

## **webMethods EntireX**

### **EntireX z/OS IMS RPC Server**

Version 9.6

April 2014

This document applies to webMethods EntireX Version 9.6.

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

Copyright © 1997-2014 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-IMSRPC-96-20140628**

## Table of Contents

1 Introduction to the IMS RPC Server .....	1
IMS-specific PCB Pointers .....	3
Inbuilt Services .....	4
Impersonation .....	7
Usage of SVM Files .....	8
2 Administering the EntireX RPC Server under z/OS IMS .....	9
Customizing the RPC Server .....	10
Configuring the RPC Server .....	12
Locating and Calling the Target Server .....	20
Using SSL or TLS with the RPC Server .....	23
Starting the RPC Server .....	26
Stopping the RPC Server .....	26
Activating Tracing for the RPC Server .....	27
3 Extractor Service .....	29
Introduction .....	30
Scope .....	31
CA Librarian Support .....	31
Enabling the Extractor Service .....	32
Disabling the Extractor Service .....	32
Restrictions .....	33
4 Deployment Service .....	35
Introduction .....	36
Scope .....	37
Enabling the Deployment Service .....	37
Disabling the Deployment Service .....	38
5 Handling SVM Files .....	39
SVM Files in the EntireX Workbench .....	40
SVM Files in the RPC Server .....	40
Source Control of SVM Files .....	41
Change Management of SVM Files .....	41
Compare SVM Files .....	41
List Deployed SVM Files .....	41
Check if an SVM File Revision has been Deployed .....	42
Access Control: Secure SVM File Deployment .....	42
Ensure that Deployed SVM Files are not Overwritten .....	42
When is an SVM File Required? .....	43
Is There a Way to Smoothly Introduce SVM Files? .....	45
6 Scenarios and Programmer Information .....	47
COBOL Scenarios .....	48
PL/I Scenarios .....	49
C Scenarios .....	50
Assembler Scenarios .....	50
Aborting RPC Server Customer Code and Returning Error to RPC Client .....	51

Automatic Syncpoint Handling ..... 52

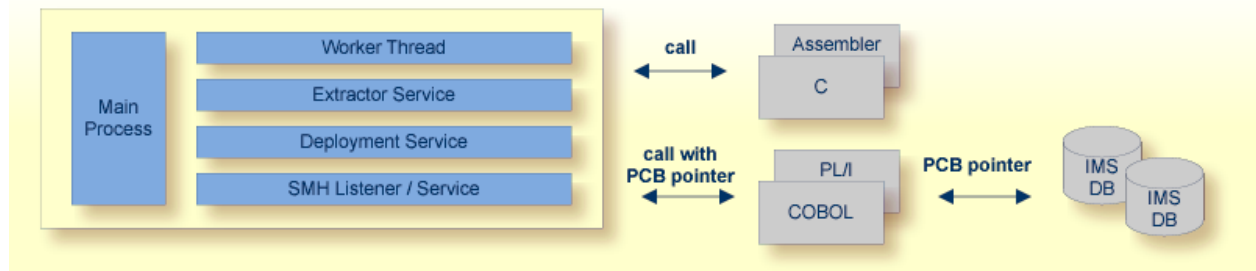
# 1 Introduction to the IMS RPC Server

---

▪ IMS-specific PCB Pointers .....	3
▪ Inbuilt Services .....	4
▪ Impersonation .....	7
▪ Usage of SVM Files .....	8

The EntireX z/OS IMS RPC Server 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 IMS RPC Server 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 (SVM) files, thus an SVM file is always required. See [Usage of SVM 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 `stubl` to the IMS RPC Server.

