webMethods EntireX

EntireX RPC Server for IMS

Version 10.9

April 2023

WEBMETHODS

This document applies to webMethods EntireX Version 10.9 and all subsequent releases.

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

Document ID: EXX-IMSRPC-109-20230403

Table of Contents

1 About this Documentation

Document Conventions

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

Online Information and Support

Product Documentation

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

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

Product Training

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

Tech Community

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

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

Product Support

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

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

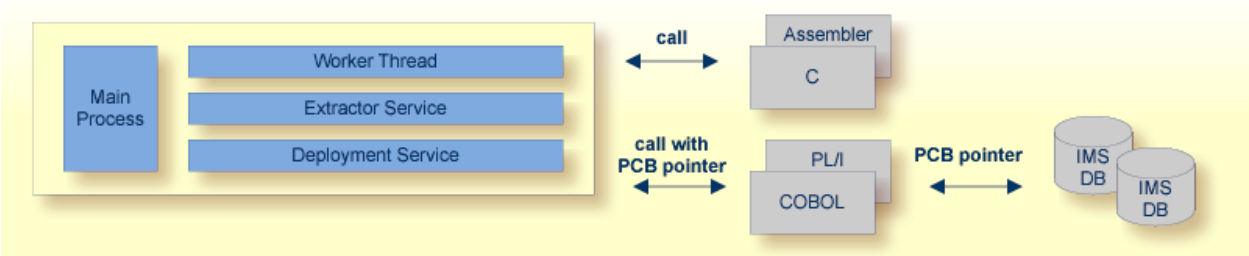
Data Protection

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

2 Introduction to the RPC Server for IMS

The EntireX RPC Server for IMS allows standard RPC clients to communicate with RPC servers on the operating system z/OS running with IMS in BMP mode. It supports the programming languages COBOL, PL/I and C and can provide IMS-specific PCB pointers for access to IMS databases if needed.

IMS-specific PCB Pointers



The RPC Server for IMS provides one worker thread. RPC requests are worked off inside the RPC server in the worker thread, which is controlled by a main thread. IMS-specific PCB pointers can be provided as parameters in the linkage section for COBOL and PL/I. They allow you to access the IMS PCB pointer `IOPCB`, for example to print data or to start an asynchronous transaction and to access IMS databases

IMS-specific PCB pointers are supported with the following programming languages:

- **COBOL**
  - If the COBOL Wrapper is used, see *IMS PSB List*.
  - If the IDL Extractor for COBOL is used, see *IMS BMP with Standard Linkage Calling Convention*.

  For COBOL, the mapping to IMS-specific PCB pointers is done with server mapping files, thus a server-mapping file is always required. See ***Usage of Server Mapping Files***.
- **PL/I**
  - If the PL/I Wrapper is used, see *PSB List*.
  - If the IDL Extractor for PL/I is used, see *Preferences*.

  For PL/I, the mapping to IMS-specific PCB pointers is done with server interface object(s). They are generated using the PL/I Wrapper (see ***Scenario III: Calling an Existing PL/I Server***) and provided with the parameter `stublib` to the RPC Server for IMS.

