webMethods EntireX

EntireX z/VSE Batch RPC Server

Version 9.7

October 2014

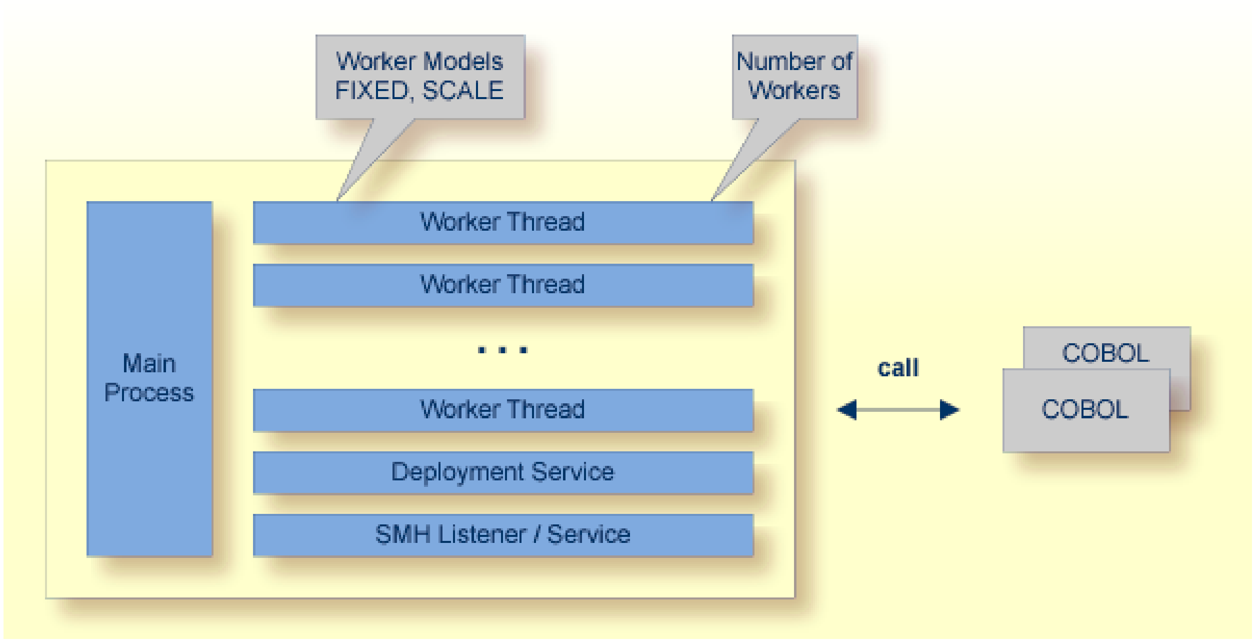
Document ID: EXX-BATCHRPC-VSE-97-20160805

Table of Contents

1 Inside the RPC Server

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

Worker Models



RPC requests are worked off inside the RPC server in worker threads, which are controlled by a main thread. Every RPC request occupies during its processing a worker thread. If you are using RPC conversations, each RPC conversation requires its own thread during the lifetime of the conversation. The Batch RPC Server provides two worker models:

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

  A maximum number (`thru` value of the `workermodel` parameter) of worker threads created can be set to restrict the system load. The minimum number (`from` value of the `workermodel` parameter), allows you to define a certain number of threads - not used by the currently executing RPC request - to wait for new RPC client requests to process. In this way the RPC server is ready to handle many RPC client requests arriving at the same time.

See parameter `workermodel` under ***Configuring the RPC Server***.