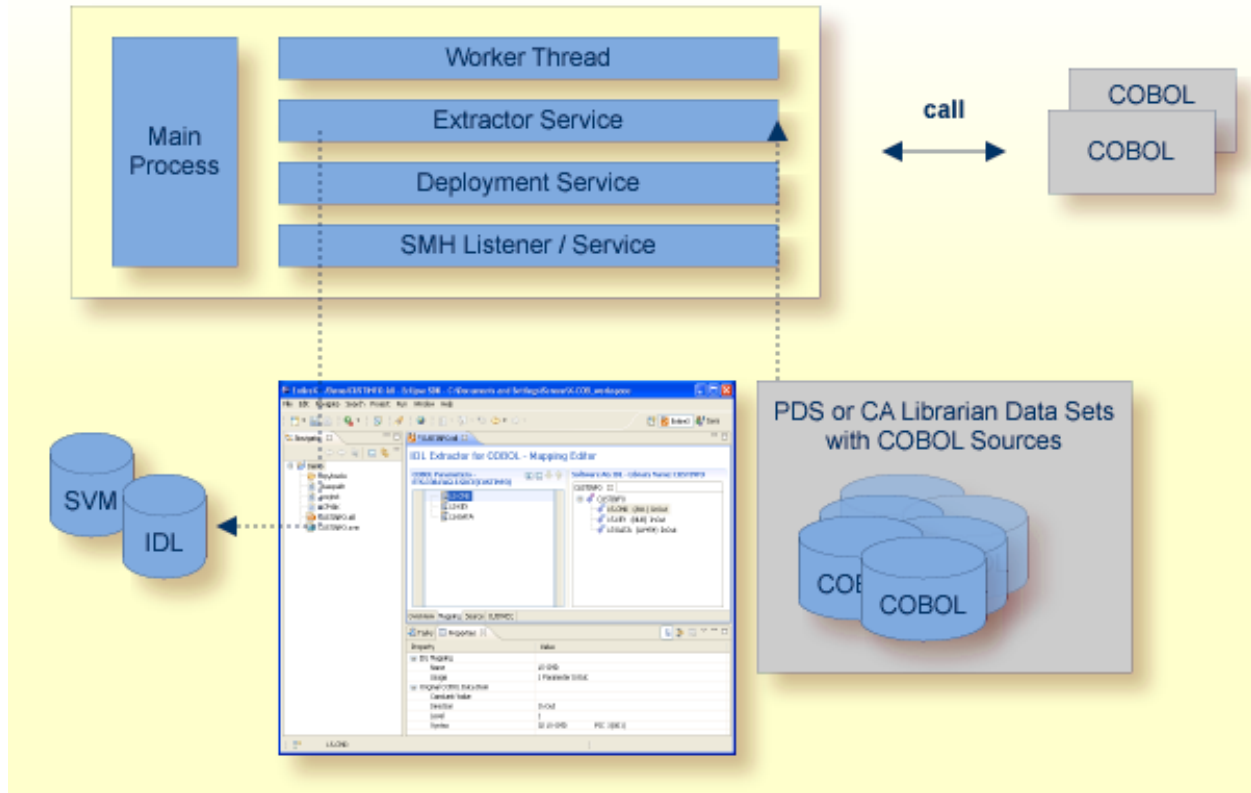
## Inbuilt Services

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

- Extractor Service
- Deployment Service
- SMH Listener Service

### Extractor Service

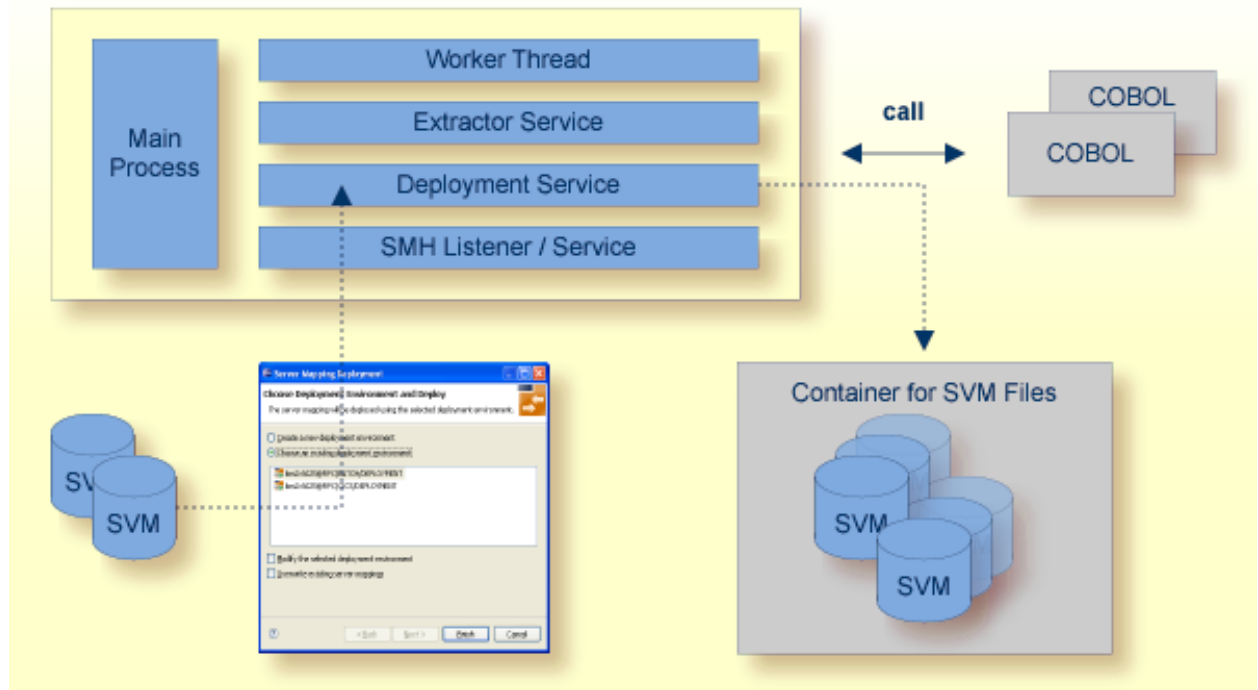
The Extractor Service is a prerequisite for remote extractions with the *IDL Extractor for COBOL* and *IDL Extractor for PL/I*. See [Extractor Service](#) for more information.





## Deployment Service

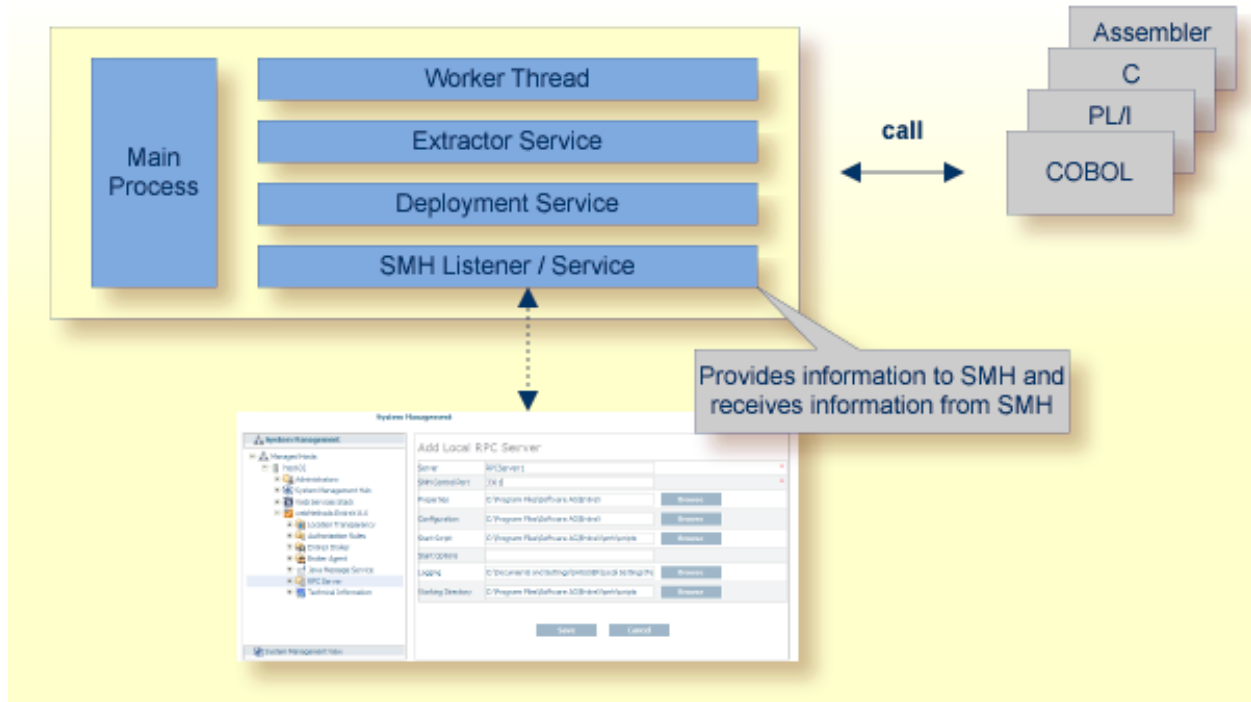
The Deployment Service allows you to deploy server mapping files (SVM files) interactively using the Deployment Wizard (see *Server Mapping Deployment*). On the RPC server side, the SVM files are stored in a VSAM file as the container. See *Deployment Service* for configuration information.



