)* in the z/OS installation documentation under *Installing EntireX RPC Servers under CICS*.

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 under *Installing EntireX RPC Servers under CICS*.

3 RPC Online Maintenance Facility

- Monitoring the RPC Server 36
- Starting the RPC Server 38
- Pinging the RPC Server 39
- Stopping the RPC Server 39
- Modifying Parameters of the RPC Server 39
- Activating Tracing for the RPC Server 40
- Console Commands for the RPC Server 40

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 Control 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*.

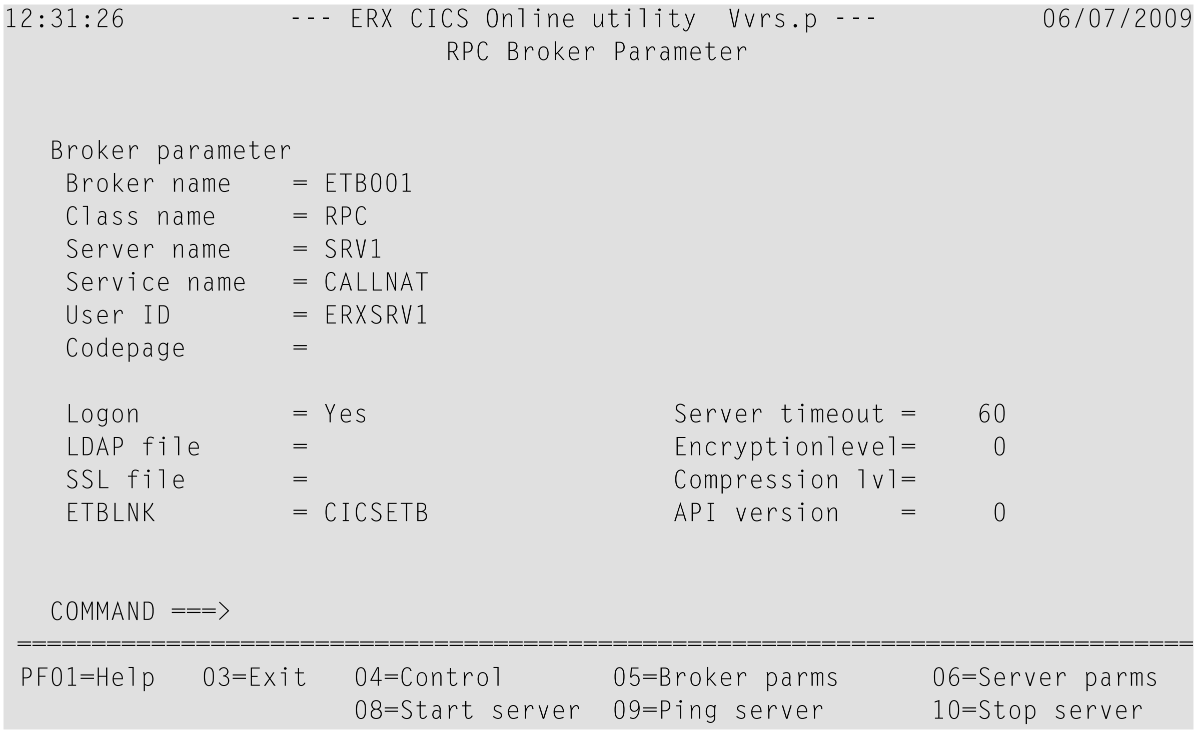
  The control parameter map is displayed:

  ```
  12:29:28          --- ERX CICS Online utility  Vvrs.p ---          06/07/2012
                                RPC Server Control
     MAIN Task
      Status          Running
     WORKER Tasks
         Registered               1
         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
  ```

  Press **PF04** from any map to reenter the control parameter map.