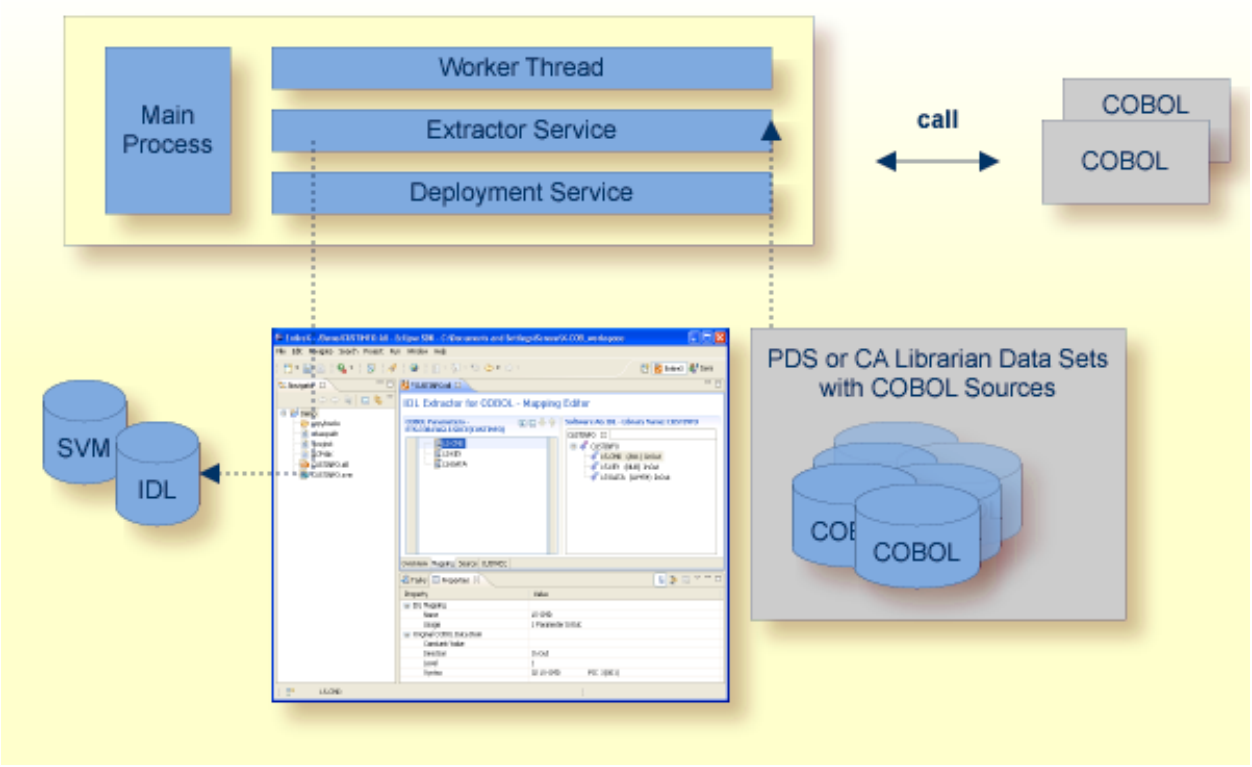
Inbuilt Services

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

- Extractor Service
- Deployment Service

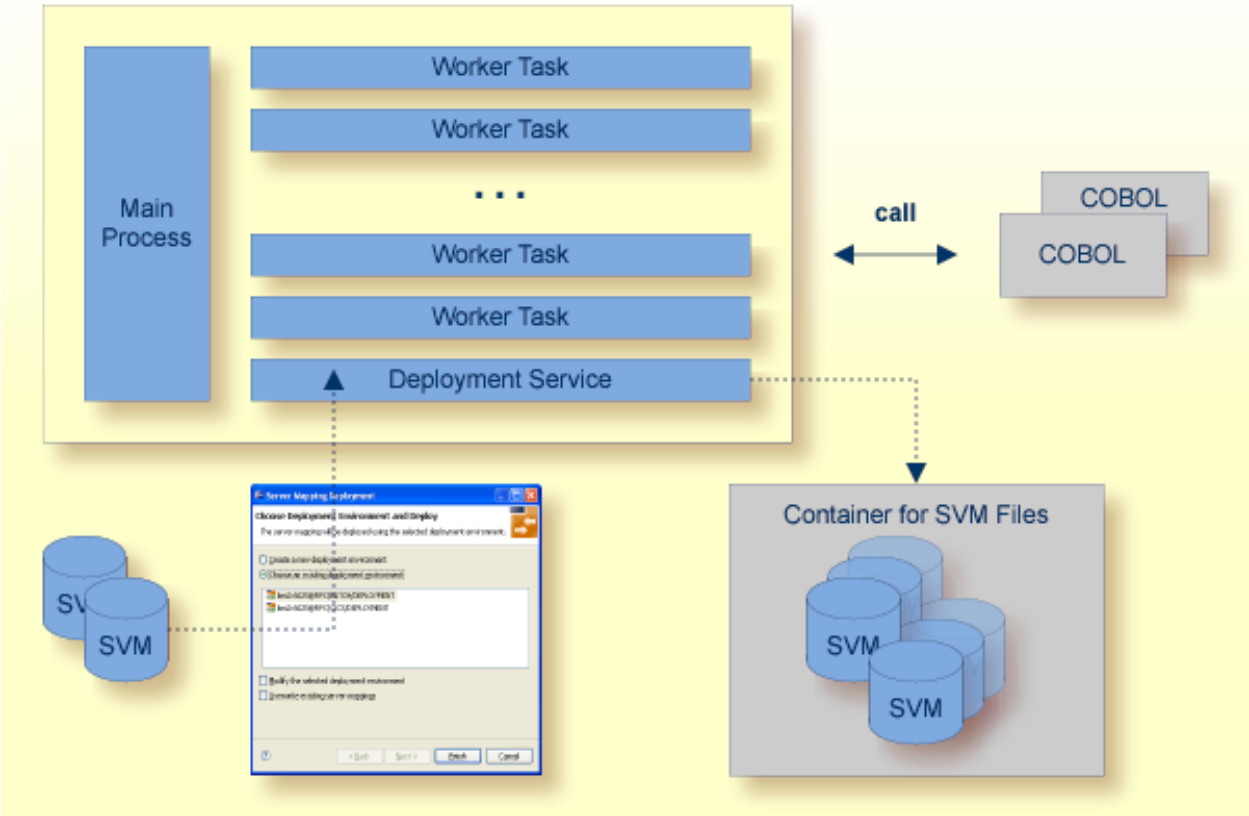
Extractor Service

The Extractor Service is a prerequisite for remote extractions with the IDL Extractor for COBOL and IDL Extractor for PL/I. See *Extractor Service* for more information.

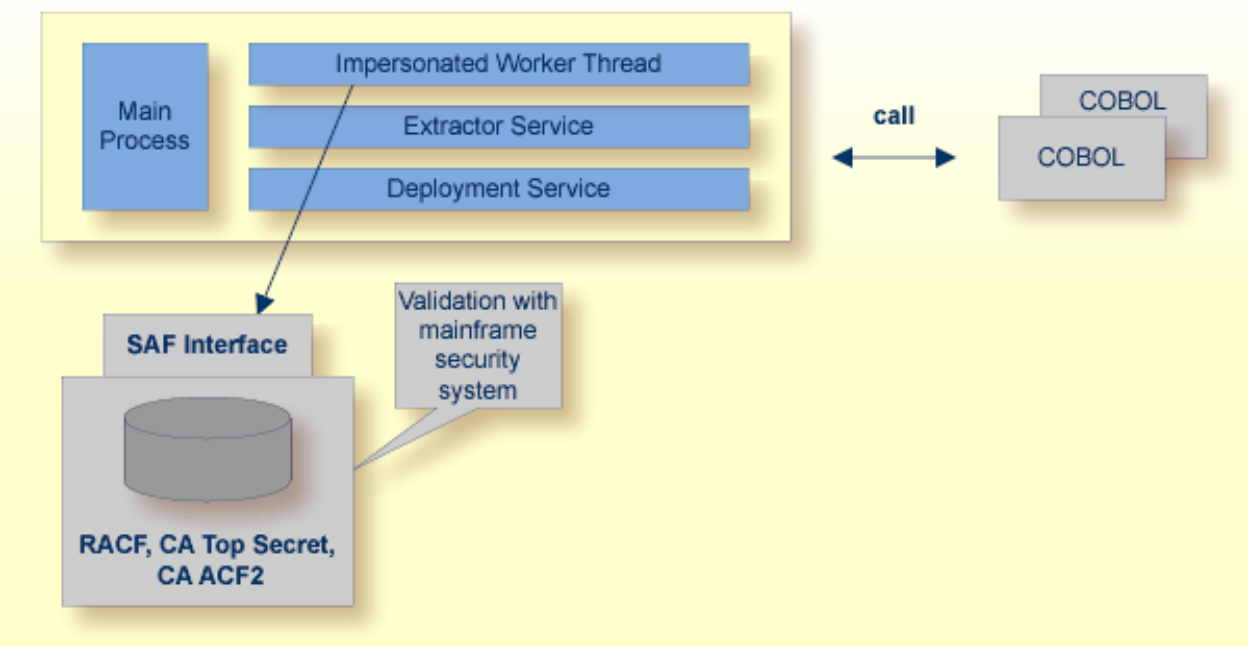


Deployment Service

The Deployment Service allows you to synchronize server-side mapping files (Designer files with extension .svm) interactively using the *Server Mapping Deployment Wizard*. Synchronizing or un-deploying server mapping files from the RPC server is part of *Migrating Server Mapping Files*. On the RPC server side, the server-side mapping files are stored in a server-side mapping container (VSAM file). See ***Server-side Mapping Files in the RPC Server*** and ***Deployment Service*** for configuration information.



Impersonation



The RPC Server for IMS can be configured to execute the RPC request impersonated under the RPC client user ID. This means that for the request execution, the worker thread gets the identity of the RPC client. This is necessary when accessing (security) protected data sets, for example with the *Extractor Service*. The way authentication is carried out can be controlled by the RPC parameter `impersonation`.