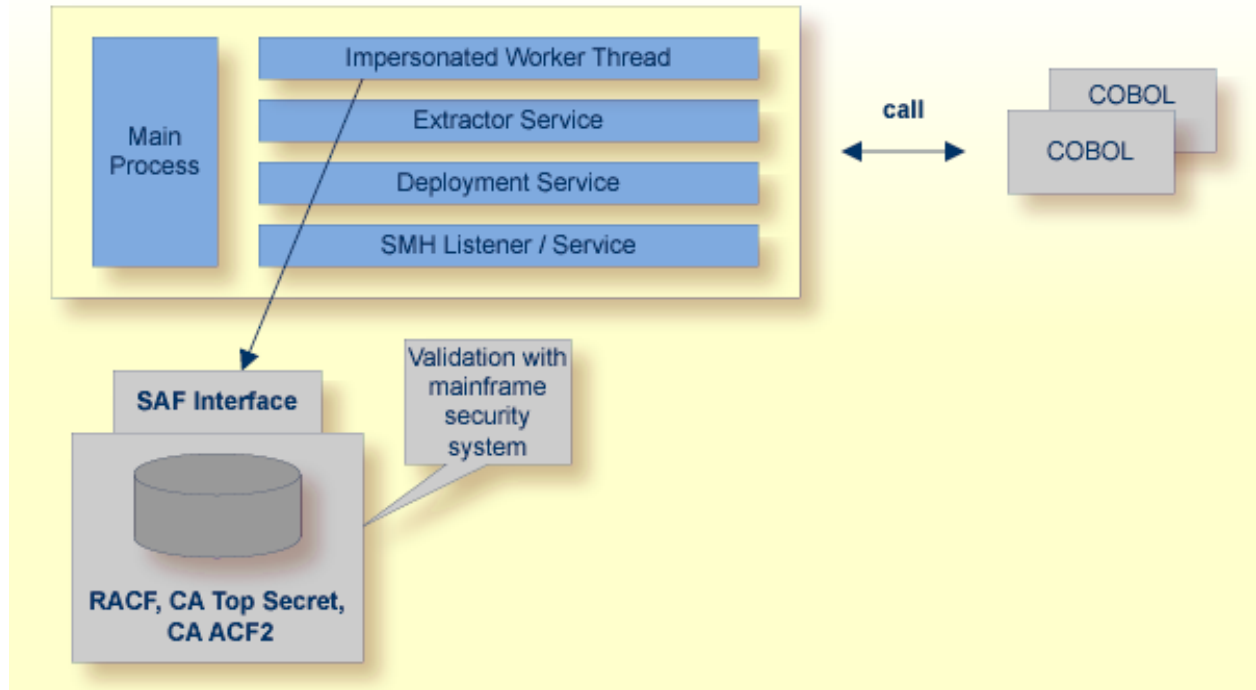
## SMH Listener Service

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 Service is switched on if the parameter `smhport` is set. See parameter `smhport` under *Configuring the RPC Server*.



## Impersonation



The IMS RPC Server 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 IMS RPC Server 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 IMS RPC Server uses the RPC user ID and password supplied by the RPC client for authentication and impersonation of the client. This means that the RPC server validates the password.

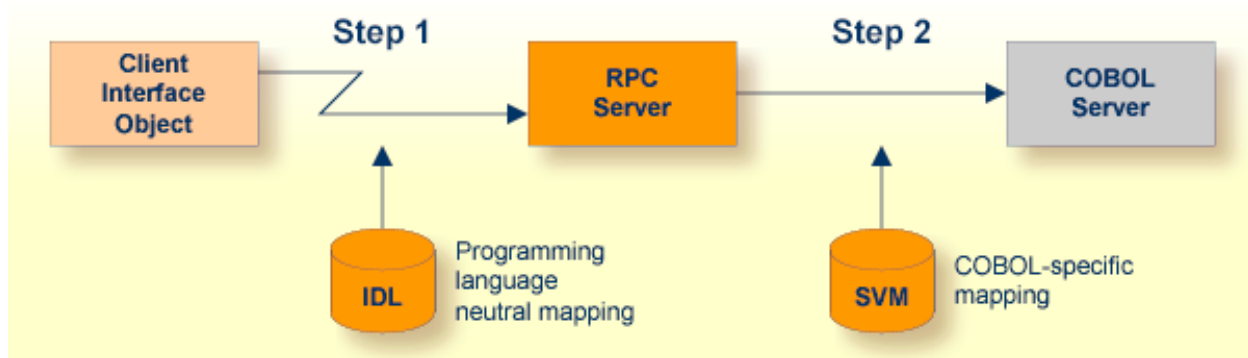
The picture above shows the configuration `impersonation=yes`.

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 SVM Files

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

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 [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 parameter `svm`. See [Configuring the RPC Server](#).



**Note:** SVM files are used for COBOL only.

## 2 Administering the EntireX RPC Server under z/OS IMS

---

▪ Customizing the RPC Server .....	10
▪ Configuring the RPC Server .....	12
▪ Locating and Calling the Target Server .....	20
▪ Using SSL or TLS with the RPC Server .....	23
▪ Starting the RPC Server .....	26
▪ Stopping the RPC Server .....	26
▪ Activating Tracing for the RPC Server .....	27

The EntireX z/OS IMS RPC Server 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 IMS RPC Server:

- [Configuration File](#)
- [IBM LE Runtime Options](#)
- [Started Task JCL](#)

### Configuration File

The name of the delivered example configuration file is `CONFIGI` (see source library `EXP960.SRCE`). The configuration file contains the configuration for the IMS RPC Server. The following settings are important:

- connection information such as broker ID, server address (class, name, service)
- location and usage of server mapping files (SVMs)
- scalability parameters
- trace settings
- etc.

The configuration file is specified as a DD definition with a user-defined DD name in the started task JCL. The DD name is passed to the RPC server with the following server startup argument:

```
CFG=DD: ddname
```

Example using the DD name `CONFIGI`:

```
CFG=DD: CONFIGI
```

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

For more information see [Configuring the RPC Server](#).

## IBM LE Runtime Options

Depending on the feature the IMS 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
SSL/TLS connections	POSIX(ON)	If not specified, TCP or NET connections are supported.
Call RPC server programs with AMODE 24 as well	ALL31(OFF),STACK(,,BELOW)	If not specified, AMODE 31 is supported.



**Note:** <sup>(1)</sup> Set internally by the IMS RPC Server 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 EXX960.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](#)
- 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 not case-sensitive.
- 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, i.e. the commands/parameters `broker=localhost` and `brok=localhost` are equivalents.

Parameter	Default	Values	Req/Opt
<code>brokerid</code>	<code>localhost</code>	Broker ID used by the server. See <i>Using the Broker ID in Applications</i> in the RPC Programming documentation.  Example: <code>brokerid=myhost.com:1971</code>	R
<code>ceeoptions</code>		Allows you to change IBM's LE runtime options. This parameter is deprecated. See <i>IBM LE Runtime Options</i> for how to set the LE runtime options.	O
<code>class</code>	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 <i>Service-specific Attributes (DEFAULTS=SERVICE)</i> under <i>Broker Attributes</i> in the platform-independent administration documentation). Case-sensitive, up to 32 characters. Corresponds to CLASS.  Example: <code>class=MyRPC</code>	R
<code>codepage</code>	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 <i>What is the Best Internationalization Approach to use?</i> under	



Parameter	Default	Values	Req/ Opt
		<p><i>Internationalization with EntireX</i> for information on which internationalization approach requires a codepage (locale string).</p> <p>By default, no codepage is transferred to the broker. For the most popular internationalization approach, <i>ICU Conversion</i> under <i>Introduction to Internationalization</i>, the correct codepage (locale string) must be provided. This means it must:</p> <ul style="list-style-type: none"> <li>■ follow the rules described under <i>Locale String Mapping</i> in the internationalization documentation</li> <li>■ be a codepage supported by the broker</li> <li>■ be the codepage used in your environment for file and terminal IO, otherwise unpredictable results may occur.</li> </ul> <p>Example: codepage=ibm-273</p>	
<u>compresslevel</u>	N	<p>Enforce compression when data is transferred between broker and server. See <i>Data Compression in EntireX Broker</i> in the general administration documentation.</p> <p>compresslevel= 0   1   2   3   4   5   6   7   8   9   Y   <u>N</u></p> <p>0-9 0=no compression 9=max. compression</p> <p><u>N</u> No compression. Y Compression level 6.</p> <p>Example: compresslevel=6</p>	O
<u>deployment</u>	NO	<p>Activates the deployment service, see <i>Deployment Service</i>. Required to use the deployment wizard. See <i>Server Mapping Deployment Wizard</i> in the COBOL Wrapper documentation.</p> <p>YES Activates the deployment service. The RPC server registers the deployment service in the broker.</p> <p>NO The deployment service is deactivated. The RPC server does not register the deployment service in the broker.</p> <p>Example: deployment=yes</p>	O