To display the Broker parameters

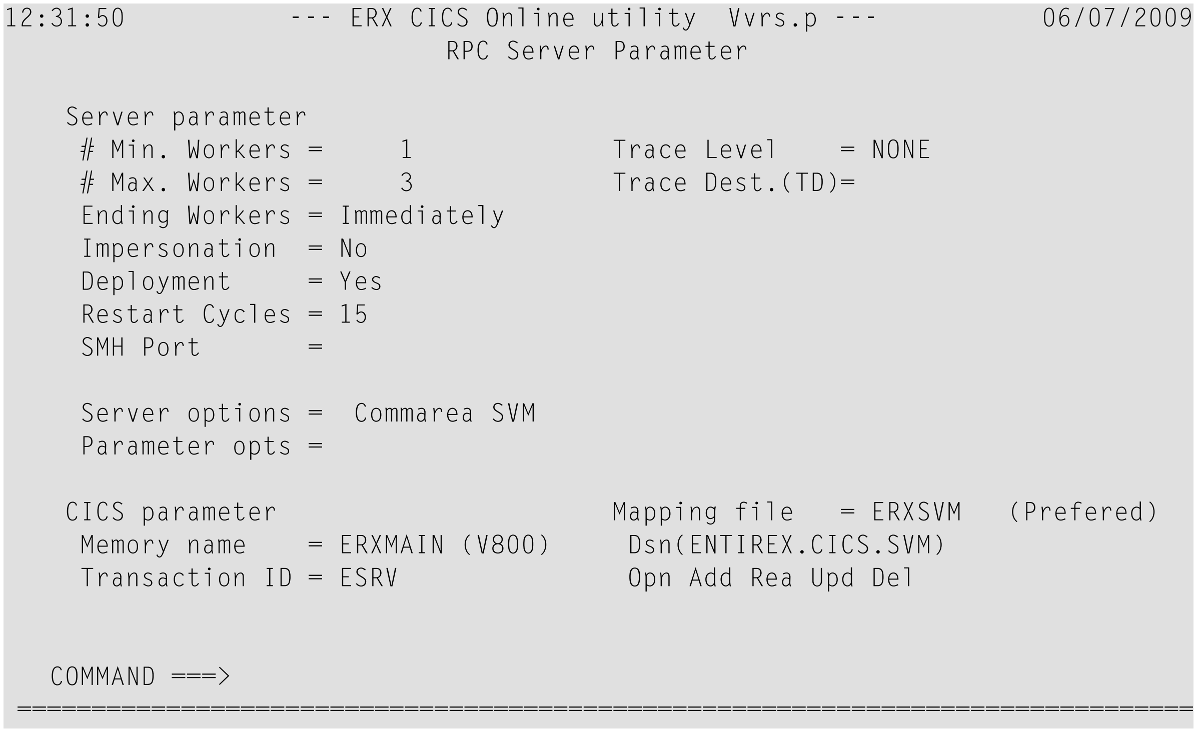
- Press **PF05** from any map and the broker parameters will be displayed:



```
12:31:26          --- ERX CICS Online utility  Vvrs.p ---          06/07/2009
                             RPC Broker Parameter


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

    Logon          = Yes                      Server timeout =    60
    LDAP file      =                          Encryptionlevel=     0
    SSL file       =                          Compression lvl=
    ETBLNK         = CICSETB                  API version    =     0


   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 Server parameters,

- Press **PF06** from any map and the server parameters will be displayed:



```
12:31:50          --- ERX CICS Online utility  Vvrs.p ---          06/07/2009
                             RPC Server Parameter

    Server parameter
     # Min. Workers =     1             Trace Level    = NONE
     # Max. Workers =     3             Trace Dest.(TD)=
     Ending Workers = Immediately
     Impersonation  = No
     Deployment     = Yes
     Restart Cycles = 15
     SMH Port       =

     Server options =  Commarea SVM
     Parameter opts =

    CICS parameter                      Mapping file   = ERXSVM   (Prefered)
     Memory name    = ERXMAIN (V800)     Dsn(ENTIREX.CICS.SVM)
     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 help for the RPC Online Maintenance Facility

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

To stop the RPC Online Maintenance Facility

- Enter `Exit` or use **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**.

Alternative Methods

- Automatically during CICS startup. See ***Autostart/Stop during CICS Start/Shutdown***.
- With 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.