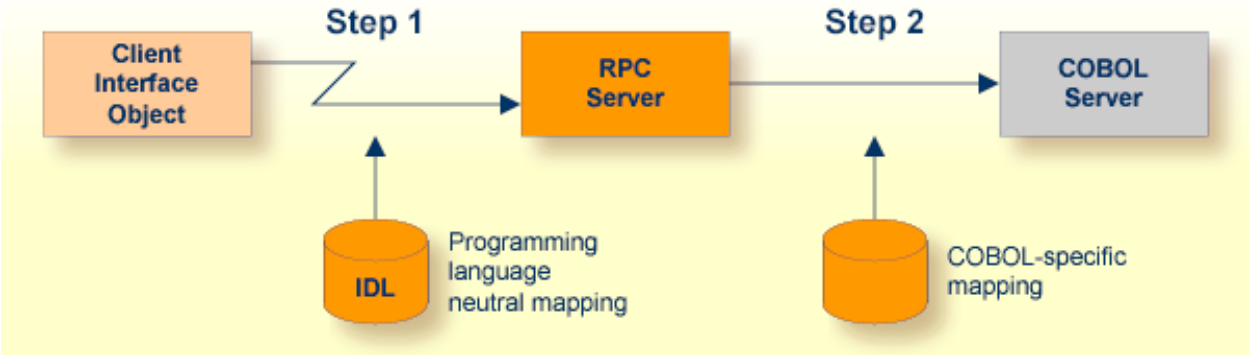
- For `impersonation` value `AUTO`, the RPC Server for IMS does not validate RPC passwords, so you have to make sure the RPC client is correctly authenticated, either by using a secure EntireX Broker (validation must be against the correct mainframe security repository where z/OS user IDs are defined) or with your own security implementation.
- For `impersonation` value `YES`, the RPC Server for IMS uses the RPC user ID and RPC password sent by the calling RPC client for authentication and impersonation of the client. This means that the RPC server validates the RPC password or - if a long RPC password is sent - as a RACF password phrase.

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

The lifetime of an impersonated 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.

Usage of Server Mapping Files

Server mapping files contain COBOL-specific mapping information that is not included in the IDL file, but is needed to successfully call the COBOL server program. There are many situations where the RPC Server for IMS requires a server mapping file to correctly support special COBOL syntax such as `REDEFINES`, `SIGN LEADING` and `OCCURS DEPENDING ON` clauses, `LEVEL-88` fields, etc.



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

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

Note: Server mapping files are used for COBOL only.

3 Administering the RPC Server for IMS

The EntireX RPC Server for IMS allows standard RPC clients to communicate with RPC servers on the operating system z/OS running with IMS in BMP mode. It supports the programming languages COBOL, PL/I and C and can provide IMS-specific PCB pointers for access to IMS databases if needed.

Customizing the RPC Server

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

- Configuration File
- IBM LE Runtime Options
- Started Task JCL

Configuration File

The name of the delivered example configuration file is `CONFIGI` (see source library EXP109.SRCE). The configuration file is specified as a DD definition with a user-defined DD name in the ***Started Task JCL***. The configuration file contains the configuration for the RPC Server for IMS. The following settings are important:

- connection information such as broker ID, server address (class, name, service)
- scalability parameters
- trace settings
- etc.

For more information see ***Configuring the RPC Server***.

IBM LE Runtime Options

Depending on the feature the RPC Server for IMS needs to support (see table below) additional runtime options for IBM's Language Environment need to be set. For a full description of LE runtime options, see ***IBM Z Publications Library Archive***.

Feature	LE Runtime Options	Description
Call RPC server programs with AMODE 24 as well	`ALL31(OFF),STACK(,,BELOW)`	If not specified, AMODE 31 is supported.

Note: [(1)] Set internally by the RPC Server for IMS and cannot be changed.

There are various ways to specify LE runtime options, for example during installation; using JCL; using `CSECT CEEUOPT` (for application-specific LE runtime options) linked to the RPC Server; etc. We recommend you use the IBM standard approach with `CEEOPTS DD` statement

in the started task JCL. See *Started Task JCL* for this purpose. Add the following lines to your started task JCL:

```
//...
//CEEOPTS  DD *
ALL31(OFF),STACK(,,BELOW)
/*
//..
```

The example above uses an in-stream data set to configure `ALL31(OFF),STACK(,,BELOW)` to allow calling of 24-bit and 31-bit programs and configure `RPTOPTS(ON)` to list all used LE runtime options to `SYSOUT`.

Started Task JCL

The name of the started task is `EXPSRVI` (see EntireX job library EXX109.JOBS). The started task contains the following:

- the target server libraries of the called COBOL or PL/I server
- for PL/I, if IMS-specific PCB pointers are used, the stub library; see `stublib`
- the configuration file used; see *Configuration File*; specified as a DD definition with a user-defined DD name as RPC server startup argument CFG:

```
CFG=DD:ddname
```

Under IMS, server startup arguments are passed with the DD name `ERXPARMS`. So to pass the argument for the example above, you need to specify a data set in the DD definition for `ERXPARMS`. This data set needs a line containing the CFG argument, for example `CFG=DD:CONFIGI`.

```
//*
//G.ERXPARMS DD *   ERXPARMS - DEFINE ONLY THE CONFIG DD NAME HERE
CFG=DD:CONFIGI
```

- LE runtime options used; see *IBM LE Runtime Options*
- etc.

Configuring the RPC Server

The following rules apply:

- In the configuration file:
  - Comments must be on a separate line.
  - Comment lines can begin with '*', '/' and ';'.
  - Empty lines are ignored.
  - Headings in square brackets [<topic>] are ignored.
  - Keywords are case-insensitive.
- Underscored letters in a parameter indicate the minimum number of letters that can be used for an abbreviated command.

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

Parameter	Default	Values	Req/ Opt
`brokerid`	`localhost`	Broker ID used by the server. See *Using the Broker ID in Applications* in the RPC Programming documentation. Example: `brokerid=myhost.com:1971`	R
`ceeoptions`		Allows you to change IBM's LE runtime options. This parameter is deprecated. See ***IBM LE Runtime Options*** for how to set the LE runtime options.	O
`class`	`RPC`	Server class part of the server address used by the server. The server address must be defined as a service in the broker attribute file (see *Service-specific Attributes*). Case-sensitive, up to 32 characters. Corresponds to `CLASS` attribute of the broker attribute file. Example: `class=MyRPC`	R
`codepage`	no codepage transferred	The codepage tells the broker the encoding of the data. The application must ensure the encoding of the data matches the codepage. The RPC server itself does not convert your application data. The application's data is shipped and received as given. Often, the codepage must also match the encoding used in the RPC server environment for file and terminal IO, otherwise unpredictable results may occur.	

Parameter	Default	Values	Req/ Opt
		By default, no codepage is transferred to the broker. It is assumed the broker's locale string defaults match. See *Locale String Mapping* If they do not match, provide the codepage here. Example: `codepage=ibm-273` Enable character conversion in the broker by setting the service-specific attribute `CONVERSION` to `"SAGTRPC"`. See also *Configuring ICU Conversion* under *Configuring Broker for Internationalization* in the platform-specific Administration documentation. More information can be found under *Internationalization with EntireX*.	

`compresslevel`	`N`	Enforce compression when data is transferred between broker and server. See *Data Compression in EntireX Broker* in the platform-independent Administration documentation.
		