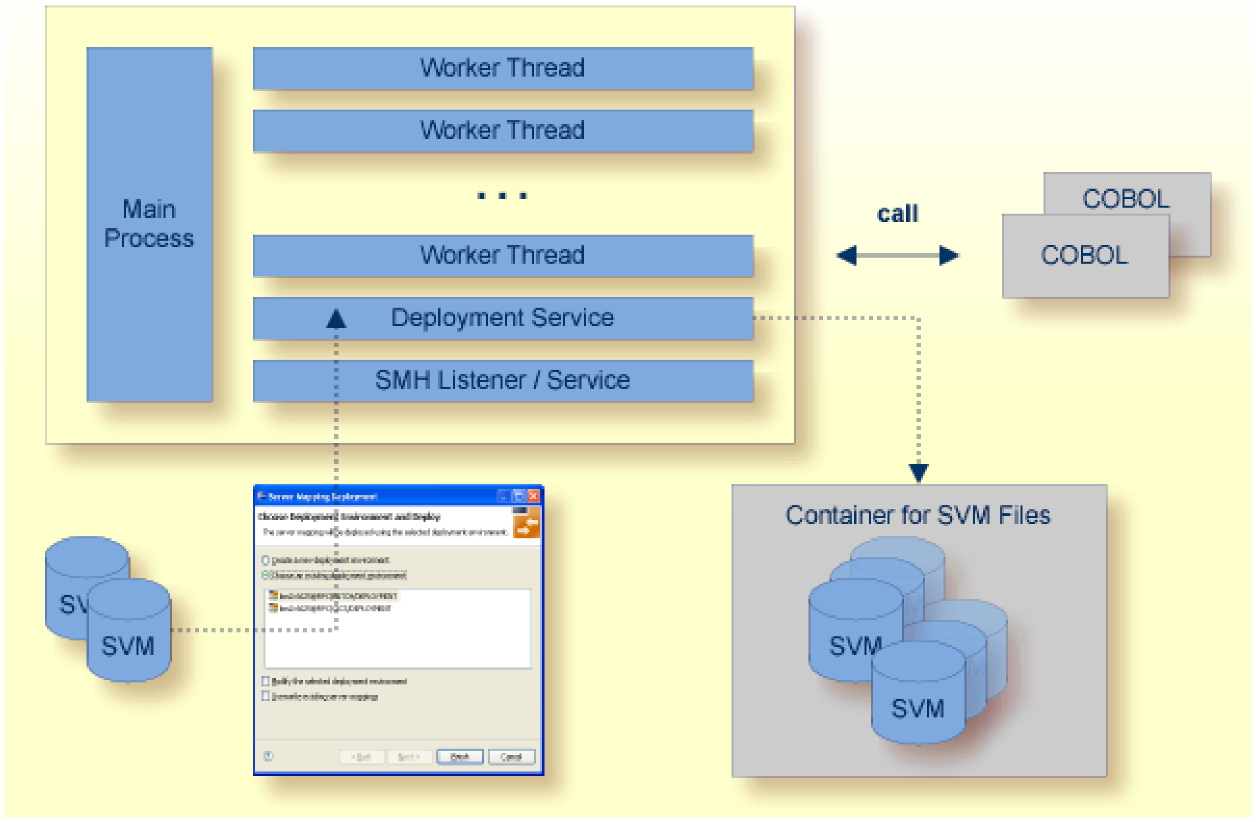
Inbuilt Services

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

- Deployment Service
- SMH Listener Service

Deployment Service

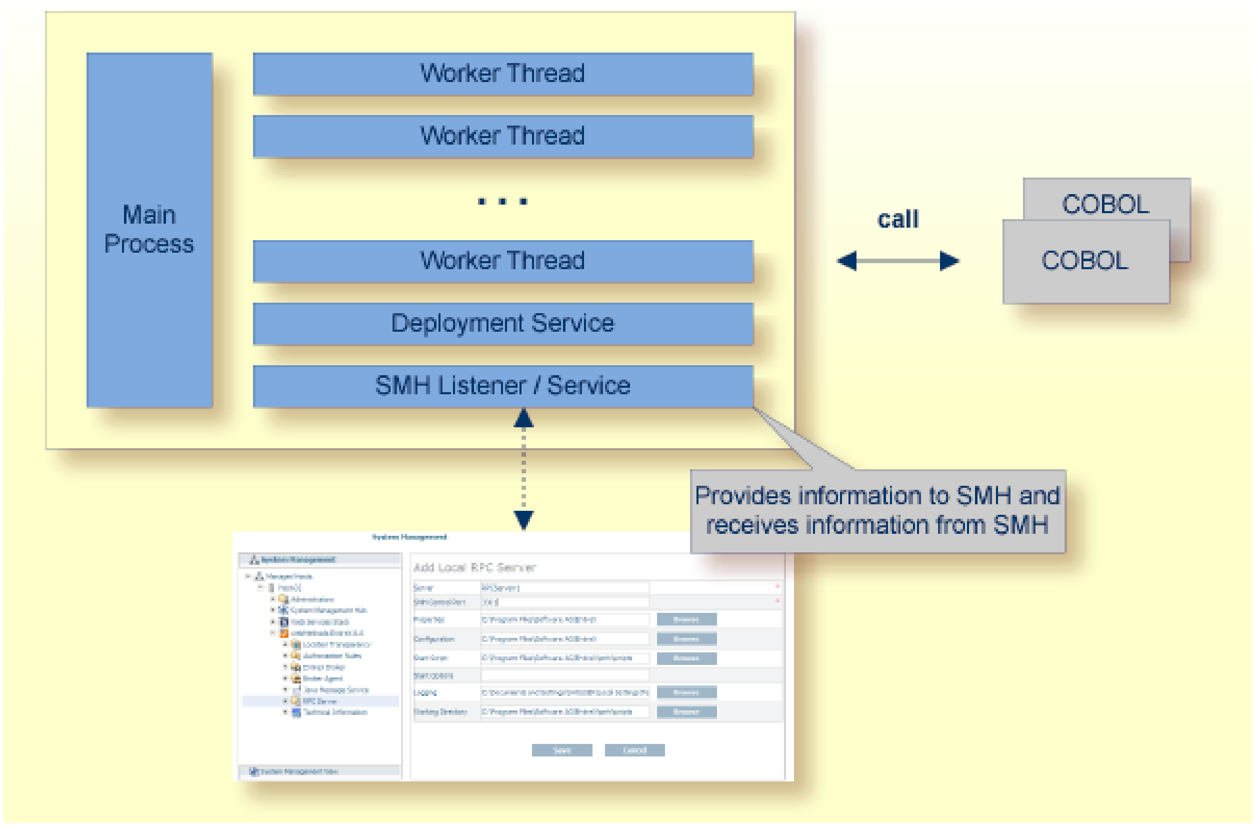
The Deployment Service allows you to deploy server-side mapping files (EntireX Workbench files with extension .svm) interactively using the *Server Mapping Deployment Wizard*. On the RPC server side, the server-side mapping files are stored in a server-side mapping container (VSAM file). See *Server-side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