Parameter	Default	Values	Req/ Opt						
<u>encryptionlevel</u>	0	<p>Enforce encryption when data is transferred between client and server. Requires EntireX Security. See <code>ENCRYPTION-LEVEL</code> under <i>Broker ACI Fields</i>.</p> <p>0 Encryption is enforced.                      1 Encryption is enforced between server and broker kernel.                      2 Encryption is enforced between server and broker kernel, and also between client and broker.</p> <p>Example:  <code>encryptionlevel=2</code></p>	O						
<u>etblnk</u>	BROKER	<p>Define the broker stub to be used. See <i>Administering Broker Stubs</i> in the z/OS administration documentation for available stubs.</p> <p>Example:  <code>etblnk=broker</code></p>	O						
<u>extractor</u>	NO	<p>The extractor service is a prerequisite for remote extractions. See <a href="#">Extractor Service</a>.</p> <p><code>extractor=YES NO</code></p> <p>Example:  <code>extractor=yes</code></p>	O						
<u>impersonation</u>	NO	<p>Defines if RPC requests are executed under the user ID of the RPC client. Depending on settings, different levels of checks are done prior to RPC server execution. See also <a href="#">Impersonation</a>.</p> <p><code>impersonation= NO   YES   AUTO [ , <u>sameuser</u>   , anyuser ]</code></p> <table border="1"> <tbody> <tr> <td>NO</td> <td>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.</td> </tr> <tr> <td>YES</td> <td>The RPC request runs impersonated under the supplied <i>RPC client user ID</i>. The IMS RPC Server validates the <i>RPC client user ID/password</i> pair against the mainframe security repository.</td> </tr> <tr> <td>AUTO</td> <td>Same as option YES above, except that no password validation is performed, that is, the client is treated as already authenticated. For this setting, make sure the RPC client is correctly authenticated; use either</td> </tr> </tbody> </table>	NO	The RPC request is executed anonymously, which means the user ID of the RPC client is not used. RPC requests are executed under the user ID of the RPC server.	YES	The RPC request runs impersonated under the supplied <i>RPC client user ID</i> . The IMS RPC Server validates the <i>RPC client user ID/password</i> pair against the mainframe security repository.	AUTO	Same as option YES above, except that no password validation is performed, that is, the client is treated as already authenticated. For this setting, make sure the RPC client is correctly authenticated; use either	O
NO	The RPC request is executed anonymously, which means the user ID of the RPC client is not used. RPC requests are executed under the user ID of the RPC server.								
YES	The RPC request runs impersonated under the supplied <i>RPC client user ID</i> . The IMS RPC Server validates the <i>RPC client user ID/password</i> pair against the mainframe security repository.								
AUTO	Same as option YES above, except that no password validation is performed, that is, the client is treated as already authenticated. For this setting, make sure the RPC client is correctly authenticated; use either								

Parameter	Default	Values	Req/ Opt				
		<ul style="list-style-type: none"> <li>■ a secure broker (validation must be against the correct mainframe security repository where the user IDs are defined) and option <code>sameuser</code></li> <li>or</li> <li>■ your own security implementation (option <code>anyuser</code> is supported for compatibility reasons if you need different broker and server user IDs - the customer-written security implementation must validate the RPC client using the <i>RPC client user ID</i>)</li> </ul> <table border="1" data-bbox="857 709 1422 1073"> <tr> <td data-bbox="857 709 1122 926"><code>sameuser</code></td> <td data-bbox="1122 709 1422 926">The IMS RPC Server checks whether the <i>broker client user ID</i> matches the <i>RPC client user ID</i>. This is the default if <code>AUTO</code> is used.</td> </tr> <tr> <td data-bbox="857 926 1122 1073"><code>anyuser</code></td> <td data-bbox="1122 926 1422 1073">The <i>RPC client user ID</i> is used for impersonation. The <i>broker client user ID</i> is ignored.</td> </tr> </table> <p><b>Note:</b></p> <ol style="list-style-type: none"> <li>1. EntireX supports two user ID/password pairs: a <i>broker client user ID/password</i> pair and an (optional) <i>RPC user ID/password</i> pair sent from RPC clients to the RPC server.</li> <li>2. With EntireX Security, the <i>broker client user ID/password</i> pair is checked. The <i>RPC user ID/password</i> pair is designed to be checked by the target RPC server. Thus it is possible to use different user IDs in the broker and target RPC server.</li> <li>3. RPC clients send the (optional) <i>RPC user ID/password</i> pair in the same way as specifying the Natural user ID/password pair for a Natural RPC Server. See for example <i>Using Natural Security</i> in the respective section of the documentation.</li> <li>4. If the RPC client does not specify the optional <i>RPC user ID/password</i> pair, the <i>broker client user ID</i> is inherited to the <i>RPC user ID</i> and thus used for impersonation by the IMS RPC Server.</li> </ol> <p>Example: <code>impersonation=auto,anyuser</code></p>	<code>sameuser</code>	The IMS RPC Server checks whether the <i>broker client user ID</i> matches the <i>RPC client user ID</i> . This is the default if <code>AUTO</code> is used.	<code>anyuser</code>	The <i>RPC client user ID</i> is used for impersonation. The <i>broker client user ID</i> is ignored.	
<code>sameuser</code>	The IMS RPC Server checks whether the <i>broker client user ID</i> matches the <i>RPC client user ID</i> . This is the default if <code>AUTO</code> is used.						
<code>anyuser</code>	The <i>RPC client user ID</i> is used for impersonation. The <i>broker client user ID</i> is ignored.						