		`compresslevel=0|1|2|3|4|5|6|7|8|9|Y|N`
		
		`0-9` 0=no compression
		9=max. compression
		`N` no compression
		`Y` compression level 6
		
		Example:
		`compresslevel=6`
			O

`deployment`	`NO`	Activates the deployment service, see ***Deployment Service***. Required to use the Server Mapping Deployment Wizard. See *Server Mapping Deployment Wizard* in the Designer documentation. `YES` Activates the deployment service. The RPC server registers the deployment service in the broker. `NO` The deployment service is deactivated. The RPC server does not register the deployment service in the broker. Example: `deployment=yes`	O
`etblnk`	`BROKER`	Define the broker stub to be used. See *Administering Broker Stubs* for available stubs. Example: `etblnk=broker`	O

Parameter	Default	Values	Req/ Opt
extractor	NO	The extractor service is a prerequisite for remote extractions. See ***Extractor Service***. extractor=YES|NO Example: extractor=yes	O
impersonation	NO	Defines if RPC requests are executed under the RPC user ID of the calling RPC client. ■ For how to send the RPC user ID/password pair from an RPC client, see *Using the Broker and RPC User ID/Password* (.NET Wrapper | Java Wrapper | C Wrapper | PL/I Wrapper | DCOM Wrapper | Web Services Wrapper | IDL Tester | Listener for XML/SOAP | Listener for IBM MQ). ■ For the COBOL Wrapper, refer to *Using Broker Logon and Logoff* and *Using RPC Authentication (Natural Security, Impersonation, Integration Server)*. ■ For non-RPC clients, see *Using the Broker and RPC User ID/Password* under *EntireX XML Tester* in the XML/SOAP Wrapper documentation. Depending on settings, different levels of checks are done prior to RPC server execution. See also ***Impersonation*** under *Introduction to the RPC Server for IMS*. impersonation=NO|YES|AUTO[,sameuser|,anyuser] NO The RPC request is executed anonymously, which means the user ID of the calling RPC client is not used. RPC requests are executed under the user ID of the RPC server. YES AUTO Same as option YES above, except that no password validation is performed, that is, the calling RPC client is treated as already authenticated. For this setting, make sure the RPC client is correctly authenticated; use either ■ a secure broker (validation must be against the correct mainframe security repository where the user IDs are defined) and option sameuser or ■ your own security implementation (option anyuser is supported for compatibility reasons if you need different broker and server user IDs - the customer-written security implementation must validate the calling RPC client using the *RPC client user ID*).	O

Parameter	Default	Values	Req/ Opt
		The following options are available for `AUTO`: ■ `sameuser` The RPC Server for IMS checks whether the *broker client user ID* matches the *RPC client user ID*. This is the default if `AUTO` is used. ■ `anyuser` The *RPC client user ID* is used for impersonation. The *broker client user ID* is ignored. Example: `impersonation=auto,anyuser` Using impersonation requires additional installation steps. See *Using z/OS Privileged Services* in the z/OS installation documentation.	
`library`	no default	`library = search_logic [- library]` where *search_logic* is one of `FIX(dllname)` | `PREFIX(prefix)` | `PREFIX()`, and *library* can be repeated up to four times, that is, five entries are possible. This parameter applies to programming language C only. Do not set if other programming languages for RPC server are used. `FIX(dllname)` The IDL library name coming from the RPC client is ignored, thus long IDL library names can be used. You have to define the DLL names for all client interface objects and RPC servers. `PREFIX(prefix)` The IDL library name coming from the RPC client is used to form the DLL name. As *prefix* you can define any character. If an RPC client sends, for example, `"SYSTEM"` as the IDL library name and `"D"` is defined as *prefix*, the DLL name derived is `"DSYSTEM"`. This configuration restricts the IDL library names to max. 7 characters. `PREFIX()` The IDL library name coming from the RPC client is used as DLL name. This configuration restricts the IDL library names to max. 8 characters.	O

Parameter	Default	Values	Req/ Opt
		Example PREFIX configuration (this configuration matches the standard names produced by the C Wrapper): library=PREFIX(D)-PREFIX() Example FIX configuration: library=FIX(MYSTUBS)-FIX(MYRPCS)	
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: logon=no	O
marshalling	COBOL	The RPC Server for IMS can be configured to support either COBOL, PL/I or C. See also ***Locating and Calling the Target Server***. marshalling=(LANGUAGE=COBOL|PLI|C) COBOL Server supports COBOL. The COBOL servers are called directly without a server interface object. So-called server mapping files are used to call the COBOL server correctly if one is available. See ***Usage of Server Mapping Files***. PLI Server supports PL/I. See prerequisites for PL/I Wrapper. C Server supports C. The modules are called using a server interface object built with the C Wrapper.	O
password	no default	The password for secured access to the broker. If possible (write access) the password is encrypted and written to parameter password.e. The parameter password is removed. To change the password, add the parameter password with the new password as value. If the ***Configuration File*** is an inline data set in the ***Started Task JCL***, password encryption does not take place. Example: password=MyPwd	O
restartcycles	15	Number of restart attempts if the broker is not available. This can be used to keep the RPC Server for IMS running while the broker is down for a short time. A restart cycle will be repeated every 60 seconds. **Note:** Internally, the server waits in periods of 10 seconds (performing six times more cycles), which you can see in the server output.	O

Parameter	Default	Values	Req/ Opt
		When the number of specified cycles is reached and a connection to the broker is not possible, the RPC Server for IMS stops. Example: `restartcycles=30` The server waits up to 30 minutes (30*6*10 seconds) before it terminates due to a missing broker connection.	

return_code	NO	Enable application-specific errors.
		`return_code=(NO|YES)`
		
		`NO` No tests of COBOL special register `RETURN-CODE` for application-provided error.
		`YES` After execution of the RPC server, tests COBOL special register `RETURN_CODE` for application provided error. See ***Returning Application Errors***.
		
		Example:
		`return_code=yes`
			O

runoption	no default	This parameter is for special purposes. It provides the RPC Server for IMS 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. Example: `runoption=<option>` `runoption=<option>`	O
servername	SRV1	Server name part of the server address used by the server. The server address must be defined as a service in the broker attribute file. See *Service-specific Attributes*. Case-sensitive, up to 32 characters. Corresponds to `SERVER` of the broker attribute file. Example: `servername=mySrv`	R
service	CALLNAT	Service part of the server address used by the server. The server address must be defined as a service in the broker attribute file. See *Service-specific Attributes*. Case-sensitive, up to 32 characters. Corresponds to `SERVICE` attribute of the broker attribute file. Example: `service=MYSERVICE`	R
stublib	no default	For PL/I server if its interface has ***IMS-specific PCB Pointers*** to access IMS databases. Provide the DD name of the library where the PL/I server interface object(s) are located. Not required for	O

Parameter	Default	Values	Req/ Opt

		COBOL or if the interface of your PL/I server does not have IMS-specific PCB pointers.
		
		`stublib=`*`ddname`*
		
		Example:
		`stublib=PLISTUBS`
		
		For the example above, define the DD name `PLISTUBS` in the started task JCL of the RPC Server for IMS (see ***Started Task JCL***) as
		
		`//PLISTUBS  DD DISP=SHR,DSN=<plistubs>`	

`svm`	`ERXSVM`	Usage and location of server-side mapping files (Designer files with extension .svm) at runtime; see ***Server-side Mapping Files in the RPC Server***. If the `svm` parameter is omitted, the RPC server tries to open the server-side mapping container using DD name `ERXSVM`. If this DD name is not available, no server-side mapping files are used. If you use server-side mapping files at runtime, the server-side mapping container must be installed and configured; see *Installing the Server-side Mapping Container for an RPC Server for IMS (Optional)* in the z/OS Installation documentation.
		