SMH Listener Service

With the SMH Listener Service you use the System Management Hub to monitor the RPC server. See *Administering the EntireX RPC Servers using System Management Hub* in the UNIX and Windows administration documentation.

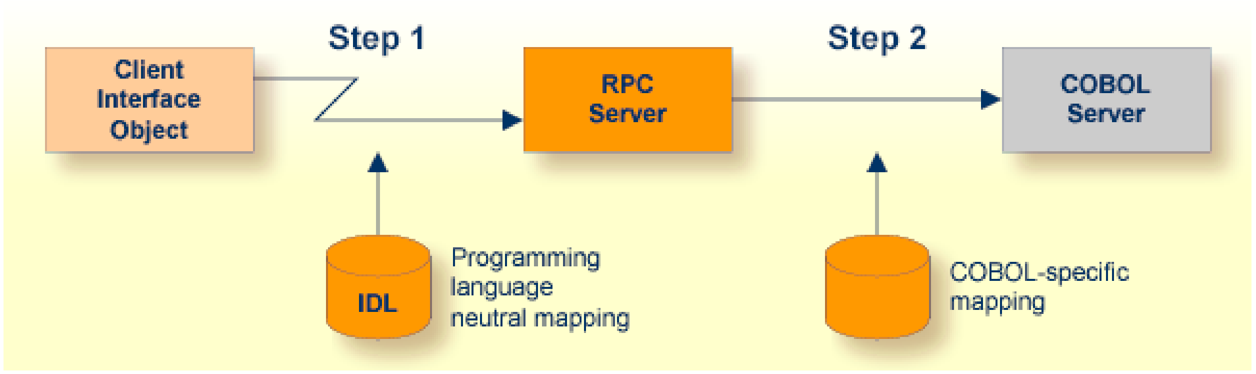
The SMH Service is switched on if the parameter `smhport` is set. See parameter `smhport` under *Configuring the RPC Server*.



Usage of Server Mapping Files

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

Server mapping files contain COBOL-specific mapping information that is not included in the IDL file, but is needed to successfully call the COBOL server program.



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

The server mapping files are retrieved as a result of the *IDL Extractor for COBOL* extraction process and the *COBOL Wrapper* if a COBOL server is generated. See *When is a Server Mapping File Required?*.

There are *server*-side mapping files (*EntireX Workbench* files with extension .svm) and *client*-side mapping files (Workbench files with extension .cvm). See *Server Mapping Files for COBOL* and *How to Set the Type of Server Mapping Files*.

If you are using server-side mapping files, you need to customize the server-side mapping container with parameter `svm`. See ***Configuring the RPC Server***.

Note: Server mapping files are used for COBOL only.

2 Administering the Batch RPC Server

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

Customizing the RPC Server with a Configuration File

The name of the delivered example configuration file is `RPCPARM.CFG` (see sublibrary EXP960). The configuration file contains the configuration for the Batch RPC Server. The following settings are important:

- connection information such as broker ID, server address (class, name, service)
- location and usage of server-side mapping container, see ***Usage of Server Mapping Files***
- scalability parameters
- trace settings
- etc.