Parameter	Default	Values	Req/ Opt
		<p>Using impersonation requires additional installation steps. See <i>Using z/OS Privileged Services</i> in the z/OS installation documentation.</p>	
<u>library</u>	no default	<p><code>library = search-logic [- library]</code> where <code>search-logic</code> is one of <code>FIX(dllname)   PREFIX(prefix)   PREFIX()</code></p> <p>This parameter applies to programming language C only. Do not set if other programming languages for RPC server are used.</p> <p><code>FIX(dllname)</code> 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.</p> <p><code>PREFIX(prefix)</code> The IDL library name coming from the RPC client is used to form the DLL name. As <code>prefix</code> you can define any character. If an RPC client sends, for example, "SYSTEM" as the IDL library name and "D" is defined as <code>prefix</code>, the DLL name derived is "DSYSTEM". This configuration restricts the IDL library names to max. 7 characters.</p> <p><code>PREFIX()</code> 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.</p> <p>Example PREFIX configuration (this configuration matches the standard names produced by the C Wrapper):  <code>library=PREFIX(D)-PREFIX()</code></p> <p>Example FIX configuration:  <code>library=FIX(MYSTUBS)-FIX(MYRPCS)</code></p>	O
<u>logon</u>	YES	<p>Execute broker functions LOGON/LOGOFF in worker threads. Must match the setting of the broker attribute AUTOLOGON. Reliable RPC requires logon set to YES. See <i>Reliable RPC</i>.</p> <p>NO No logon/logoff functions are executed.</p> <p><u>YES</u> Logon/logoff functions are executed.</p> <p>Example:  <code>logon=no</code></p>	O

Parameter	Default	Values	Req/Opt										
<u>marshalling</u>	COBOL	<p>The IMS RPC Server can be configured to support either COBOL, PL/I or C. See also <a href="#">Locating and Calling the Target Server</a>.</p> <p>marshalling=(LANGUAGE=COBOL PLI [flavor=ENTERPRISE   MVS]   C)</p> <table border="1"> <tr> <td>COBOL</td> <td>Server supports COBOL. The COBOL servers are called directly without a server interface object. So-called server mapping (SVM) files are used to call the COBOL server correctly if one is available. See <i>Server Mapping Deployment</i>.</td> </tr> <tr> <td>PLI</td> <td>Server supports PL/I Server compiled with IBM's PL/I. In z/OS load modules produced by newer IBM PL/I compilers and linkers, the flavor can be detected automatically, thus <code>flavor</code> can be omitted.                             <table border="1"> <tr> <td>ENTERPRISE</td> <td>Enterprise compiler z/OS. This is the default if PL/I is used. See prerequisites for z/OS.</td> </tr> <tr> <td>MVS</td> <td>Server supports PL/I Server compiled with older IBM compiler PL/I MVS &amp; VM V1R1 and above. See prerequisites for z/OS.</td> </tr> </table> </td> </tr> <tr> <td>C</td> <td>Server supports C. The modules are called using a server interface object built with the <i>C Wrapper</i>.</td> </tr> </table>	COBOL	Server supports COBOL. The COBOL servers are called directly without a server interface object. So-called server mapping (SVM) files are used to call the COBOL server correctly if one is available. See <i>Server Mapping Deployment</i> .	PLI	Server supports PL/I Server compiled with IBM's PL/I. In z/OS load modules produced by newer IBM PL/I compilers and linkers, the flavor can be detected automatically, thus <code>flavor</code> can be omitted. <table border="1"> <tr> <td>ENTERPRISE</td> <td>Enterprise compiler z/OS. This is the default if PL/I is used. See prerequisites for z/OS.</td> </tr> <tr> <td>MVS</td> <td>Server supports PL/I Server compiled with older IBM compiler PL/I MVS &amp; VM V1R1 and above. See prerequisites for z/OS.</td> </tr> </table>	ENTERPRISE	Enterprise compiler z/OS. This is the default if PL/I is used. See prerequisites for z/OS.	MVS	Server supports PL/I Server compiled with older IBM compiler PL/I MVS & VM V1R1 and above. See prerequisites for z/OS.	C	Server supports C. The modules are called using a server interface object built with the <i>C Wrapper</i> .	O
COBOL	Server supports COBOL. The COBOL servers are called directly without a server interface object. So-called server mapping (SVM) files are used to call the COBOL server correctly if one is available. See <i>Server Mapping Deployment</i> .												
PLI	Server supports PL/I Server compiled with IBM's PL/I. In z/OS load modules produced by newer IBM PL/I compilers and linkers, the flavor can be detected automatically, thus <code>flavor</code> can be omitted. <table border="1"> <tr> <td>ENTERPRISE</td> <td>Enterprise compiler z/OS. This is the default if PL/I is used. See prerequisites for z/OS.</td> </tr> <tr> <td>MVS</td> <td>Server supports PL/I Server compiled with older IBM compiler PL/I MVS &amp; VM V1R1 and above. See prerequisites for z/OS.</td> </tr> </table>	ENTERPRISE	Enterprise compiler z/OS. This is the default if PL/I is used. See prerequisites for z/OS.	MVS	Server supports PL/I Server compiled with older IBM compiler PL/I MVS & VM V1R1 and above. See prerequisites for z/OS.								
ENTERPRISE	Enterprise compiler z/OS. This is the default if PL/I is used. See prerequisites for z/OS.												
MVS	Server supports PL/I Server compiled with older IBM compiler PL/I MVS & VM V1R1 and above. See prerequisites for z/OS.												
C	Server supports C. The modules are called using a server interface object built with the <i>C Wrapper</i> .												
<u>password</u>	no default	<p>Password for broker logon. Case-sensitive, up to 32 characters. For more information see broker ACI control block field PASSWORD.</p> <p>Example: password=MyPwd</p>	O										
<u>restartcycles</u>	15	<p>Number of restart attempts if the broker is not available. This can be used to keep the IMS 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:</p> <p>timeout + ETB_TIMEOUT + 60 seconds</p> <p>where <code>timeout</code> is the RPC server parameter (see this table), and</p> <p><code>ETB_TIMEOUT</code> is the environment variable (see <i>Environment Variables in EntireX</i> in the general administration documentation)</p>	O										

Parameter	Default	Values	Req/ Opt
		<p>When the number of cycles is reached and a connection to the broker is not possible, the RPC server stops.</p> <p>Example: restartcycles=30</p>	
<u>return_code</u>	NO	<p>Enable application-specific errors. return_code=(<u>NO</u> YES)</p> <p>NO No tests of COBOL special register RETURN-CODE for application-provided error.</p> <p>YES After execution of the RPC server, tests COBOL special register RETURN_CODE for application provided error. See <a href="#">Aborting RPC Server Customer Code and Returning Error to RPC Client</a>.</p> <p>Example: return_code=yes</p>	O
<u>runoption</u>	no default	<p>This parameter is for special purposes. It provides the IMS RPC Server with additional information. The runoptions are normally set to meet the platform's requirements. Set this parameter only if a support representative provides you with an option and asks you to do so. The parameter can be defined multiple times.</p> <p>Example: runoption=&lt;option&gt; runoption=&lt;option&gt;</p>	O
<u>servername</u>	SRV1	<p>Server name part of the server address used by the server. The server address must be defined as a service in the broker attribute file. See <i>Service-specific Attributes (DEFAULTS=SERVICE)</i> under <i>Broker Attributes</i> in the platform-independent administration documentation. Case-sensitive, up to 32 characters. Corresponds to SERVER of the broker attribute file.</p> <p>Example: servername=mySrv</p>	R
<u>service</u>	CALLNAT	<p>Service part of the server address used by the server. The server address must be defined as a service in the broker attribute file. See <i>Service-specific Attributes (DEFAULTS=SERVICE)</i> under <i>Broker Attributes</i> in the platform-independent administration documentation. Case-sensitive, up to 32 characters. Corresponds to SERVICE attribute of the broker attribute file.</p> <p>Example: service=MYSERVICE</p>	R