		Server mapping files with extension .svm are no longer supported at design time by the Designer. You can still use them at runtime in a server-side mapping container. All special COBOL syntax and features supported by server mapping files with extension .svm are also covered by server mapping files with extension .cvm. See *When is a Server Mapping File Required?*
		We recommend migrating .svm files to .cvm files. See *Migrating Server Mapping Files* under *Server Mapping Files for COBOL* in the Designer documentation.
		
		`svm = no|`*`ddname`*
		
		`no` No server-side mapping files are used.
		`ddname` DD name of the server-side mapping container in the started task JCL of the RPC Server for IMS.
		
		Example:
		`svm=MYSVM`
		
		For the example above, define the DD name `MYSVM` in the JCL of the RPC Server for IMS (see ***Started Task JCL***) as:
		`//MYSVM DD DISP=SHR,DSN=<svm.cluster>`
			O

Parameter	Default	Values	Req/ Opt
		See also *Usage of Server Mapping Files*.	
timeout	60	Timeout in seconds, used by the server to wait for broker requests. See WAIT under *Broker ACI Fields* for more information. Also influences restartcycles. Example: timeout=300	O
tracedestination	DD:ERXTRACE	The name of the destination file for trace output. See also *Activating Tracing for the RPC Server*. tracedestination=DD:*ddname*, where *ddname* is the name of the trace file. Example: tracedestination=DD:MYTRACE The DD name MYTRACE must be defined in the started task of the RPC Server for IMS (see *Started Task JCL*): //MYTRACE DD DISP=SHR,DSN=<rpctrace-file>	O
tracelevel	None	Trace level for the server. See also *Activating Tracing for the RPC Server*. tracelevel=None|Standard|Advanced|Support None — No trace output. Standard — For minimal trace output. Advanced — For detailed trace output. Support — This trace level is for support diagnostics. Use only when requested by Software AG Support. Example: tracelevel=standard	O
traceoption	None	Additional trace option if trace is active. See also *Activating Tracing for the RPC Server*. None — No additional trace options. STUBLOG — If tracelevel is Advanced or Support, the trace additionally activates the broker stublog.	O

Parameter	Default	Values	Req/ Opt
		NOTRUNC Normally if a data buffer larger than 8 KB is traced, the buffer trace is truncated. Set this option to write the full amount of data without truncation. **Note:** This can increase the amount of trace output data dramatically if you transfer large data buffers. Example: `traceoption=(STUBLOG,NOTRUNC)`	
`userid`	`ERX-SRV`	The user ID for access to the broker. The default `ERX-SRV` will be used if this parameter is omitted or specified without a value: "`userid=`". Example: `userid=MyUid`	O

Locating and Calling the Target Server

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

- COBOL
- C
- Assembler (IBM 370)

COBOL

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

To use the RPC Server for IMS with COBOL

1 Make sure that all z/OS modules called as RPC servers

- are compiled with IBM's Language Environment (see ***IBM Z Publications Library Archive*** for more information)
- use COBOL calling conventions

- can be called dynamically ("fetched") from any Language Environment program
- are accessible through the RPC Server for IMS started task JCL `STEPLIB` concatenation. See ***Started Task JCL***.

2 Configure the parameter `marshalling` for COBOL, for example:

```
marshalling=COBOL
```

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

C

The approaches needed to derive the dynamic-link library (DLL) names for the RPC server are more complex for C, for the following reasons:

- the limitation of 8 characters per (physical) member (DLL name in PDSE)
- the maximum length of 128 characters per IDL library name (see *Rules for Coding Library, Library Alias, Program, Program Alias and Structure Names*).

Either you restrict yourself in short IDL library names (up to 8 characters) and use the flexible `PREFIX` configuration, or, if you need independence from the IDL library length and names, use the `FIX` configuration. The parameter `library` is used for this purpose.

To use the RPC Server for IMS with C

1 Make sure all dynamic-link libraries (DLLs) called as RPC servers and client interface objects are accessible through the RPC Server for IMS started task JCL `STEPLIB` concatenation. See *Started Task JCL*.

2 Configure the parameter `marshalling` for C, for example `marshalling=C`.

3 Configure the parameter `library` either with the `FIX` configuration or `PREFIX` configuration, depending on how you have built your DLLs. See *Using the C Wrapper for the Server Side (z/OS, Linux, Windows, BS2000)*.

See also ***Scenario V: Writing a New C Server***.

Assembler (IBM 370)

There is a simple mechanism to derive the RPC server z/OS module name:

- The IDL program name is used as the z/OS module name
- The IDL library name is not used.

To use the RPC Server for IMS with Assembler

- Make sure all z/OS modules called as RPC Servers
  - are accessible through the RPC Server for IMS started task JCL `STEPLIB` concatenation. See *Started Task JCL*.
  - Use PL/I or COBOL calling conventions. Configure the parameter `marshalling` for PL/I or COBOL.

See also ***Scenario VI: Writing a New Assembler Server***.

Using SSL/TLS with the RPC Server

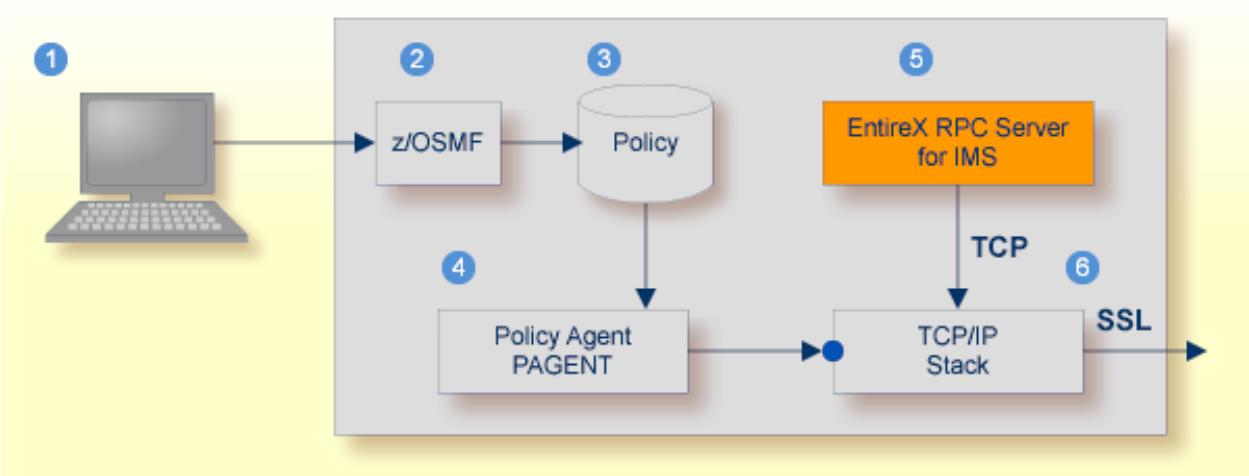
RPC servers can use Secure Sockets Layer/Transport Layer Security (SSL/TLS) as the transport medium. The term "SSL" in this section refers to both SSL and TLS. RPC-based servers are always SSL clients. The SSL server can be either the EntireX Broker, Broker SSL Agent, or Direct RPC in webMethods Integration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

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

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

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

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



1. Client to interact with z/OS Management Facility (z/OSMF).
2. AT-TLS rules are defined with z/OSMF policy management.
3. Policy Repository with AT-TLS rules stored as z/OS files.
4. Policy Agent, MVS task PAGENT, provides AT-TLS rules through a policy enforcement point (PEP) to TCP/IP stack.
5. Application using TCP connection.
6. If AT-TLS rules match, the TCP connection is turned into an SSL connection.