Alternative Methods

- Automatically during CICS shutdown. See *Autostart/Stop during CICS Start/Shutdown*.
- With 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 EntireX 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 EntireX 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 `TRLV` under ***Configuring the RPC Server***.

  Example: `tracelevel=Standard`

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

Console Commands for the RPC Server

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

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

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

- Use the following z/OS modify command:

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

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

- Use the following z/OS modify command:

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

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

- Use the following z/OS modify command:

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

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

- Use the following z/OS modify command:

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

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

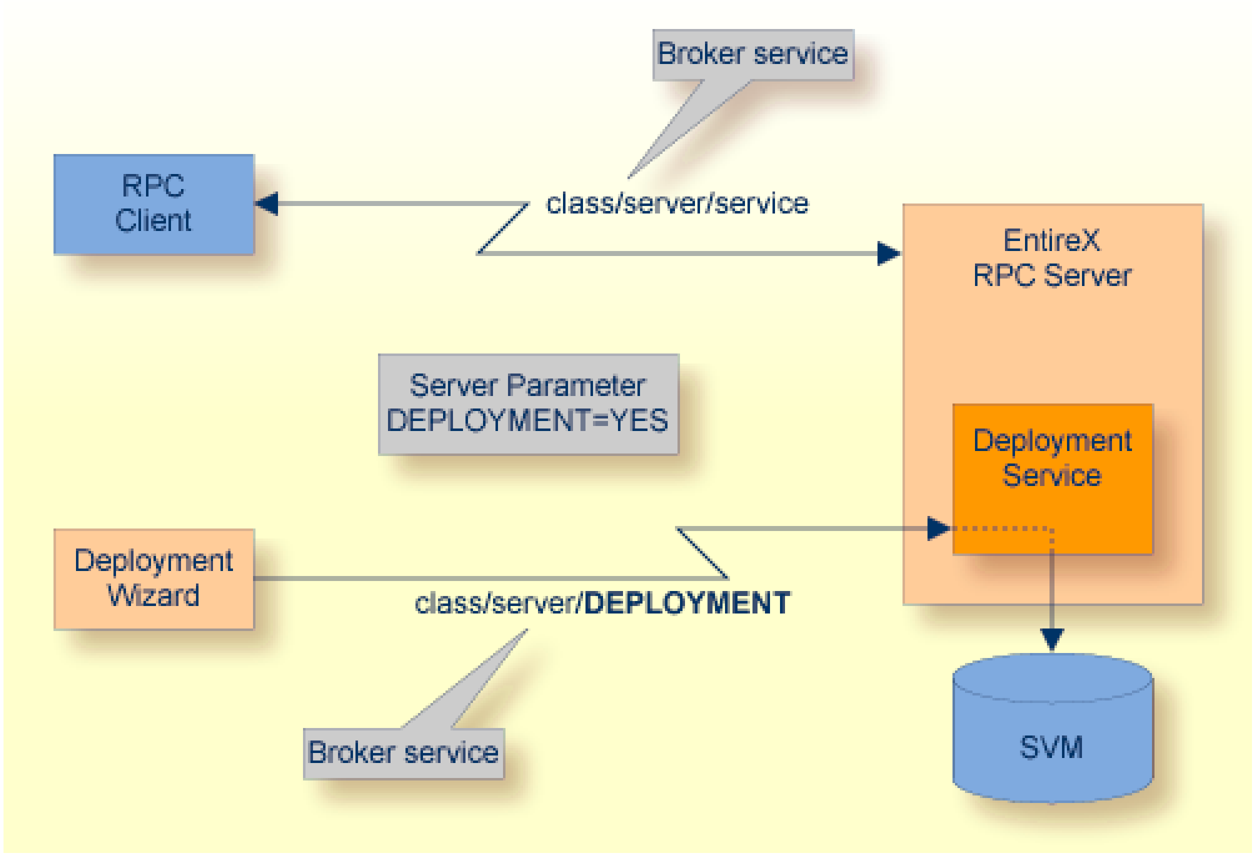
4 Deployment Service

- Introduction 44
- Scope 45
- Enabling the Deployment Service 45
- Disabling the Deployment Service 46

Introduction

The deployment service

- is the (server-side) counterpart to the deployment wizard; see *Server Mapping Deployment Wizard*.
- 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.