Parameter	Default	Values	Req/Opt
<u>smhport</u>	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.  Example: smhport=3001	O
<u>ssl_file</u>	no default	Set the SSL parameters. See <a href="#">Using SSL or TLS with the RPC Server</a> for examples and more information.	O
<u>stublic</u>	no default	For PL/I server if its interface has <a href="#">IMS-specific PCB Pointers</a> to access IMS databases. Provide the DD name of the library where the PL/I server interface object(s) are located. Not required for COBOL or if the interface of your PL/I server does not have IMS-specific PCB pointers.  stublic=ddname  Example: stublic=PLISTUBS  For the example above, define the DD name PLISTUBS in the started task JCL of the IMS RPC Server (see <a href="#">Started Task JCL</a> ) as  <pre>//PLISTUBS DD DISP=SHR,DSN=&lt;plistubs&gt;</pre>	O
<u>svm</u>	ERXSVM	Usage and location of SVM files. If no svm parameter is given, the RPC server tries to open the SVM container using DD name ERXSVM. If this DD name is not available, no server mappings are used. For more information see <a href="#">SVM Files</a> .  svm = no   ddname  no No SVM files are used.  ddname DD name of the SVM file container in the JCL of the IMS RPC Server.  Example: svm=MYSVM	O
<u>timeout</u>	60	Timeout in seconds, used by the server to wait for broker requests. See broker ACI control block field WAIT for more information. Also influences <a href="#">restartcycles</a> .  Example: timeout=300	O

Parameter	Default	Values	Req/Opt
<code>tracedestination</code>	DD:ERXTRACE	<p>The name of the destination file for trace output.</p> <p><code>tracedestination=DD: ddname</code>, where <i>ddname</i> is the name of the trace file.</p> <p>Example:  <code>tracedestination=DD:MYTRACE</code></p> <p>The DD name MYSVM must be defined in the started task of the IMS RPC Server (see <a href="#">Started Task JCL</a>):</p> <pre>//MYTRACE DD DISP=SHR,DSN=&lt;rpctrace-file&gt;</pre>	O
<code>tracelevel</code>	None	<p>Trace level for the server. See also <a href="#">Activating Tracing for the RPC Server</a>.</p> <pre>tracelevel = <u>None</u>   Standard   Advanced   Support</pre> <p>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.</p> <p>Example:  <code>tracelevel=standard</code></p>	O
<code>userid</code>	ERX-SRV	<p>Used to identify the server to the broker. See broker ACI control block field USER-ID. Case-sensitive, up to 32 characters.</p> <p>Example:  <code>userid=MyUid</code></p>	R

## Locating and Calling the Target Server

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

- COBOL
- C



- [Assembler \(IBM 370\)](#)

## COBOL

The approach used to derive the z/OS module 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 z/OS module name of the RPC server is derived from the SVM entry. In this case the IDL program name can be different to the z/OS module 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 z/OS module name of the RPC server (the IDL library name is ignored).

### ▶ To use the IMS RPC Server with COBOL

- 1 Make sure that all z/OS modules called as RPC servers
  - are compiled with IBM's Language Environment (see [z/OS V1R4.0 Lang Env Prog Guide](#) for more information)
  - use COBOL calling conventions
  - can be called dynamically ("fetched") from any Language Environment program
  - are accessible through the IMS RPC Server started task JCL STEPLIB concatenation. See [Started Task JCL](#).
- 2 Configure the parameter `marshalling` for COBOL, for example:
 

```
marshalling=COBOL
```
- 3 Configure the 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](#).

## C

The approaches needed to derive the dynamic-link libraries (DLLs) 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* under *Software AG IDL File* in the IDL Editor documentation).

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 IMS RPC Server with C

- 1 Make sure all dynamic-link libraries (DLLs) called as RPC servers and client interface objects are accessible through the IMS RPC Server started task JCL STEPLIB concatenation. See *Started Task JCL* under *Administering the Batch RPC Server*.
- 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, UNIX, Windows, BS2000/OSD, IBM i)*.

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 IMS RPC Server with Assembler

- Make sure all z/OS modules called as RPC Servers
  - are accessible through the IMS RPC Server started task JCL STEPLIB concatenation. See *Started Task JCL* under *Administering the Batch RPC Server*.
  - 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 or TLS with the RPC Server

The IMS RPC Server supports certificates stored in RACF as keyrings. There are two ways of specifying the RACF keyring and other SSL or TLS parameters, depending on the complexity of the parameters:

- as part of the Broker ID for short parameters, the simplest way
- using the SSL file, a text file containing more complex parameters.

As an alternative, you can use for this purpose 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.

This section covers the following topics:

- [Specifying the SSL or TLS Parameters as Part of the Broker ID](#)
- [Specifying the SSL or TLS Parameters in a Separate File](#)
- [Using IBM's Application Transparent Transport Layer Security \(AT-TLS\)](#)

For more information, see *SSL or TLS and Certificates with EntireX*.

### Specifying the SSL or TLS Parameters as Part of the Broker ID

#### ► To specify the SSL or TLS parameters as part of the Broker ID

- 1 In the *Started Task JCL* set the LE runtime option `POSI(ON)`, see [IBM LE Runtime Options](#).
- 2 Add the RACF keyring `<user-id>/<ring-name>` and other SSL or TLS parameters to the server parameter `brokerid` in the *Configuration File*. SSL or TLS parameters are separated by ampersand (&).

Example with *Transport-method-style Broker ID*:

```
ETB024:1609:SSL?TRUST_STORE=<user-id>/<ring-name>&VERIFY_SERVER=N
```

Example with *URL-style Broker ID*:

```
ssl://localhost:2010?TRUST_STORE=<user-id>/<ring-name>&VERIFY_SERVER=N
```

- 3 Make sure the target the IMS 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

- *Setting 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

## Specifying the SSL or TLS Parameters in a Separate File

### ▶ To specify the SSL or TLS parameters in the SSL file

- 1 In the *Started Task JCL* set the LE runtime option `POSI(ON)`, see [IBM LE Runtime Options](#).
- 2 Define a so-called SSL file in text format (for example as a PDS member) with the RACF keyring `<user-id>/<ring-name>` and other SSL or TLS parameters.

Example:

```
TRUST_STORE=<user-id>/<ring-name>
VERIFY_SERVER=N
```



**Note:** Each line in the SSL file must be terminated with hexadecimal zero.

- 3 In the *Configuration File*, define a DDNAME to be used in the *Started Task JCL* to enable the IMS RPC Server to access the SSL file defined in Step 2 above.

Example:

```
...
SS_FILE=DD:MYSSL
...
```

- 4 Add a DD statement to the *Started Task JCL* using the DDNAME defined in Step 3 above to point to the SSL file defined in Step 2 above.

Example:

```
//...
//MYSSL DD DISP=SHR,DSN=<high-level-qualifier>.MYPDS(SSLFILE)
//...
```

- 5 In the *Configuration File* define the server parameter `brokerid` for SSL or TLS connections.