Notes:

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

To set up SSL with AT-TLS

1 To operate with SSL, certificates need to be provided and maintained. Depending on the platform, Software AG provides sample certificates, but we strongly recommend that you create your own. See *SSL/TLS Sample Certificates Delivered with EntireX* in the EntireX Security documentation.

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

```
ETB024:1699:TCP
```

3 Configure AT-TLS to turn the TCP/IP connection to an SSL connection, using a client to interact with the z/OS Management Facility (z/OSMF) See *z/OSMF Considerations* in the z/OS Administration documentation. The outcome of this configuration is a Policy Repository with AT-TLS rules stored as z/OS files. This file is the configuration file for the Policy Agent, MVS task `PAGENT`.

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

- *Running Broker with SSL/TLS Transport* in the platform-specific Administration documentation
- Broker SSL Agent in the platform-specific Administration documentation
- *Support for SSL/TLS* in the EntireX Adapter documentation (for Direct RPC)

Starting the RPC Server

To start the RPC Server for IMS

1 Modify the member `EXPSRVI` (see EntireX job library EXX109.JOBS) according to your system requirements and copy the started task JCL to your system `PROCLIB` concatenation. See ***Started Task JCL***.

2 Modify the server parameters ***Configuration File*** according to your system requirement. For details, see ***Configuring the RPC Server***.

3 Start the task manually with

```
/s EXPSRVI
```

Or:

Add the task to your system automation tool(s)

Stopping the RPC Server

To stop the RPC Server for IMS

- Use the operator command `STOP`. Examples:

```
/p EXPSRVI
/f EXPSRVI,STOP
```

Or:

Add the `STOP` command to your system automation tool(s).

Activating Tracing for the RPC Server

To switch on tracing for the RPC Server for IMS

1. Set the parameters `tracelevel`, `traceoption` and `tracedestination`. See ***Configuring the RPC Server***.
2. Start the RPC Server for IMS. See ***Starting the RPC Server***.
3. Temporarily change the trace level with the operator command

```
F EXPSRVI,TRACELEVEL=tracelevel,
```

for valid *tracelevel* values, see `tracelevel`.

The `TRACELEVEL` command without any value will report the currently active trace options, for example:

```
F EXPSRVI,TRACELEVEL
```

might reply with the operator message

```
Tracelevel=0 TraceFile=DD:ERXTRACE
```

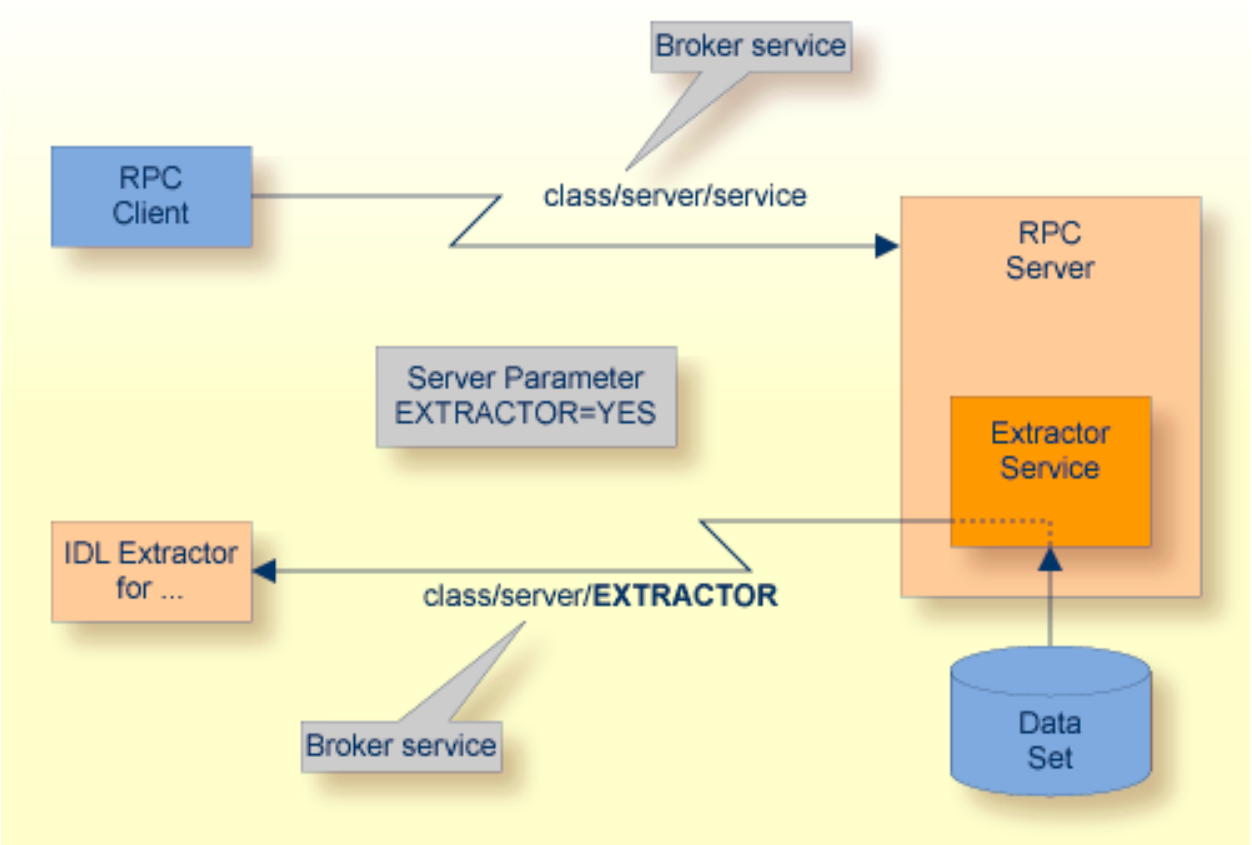
To switch off tracing

- Set the `tracelevel` parameter to `None`.

4 Extractor Service

Introduction

- The Extractor Service provides access to PDS data sets and CA Librarian DA master files defined within the z/OS catalog.
- The Extractor Service provides access to security-protected data sets (protected e.g. by RACF, CA ACF2, CA Top Secret).
- It is a built-in service of the RPC Server for IMS and can be enabled/disabled by RPC server configuration settings.
- Depending on the platform where the broker is running, you can restrict usage of the Extractor Service to certain users or group of users, using EntireX Security. See *Authorization of Client and Server* in the EntireX Security documentation.



Scope

The Extractor Service is a prerequisite for the

- **IDL Extractor for COBOL**
  used together with a remote extractor environment, see *Step 2: Select a COBOL Extractor Environment or Create a New One*.
- **IDL Extractor for PL/I**
  used together with an RPC environment, see *RPC Environment Manager* in the IDL Extractor for PL/I documentation

The Extractor Service uses the same class and server names as defined for the RPC server, and "EXTRACTOR" as the service name, resulting in *class*/*server*/EXTRACTOR as the broker service.