Scope

The deployment service is used for the

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

See *Server Mapping Deployment Wizard*.

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 mapping file VSAM (container) must be installed and configured. See *Install the SVM File for a CICS RPC Server (Optional)* in the z/OS installation documentation.
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 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 Handling SVM Files

- SVM Files in the EntireX Workbench 48
- SVM Files in the RPC Server 48
- Source Control of SVM Files 49
- Change Management of SVM Files 49
- Compare SVM Files 49
- List Deployed SVM Files 49
- Check if an SVM File Revision has been Deployed 50
- Access Control: Secure SVM File Deployment 50
- Ensure that Deployed SVM Files are not Overwritten 50
- When is an SVM File Required? 51
- Is There a Way to Smoothly Introduce SVM Files? 53

A server mapping file (SVM) enables the RPC server to correctly support special COBOL syntax such as `REDEFINE`s, `JUSTIFIED`, `SYNCHRONIZE` and `OCCURS DEPENDING ON` clauses, `LEVEL-88` fields, etc. If one of these elements is used, the EntireX Workbench automatically extracts an SVM file in addition to the IDL (interface definition language), or an SVM file is generated by the COBOL Wrapper for a server skeleton. The SVM file is used at runtime to marshal and unmarshal the RPC data stream.

SVM Files in the EntireX Workbench

In the *EntireX Workbench*, an SVM file has to relate to an appropriate IDL file. Therefore, you always have to keep the IDL file and the SVM file together in the same folder.

If there is an SVM file and a corresponding IDL file,

- at least one of the IDL programs in the corresponding IDL file requires server-mapping information to correctly call the target server. For those IDL programs, there is an SVM entry (line) in the Workbench SVM file.
- deployment of the SVM file to the RPC server is mandatory, see *Server Mapping Deployment*.

If there is an IDL file but no corresponding SVM file,

- there is no IDL program that requires server mapping information.

SVM Files in the RPC Server

Under z/OS, SVM entries of EntireX Workbench SVM files are stored as records within one VSAM file (containing all SVM entries from all Workbench SVM files). 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). The CICS, Batch and IMS RPC servers use a VSAM file as the container.

If *one* server requires an SVM file, you need to provide this to the RPC server:

- Development environments: to allow the deployment of new SVM files, enable the deployment service. See ***Enabling the Deployment Service***.
- Production environments: provide SVM files to the RPC server. See `ERXMAIN` macro parameter `SVM`.

If *no* server requires an SVM file, you can execute the RPC server without SVM files:

- Development environments: you can disable the deployment service. See ***Disabling the Deployment Service***.

- Production environments: there is no need to provide SVM files to the RPC server. See `ERXMAIN` macro parameter `SVM`.

Source Control of SVM Files

Because SVM entries within an SVM file contain text data only, a Workbench SVM file is text-based (although it is not intended for human consumption). Therefore, you can include it in your source control management together with the IDL file and the COBOL source(s) as a triplet that should always be kept in sync.

Change Management of SVM Files

For z/OS, change management for a VSAM file (SVM container) 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 Workbench SVM files added since the last backup should be available.

Compare SVM Files

For SVM files in the *EntireX Workbench* format, you can use a third party file/text compare tool to check if two files are identical.

The SVM entries (corresponding to lines in a Workbench SVM file) contain a creation timestamp at offset 276 (decimal) in the format *`YYYYMMDDHHIISST`*. The precision is 1/10 of a second.