Example with *Transport-method-style Broker ID*:

```
ETB024:1609:SSL
```

Example with *URL-style Broker ID*:

```
ssl://localhost:2010
```

- 6 Make sure the target the IMS 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
  - *Setting 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

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

### ▶ To set up SSL or TLS with AT-TLS

- 1 Set up the IMS RPC Server for a TCP/IP connection.
- 2 Configure the rules for the AT-TLS policy agent the IMS RPC Server matches, for example by using the job name and remote port number the IMS 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 IMS 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
  - *Setting 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

## Starting the RPC Server

---

### ▶ To start the IMS RPC Server

- 1 Modify the member `EXPSRVI` (see EntireX job library `EXX960.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 IMS RPC Server

- Use the operator command `STOP`. Examples:

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

Or:

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

Or:

Use the System Management Hub; this method ensures that the deregistration from the Broker is correct. See *Broker Administration using System Management Hub* in the UNIX and Windows administration documentation.

## Activating Tracing for the RPC Server

### ▶ To switch on tracing for IMS RPC Server

- 1 Set the parameters `tracelevel` and `tracedestination`.
- 2 Dynamically 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 evaluate the return codes, see *Component Return Codes in EntireX* under *Error Messages and Codes*.





# 3

## Extractor Service

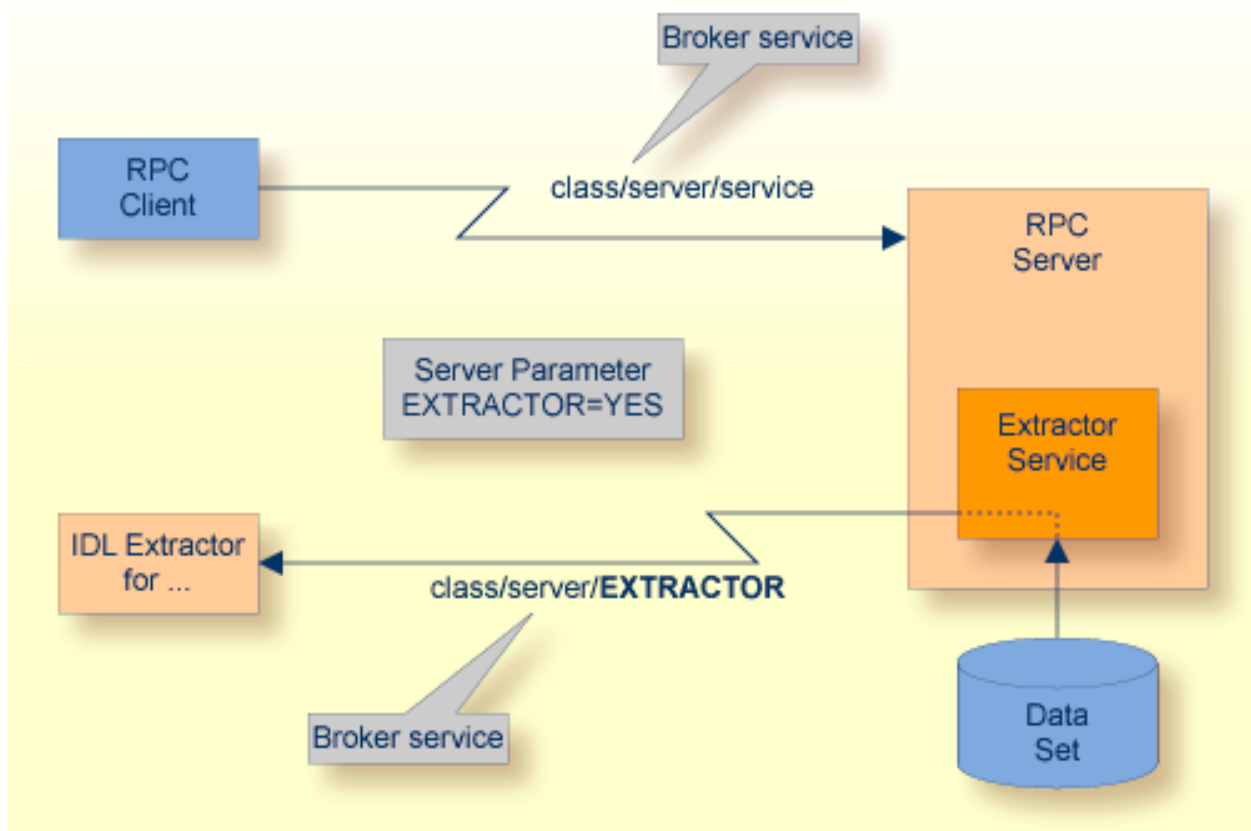
---

- Introduction ..... 30
- Scope ..... 31
- CA Librarian Support ..... 31
- Enabling the Extractor Service ..... 32
- Disabling the Extractor Service ..... 32
- Restrictions ..... 33

## Introduction

The extractor service

- provides access to PDS data sets and CA Librarian DA master files defined within the z/OS catalog
- provides access to security-protected data sets (protected e.g. by RACF, CA ACF2, CA Top Secret)
- is a built-in service of the RPC server, which can be enabled/disabled by RPC server configuration settings
- depending on the platform where the broker is running, 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 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* in the IDL Extractor for COBOL documentation.
- **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. Please note "EXTRACTOR" is a service name reserved by Software AG. See `SERVICE` under *Broker Attributes* in the platform-independent administration documentation.

---

## 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 IMS RPC Server 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* 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.
- 4 Optional. Use the `impersonation` feature of the IMS RPC Server 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 IMS RPC Server parameter `extractor=no`. See `extractor` under *Configuring the RPC Server*. The IMS RPC Server 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).