Note: EXTRACTOR is a service name reserved by Software AG. See SERVICE under *Service-specific Broker Attributes*.

CA Librarian Support

- Supported Features:
  - Traditional CA Librarian DA master files are supported.
  - Extraction from multiple CA Librarian data sets is possible.
  - CA Librarian archive levels (history) are supported for the COBOL (main) source where the extraction starts.
  - Security RACROUTE is supported.
  - PDS data sets and CA Librarian data sets can be mixed, that is:
    - the COBOL source can reside in a PDS, and some copybooks in CA Librarian, and others in PDS
    - the COBOL source can reside in CA Librarian, and some copybooks in PDS and others in CA Librarian
    - -INC and COBOL COPY statements can be mixed in one source

Enabling the Extractor Service

To enable the Extractor Service

1 Set the RPC Server for IMS parameter `extractor=yes`. See `extractor` under ***Configuring the RPC Server***.

2 Define in the broker attribute file, under the RPC service, an additional broker service with "`EXTRACTOR`" 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 extractor service requires the following additional service definition in the Broker attribute file:

```
CLASS = RPC    SERVER = SRV1    SERVICE = EXTRACTOR
```

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

- For a z/OS broker, see *Resource Profiles in EntireX Security* under *EntireX Security under z/OS* in the EntireX Security documentation.
- For a Linux or Windows broker, see *Authorization Rules* in the platform-independent Administration documentation.
- Not applicable to a BS2000 broker.

4 Optional. Use the `impersonation` feature of the RPC Server for IMS to enable access to security-protected data sets (protected e.g. by RACF, CA ACF2, CA Top Secret). See `impersonation` under ***Configuring the RPC Server***.

Disabling the Extractor Service

To disable the extractor service

- Set the RPC Server for IMS parameter `extractor=no`. See `extractor` under ***Configuring the RPC Server***.

  The RPC Server for IMS will not register the extractor service in the broker.

Restrictions

The following restrictions apply to CA Librarian:

- Filtering with programmer and type, as is done by the CA Librarian ELIPS (Extended Librarian Interactive Productivity Services) application, is not supported.
- CA Librarian Wide Record Master Files (PDS/E - PO) are not supported.
- CA Librarian MCD Security is not supported
- CA Librarian member passwords (NOBYPP installations) are not supported
- The optional syntax elements `seq1`, `seq2` and `ARC` of the CA Librarian `-INC module-name[,seq1[,seq2][,ARC={date | Lx | -y}]]` statement are not supported. Therefore CA Librarian archive levels (history) are not supported for COBOL copybooks. It is always the most recent member (last update) that is delivered by the extractor service.

No access is provided to other data sets (e.g. CA Panvalet) or to data sets not defined in the z/OS catalog (e.g. defined in VTOC only).

5 Deployment Service

Introduction

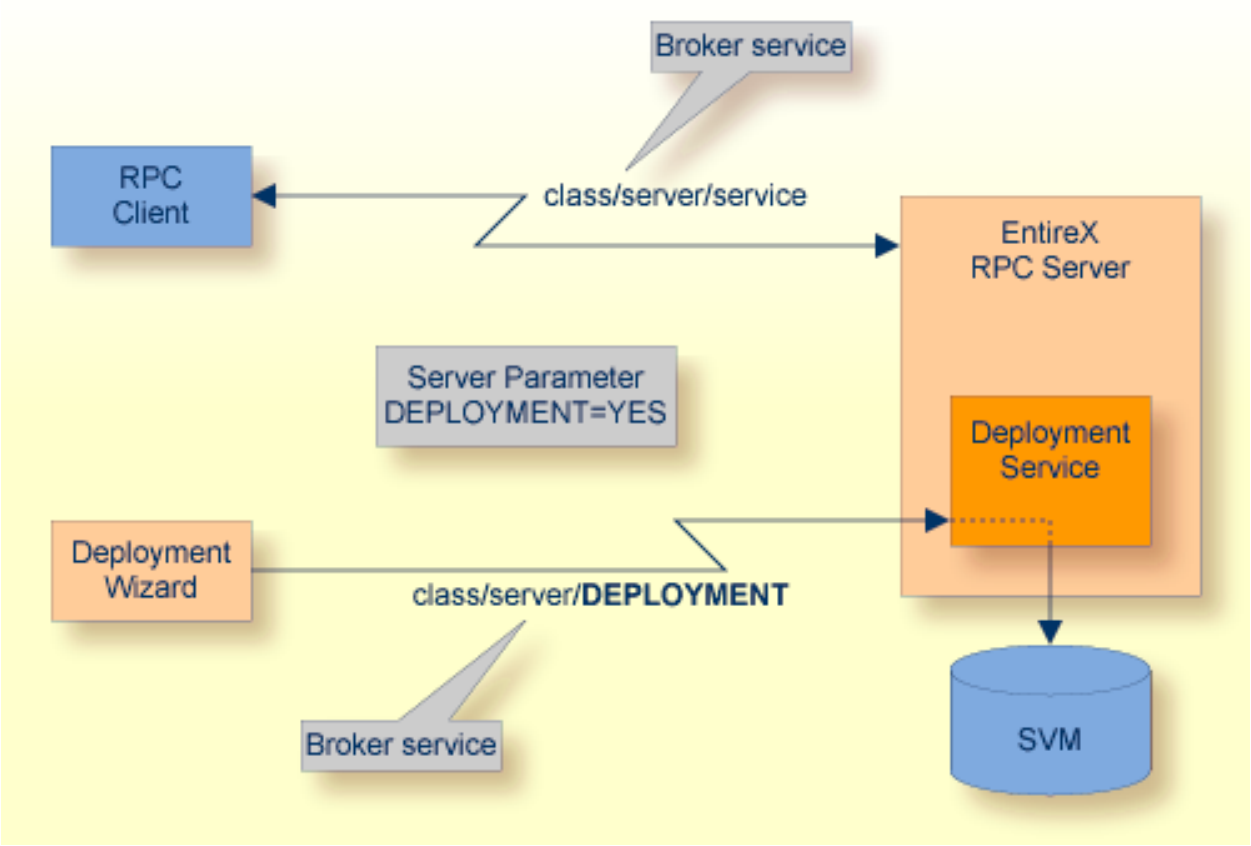
The deployment service is the (server-side) counterpart to the deployment wizard; see *Server Mapping Deployment Wizard*, the tool used when *Migrating Server Mapping Files*. It is a built-in service of the EntireX RPC server, which can be enabled/disabled by EntireX RPC server configuration settings.

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

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

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

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



Scope

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

The deployment service uses the same class and server names as defined for the EntireX RPC server, and `DEPLOYMENT` as the service name, resulting in *`class`*`/`*`server`*`/DEPLOYMENT` as the broker service.

Note: `DEPLOYMENT` is a service name reserved by Software AG. See broker attribute `SERVICE`.