List Deployed SVM Files

Use IDCAMS:

```
//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 SVM records of the VSAM file.

Check if an SVM File Revision has been Deployed

SVM entries (corresponding to lines in Workbench SVM files) 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 VSAM file (SVM container).

Access Control: Secure SVM File Deployment

For deployment with the *Server Mapping Deployment Wizard*, use EntireX Security if the broker is running on a platform other than BS2000/OSD. See ***Enabling the Deployment Service***.

For IBM z/OS deployment tools (for instance IDCAMS), use RACF to secure deployment.

Ensure that Deployed SVM Files are not Overwritten

For IDCAMS, use the `NOREPLACE` option to disallow overwriting of duplicate SVM records in the VSAM file (container). See *Server Mapping Deployment to z/OS, using FTP and IDCAMS*.

When is an SVM File Required?

For the IDL Extractor for COBOL

Interface Type	COBOL Syntax	COBOL Mapping Editor	SVM Required	More Information
CICS with DFHCOMMAREA Calling Convention and `IN` different to `OUT`	all		yes	*CICS with `DFHCOMMAREA` Calling Convention* under *Introduction to the IDL Extractor for COBOL* | *CICS `DFHCOMMAREA`* under *COBOL Parameter Selection*
CICS Channel Container Calling Convention	all		yes	*CICS with Channel Container Calling Convention*
CICS with DFHCOMMAREA Large Buffer Interface	all		yes	*CICS with `DFHCOMMAREA` Large Buffer Interface*
IMS MPP Message Interface (IMS Connect)	all		yes	*IMS MPP Message Interface (IMS Connect)*
IMS BMP with Standard Linkage Calling Convention	all		yes	*IMS BMP with Standard Linkage Calling Convention*
Micro Focus with Standard Linkage Calling Convention	`BINARY` clause		yes	*Micro Focus with Standard Linkage Calling Convention*
all	`OCCURS DEPENDING ON` clause		yes	*Tables with Variable Size - `DEPENDING ON` Clause* under *COBOL to IDL Mapping* in the IDL Extractor for COBOL documentation
all	`REDEFINES` clause		yes	*`REDEFINE` Clause*
all	`TRAILING [SEPARATE]` clause		yes	*`SIGN LEADING` and `TRAILING SEPARATE` Clause*
all	`LEADING [SEPARATE]` clause		yes	*`SIGN LEADING` and `TRAILING SEPARATE` Clause*
all	`ALIGNED RIGHT` attribute		yes	
all	all	Rename of program	yes	*The Software AG IDL Tree Pane* under *Mapping Editor User Interface* in the IDL Extractor for COBOL documentation

Interface Type	COBOL Syntax	COBOL Mapping Editor	SVM Required	More Information
all	all	Map to operation	yes	*Context Menu* under *The COBOL Parameters Pane*
all	all	Map to constant	yes	*Context Menu*
all	all	Suppress	yes	*Context Menu*
other combinations			no	

For the COBOL Wrapper

This depends on the interface type chosen and the IDL type:

Interface Type	IDL Type	COBOL Wrapper	SVM Required	More Information
CICS with DFHCOMMAREA Large Buffer Interface	all		yes	*CICS with DFHCOMMAREA Large Buffer Interface* under *COBOL Server Interface Types*
CICS with Channel Container Calling Convention	all		yes	*CICS with Channel Container Calling Convention*
IMS BMP with Standard Linkage Calling Convention	all		yes	*IMS BMP with Standard Linkage Calling Convention*
Micro Focus	I2 or I4		yes	*Micro Focus with Standard Linkage Calling Convention* | *IDL Data Types* under *Software AG IDL File* in the IDL Editor documentation
all	IDL unbounded array		yes	`array-definition` under *Software AG IDL Grammar* in the *IDL Editor* documentation
all	IDL unbounded group		yes	`group-parameter-definition` under *Software AG IDL Grammar* in the *IDL Editor* documentation
all	all	IDL program name is not a valid COBOL name and is therefore adapted, or the COBOL program name is customized	yes	*Customize Automatically Generated Server Names*
other combinations			no	

Is There a Way to Smoothly Introduce SVM Files?

All EntireX RPC servers can be executed without SVM files. There is no need to install the SVM container (see ***SVM Files in the RPC Server***) as long as you do not use features that require SVM files (see ***When is an SVM File Required?***). You can also call COBOL servers generated or extracted with previous versions of EntireX mixed with a COBOL server that requires SVM files. All EntireX RPC servers are backward compatible.