# 4 Deployment Service

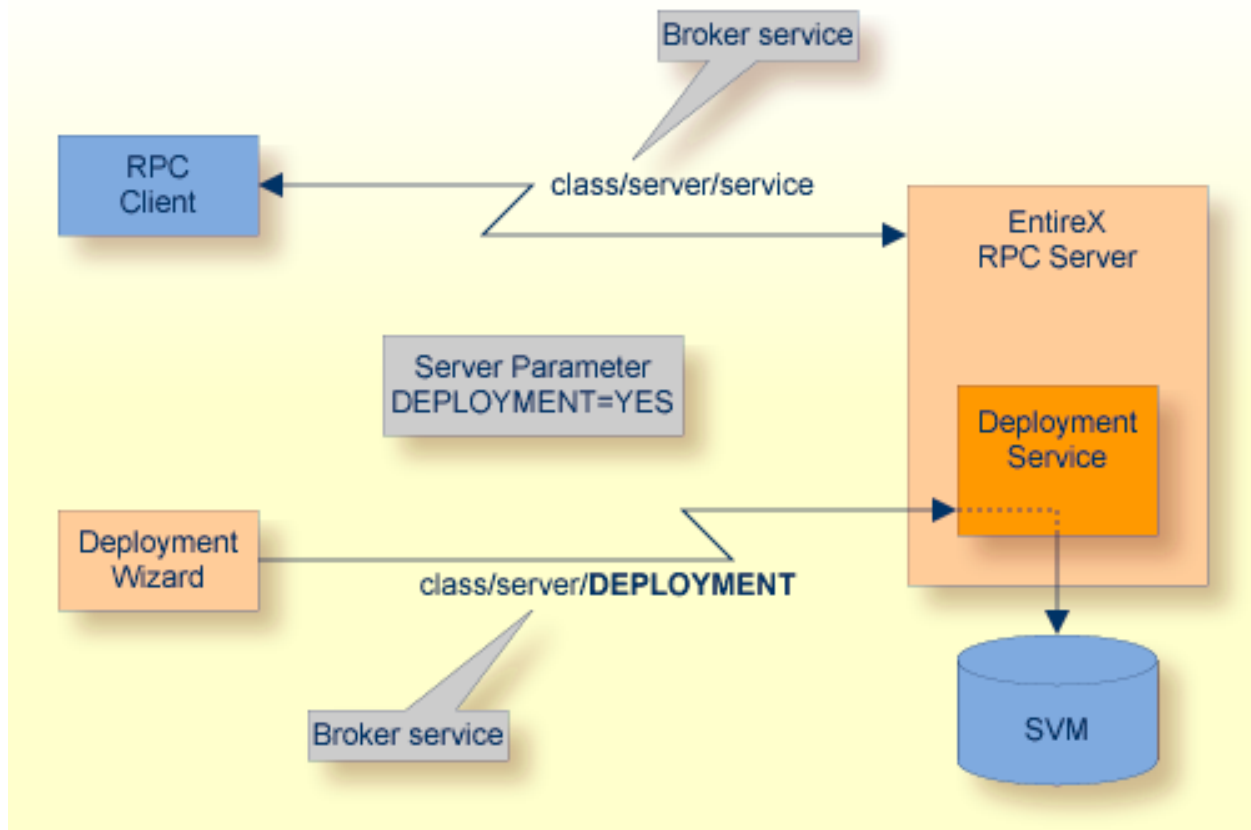
---

- Introduction ..... 36
- Scope ..... 37
- Enabling the Deployment Service ..... 37
- Disabling the Deployment Service ..... 38

## 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 an IMS RPC Server, the server mapping file VSAM (container) must be installed and configured. See *Install the SVM File for a Batch RPC Server (Optional)* under *Installing the EntireX RPC Servers under z/OS* in the z/OS installation documentation 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* in the EntireX Security documentation.

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

## Disabling the Deployment Service

---

### ▶ To disable the deployment service

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

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

# 5 Handling SVM Files

---

- SVM Files in the EntireX Workbench ..... 40
- SVM Files in the RPC Server ..... 40
- Source Control of SVM Files ..... 41
- Change Management of SVM Files ..... 41
- Compare SVM Files ..... 41
- List Deployed SVM Files ..... 41
- Check if an SVM File Revision has been Deployed ..... 42
- Access Control: Secure SVM File Deployment ..... 42
- Ensure that Deployed SVM Files are not Overwritten ..... 42
- When is an SVM File Required? ..... 43
- Is There a Way to Smoothly Introduce SVM Files? ..... 45

A server mapping file (SVM) enables the RPC server to correctly support special COBOL syntax such as `REDEFINES`, `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 the 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 configuration 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 configuration 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

---

Under 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 platforms `z/OS`, `UNIX`, `Windows` or `z/VSE`. See [Enabling the Deployment Service](#).

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

## Ensure that Deployed 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 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	<i>CICS with DFHCOMMAREA Calling Convention under Introduction to the IDL Extractor for COBOL   CICS DFHCOMMAREA under COBOL Parameter Selection</i>
CICS Channel Container Calling Convention	all		yes	<i>CICS with Channel Container Calling Convention</i>
CICS with DFHCOMMAREA Large Buffer Interface	all		yes	<i>CICS with DFHCOMMAREA Large Buffer Interface</i>
IMS MPP Message Interface (IMS Connect)	all		yes	<i>IMS MPP Message Interface (IMS Connect)</i>
IMS BMP with Standard Linkage Calling Convention	all		yes	<i>IMS BMP with Standard Linkage Calling Convention</i>
Micro Focus with Standard Linkage Calling Convention	BINARY clause		yes	<i>Micro Focus with Standard Linkage Calling Convention</i>
all	OCCURS DEPENDING ON clause		yes	<i>Tables with Variable Size - DEPENDING ON Clause under COBOL to IDL Mapping in the IDL Extractor for COBOL documentation</i>
all	REDEFINES clause		yes	<i>REDEFINE Clause</i>
all	TRAILING [SEPARATE] clause		yes	<i>SIGN LEADING and TRAILING SEPARATE Clause</i>
all	LEADING [SEPARATE] clause		yes	<i>SIGN LEADING and TRAILING SEPARATE Clause</i>
all	ALIGNED RIGHT attribute		yes	
all	all	Rename of program	yes	<i>The Software AG IDL Tree Pane under Mapping Editor User Interface in the IDL Extractor for COBOL documentation</i>

Interface Type	COBOL Syntax	COBOL Mapping Editor	SVM Required	More Information
all	all	Map to operation	yes	<i>Context Menu</i> under <i>The COBOL Parameters Pane</i>
all	all	Map to constant	yes	<i>Context Menu</i>
all	all	Suppress	yes	<i>Context Menu</i>
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	<i>CICS with DFHCOMMAREA Large Buffer Interface</i> under <i>COBOL Server Interface Types</i>
CICS with Channel Container Calling Convention	all		yes	<i>CICS with Channel Container Calling Convention</i>
IMS BMP with Standard Linkage Calling Convention	all		yes	<i>IMS BMP with Standard Linkage Calling Convention</i>
Micro Focus	I2 or I4		yes	<i>Micro Focus with Standard Linkage Calling Convention</i>   <i>IDL Data Types</i> under <i>Software AG IDL File</i> in the IDL Editor documentation
all	IDL unbounded array		yes	<i>array-definition</i> under <i>Software AG IDL Grammar</i> in the IDL Editor documentation
all	IDL unbounded group		yes	<i>group-parameter-definition</i> under <i>Software AG IDL Grammar</i> 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	<i>Customize Automatically Generated Server Names</i>
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.



# 6 Scenarios and Programmer Information

---

- COBOL Scenarios ..... 48
- PL/I Scenarios ..... 49
- C Scenarios ..... 50
- Assembler Scenarios ..... 50
- Aborting RPC Server Customer Code and Returning Error to RPC Client ..... 51
- Automatic Syncpoint Handling ..... 52

## 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 of the extraction, also an SVM file. If your COBOL server uses PCB pointers, see [IMS-specific PCB Pointers](#).
- 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 IMS BMP* 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*. If your COBOL server uses PCB pointers, see [IMS-specific PCB Pointers](#).
- 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 IMS BMP* 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 *Software AG IDL File* in the IDL Editor documentation 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. 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 IMS BMP* 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. 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 IMS BMP* 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, UNIX, Windows, BS2000/OSD, IBM i)*.
- 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*

## 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 IMS RPC Server 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`) 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. 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*

## Aborting RPC Server Customer Code and Returning Error to RPC Client

### 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 IMS RPC Server 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 1002nnnn.
- 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 IMS RPC Server with `return_code=yes`.

## Automatic Syncpoint Handling

---

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

- After a successful non-conversational request or an end-of-conversation, the server issues an IMS SYNC call.
- 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.