Enabling the Deployment Service

To enable the deployment service

1 For an RPC Server for IMS, the server-side mapping container (VSAM file) must be installed and configured. See *Installing the Server-side Mapping Container for an RPC Server for IMS (Optional)* in the z/OS Installation documentation.

2 Set the RPC server parameter `deployment=yes`. See `deployment` under ***Configuring the RPC Server***.

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

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

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

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

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

- For a z/OS broker, see *Resource Profiles in EntireX Security*.
- For a Linux or Windows broker, see *Authorization Rules* in the platform-independent Administration documentation.
- Not applicable to a BS2000 broker.

Disabling the Deployment Service

To disable the deployment service

- Set the RPC Server for IMS parameter `deployment=no`. See `deployment` under ***Configuring the RPC Server***.

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

6 Server-side Mapping Files

Server-side mapping files have the extension .svm.

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

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

Server-side Mapping Files in the RPC Server

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

- If *one* server requires a server-side mapping file (.svm), you need to provide the server-side mapping container to the RPC server. See configuration parameter `svm`.
- If *no* server requires server-side mapping (.svm) or you use server mapping files with extension .cvm only, you can execute the RPC server without the server-side mapping container.

Undeploying Server-side Mapping Files from the RPC Server

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

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

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

2 Make sure your IDL file is within a Designer directory (folder) without the related server-side mapping file (.svm) or renamed to extension .cvm if you are *Migrating Server Mapping Files*.

3 From the context menu of your IDL file, choose **COBOL > Deploy/Synchronize Server Mapping** and call the Server Mapping Deployment Wizard. See *Server Mapping Deployment Wizard* in the Designer documentation. Because there is no related server-side mapping file

(.svm) in the Designer, all server mapping information related to the IDL file in the server-side mapping container of the RPC server will be removed.

Change Management of Server-side Mapping Files

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

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

List Deployed Server-side Mapping Files

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

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

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

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

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

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

Access Control: Secure Usage of Server Mapping Deployment Wizard

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

7 Scenarios and Programmer Information

COBOL Scenarios

Scenario I: Calling an Existing COBOL Server

To call an existing COBOL server

1 Use the IDL Extractor for COBOL to extract the Software AG IDL and, depending on the complexity, also a server mapping file. See *When is a Server Mapping File Required?* under *Server Mapping Files for COBOL* in the Designer documentation. If your COBOL server uses PCB pointers, see ***IMS-specific PCB Pointers***.

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

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

See *Client and Server Examples for z/OS IMS BMP* in the COBOL Wrapper documentation for COBOL RPC Server examples.

Scenario II: Writing a New COBOL Server

To write a new COBOL server

1 Use the COBOL Wrapper to generate a COBOL server skeleton and, depending on the complexity, also a server mapping file. See *When is a Server Mapping File Required?* under *Server Mapping Files for COBOL* in the Designer documentation. Write your COBOL server and proceed as described under *Using the COBOL Wrapper for the Server Side*. If your COBOL server uses PCB pointers, see ***IMS-specific PCB Pointers***.

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

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

See *Client and Server Examples for z/OS IMS BMP* in the COBOL Wrapper documentation for COBOL RPC Server examples.

PL/I Scenarios

Scenario III: Calling an Existing PL/I Server

To call an existing PL/I server

1 Use the IDL Extractor for PL/I to extract the Software AG IDL.

2 If your PL/I server uses PCB pointers, generate one or more server interface objects, using the IDL File extracted in Step 1 above. For more information see *Extraction Result* in the IDL Extractor for PL/I documentation. See also ***IMS-specific PCB Pointers***.

3 Build an EntireX RPC client using any EntireX wrapper. For a quick test you can:

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

See *Client and Server Examples for z/OS IMS BMP* in the PL/I Wrapper documentation for PL/I RPC Server examples.

Scenario IV: Writing a New PL/I Server

To write a new PL/I server

1 Use the PL/I Wrapper to generate a PL/I server skeleton. Write your PL/I server and proceed as described under *Using the PL/I Wrapper for the Server Side*. If your PL/I server uses PCB pointers, see ***IMS-specific PCB Pointers***.

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

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

See *Client and Server Examples for z/OS IMS BMP* in the PL/I Wrapper documentation for PL/I RPC Server examples.

C Scenarios

Scenario V: Writing a New C Server

To write a new C server

1 Use the C Wrapper to generate a C server skeleton and a C server interface object. Write your C server and proceed as described under *Using the C Wrapper for the Server Side (z/OS, Linux, Windows, BS2000)*.

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

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

Assembler Scenarios

Scenario VI: Writing a New Assembler Server

To write a new Assembler (IBM 370) server

1 Build an RPC server in Assembler. Here are some hints:

- The RPC server is dynamically callable (no pre-initialization required).
- The parameter interface is either compatible with the COBOL or PL/I calling convention (IDL level parameter will be passed in the address list).
- The alignment of integer or float data types is considered. The HASM Assembler aligns integer or float data types to appropriate boundaries. For example:

```
...
      MyLabel   DSECT
      MyField1  DS   H                      I2
      MyField2  DS   F                      I4
      MyField3  DS   E                      F4
      MyField4  DS   L                      F8
```

- The RPC Server for IMS will not align these data types by default.

- To force alignment by definition in your IDL file (see the aligned attribute within the `attribute-list` under *Software AG IDL Grammar* in the IDL Editor documentation) before generating your RPC client. For information on whether your client supports the aligned attribute, see *Mapping the `ALIGNED` Attribute* in the respective Wrapper documentation.

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

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

Returning Application Errors

Using RETURN-CODE Special Register (COBOL only)

The `RETURN-CODE` special register (an IBM extension to the COBOL programming language) is used by your RPC server to report an error.

Upon return, the value contained in the `RETURN-CODE` special register is detected by the RPC Server for IMS and sent back to the RPC client instead of the application's data.

For IBM compilers the `RETURN-CODE` special register has the implicit definition:

```
RETURN-CODE GLOBAL PICTURE S9(4) USAGE BINARY VALUE ZERO
```

Special registers are reserved words that name storage areas generated by the compiler. Their primary use is to store information produced through specific COBOL features. Each such storage area has a fixed name, and must not be defined within the program. See your compiler documentation for more information.

The following rules apply to application error codes:

- The value range for application errors is 1-9999. No other values are allowed.
- On the RPC client side, the error is prefixed with the error class 1002 “Application User Error” and presented as error 1002*nnnn*.
- No application data is sent back to the RPC client in case of an error.
- It is not possible to return an error text to the RPC client.

Example

```
. . .
        IF error occurred THEN
               MOVE <error-number> TO RETURN-CODE
               GO TO MAIN-EXIT
        END-IF.
        . . .

   MAIN-EXIT.
       EXIT PROGRAM.
END PROGRAM RETCODE.
```

Note: To enable this feature, configure the RPC Server for IMS with `return_code=yes`.

Automatic Syncpoint Handling

The RPC Server for IMS issues a `SYNC | ROLB` call under the following circumstances:

- The server issues an `IMS SYNC` call after a successful non-conversational request or an end-of-conversation.
- After abnormal termination of a non-conversational request or a conversation due to an error, the server performs an `IMS ROLB` call to back out any pending database modifications.