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

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

Parameter	Default	Values	Req/ Opt
		`Y` Compression level 6. Example: `compresslevel=6`	
deployment	`NO`	Activates the deployment service, see ***Deployment Service***. Required to use the Server Mapping Deployment Wizard. See *Server Mapping Deployment Wizard* in the EntireX Workbench 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
encryptionlevel	0	Enforce encryption when data is transferred between client and server. Requires EntireX Security. See `ENCRYPTION-LEVEL` under *Broker ACI Fields*. `0` Encryption is enforced. `1` Encryption is enforced between server and broker kernel. `2` Encryption is enforced between server and broker kernel, and also between client and broker. Example: `encryptionlevel=2`	O
etblnk	`BKIMB`	Define the broker stub to be used. See *Administration of Broker Stubs under z/VSE* for available stubs. Example: `ETBL=BKIMB`	O
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

Parameter	Default	Values	Req/ Opt
`marshalling`	`COBOL`	The Batch RPC Server can be configured to support either COBOL or C. See also ***Locating and Calling the Target Server***. `marshalling=(LANGUAGE=COBOL | 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***. `C`: Server supports C. The modules are called using a server interface object built with the *C Wrapper*.	O
`password`	no default	Password for broker logon. Case-sensitive, up to 32 characters. For more information see broker ACI control block field `PASSWORD`. Example: `password=MyPwd`	O
`restartcycles`	15	Number of restart attempts if the broker is not available. This can be used to keep the Batch RPC Server running while the broker is down for a short time. A restart cycle will be repeated at an interval which is calculated as follows: `timeout` + `ETB_TIMEOUT` + 60 seconds where `timeout` is the RPC server parameter (see this table), and `ETB_TIMEOUT` is the environment variable (see *Environment Variables in EntireX* in the general administration documentation) When the number of cycles is reached and a connection to the broker is not possible, the RPC server stops. Example: `restartcycles=30`	O
`runoption`	no default	This parameter is for special purposes. It provides the Batch 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.	O

Parameter	Default	Values	Req/ Opt
		Example: `runoption=<option>` `runoption=<option>`	
`servername`	`SRV1`	Server name part of the server address used by the server. The server address must be defined as a service in the broker attribute file. See *Service-specific Attributes* (`DEFAULTS=SERVICE`) under *Broker Attributes* in the platform-independent administration documentation. 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* (`DEFAULTS=SERVICE`) under *Broker Attributes* in the platform-independent administration documentation. Case-sensitive, up to 32 characters. Corresponds to `SERVICE` attribute of the broker attribute file. Example: `service=MYSERVICE`	R
`smhport`	0	The port where the server listens for commands from the System Management Hub (SMH). If this port is 0 (default), no port is used and management by the SMH is disabled. Example: `smhport=3001`	O
`svm`	`ERXSVM`	Usage and location of server-side mapping files; see ***Server-side Mapping Files in the RPC Server***. If no `svm` parameter is given, the RPC server tries to open the server-side mapping container using DLBL name `ERXSVM`. If this DLBL name is not available, no server-side mapping files are used. If you use server-side mapping files, the server-side mapping container must be installed and configured; see *Step 3: Customize the Batch RPC Server Startup JCL - `RUNRPC.J`* in the z/VSE Installation documentation. There are also client-side mapping files that do not require configuration here; see *Server Mapping Files in the EntireX Workbench* in the EntireX Workbench documentation. `svm = no| *dlblname*` `no` No server-side mapping files are used.	O

Parameter	Default	Values	Req/ Opt

		`dlblname` DLBL name of the server-side mapping container in the startup JCL of the Batch RPC Server.
		
		Example:
		`svm=MYSVM`
		
		For the example above, define the DLBL name `MYSVM` in the startup JCL of the Batch RPC Server as
		
		`// DLBL ↩`
		`MYSVM,'ENTIREX.SVMDEV.KSDS',0,VSAM,CAT=VSESPUC`
		
		See also ***Usage of Server Mapping Files***.	

`timeout`	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 `restartcycles`. Example: `timeout=300`	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 and should only be switched on when requested by Software AG support.
		
		Example:
		`tracelevel=standard`
			O

`userid`	`ERX-SRV`	Used to identify the server to the broker. See broker ACI control block field `USER-ID`. Case-sensitive, up to 32 characters. Example: `userid=MyUid`	R
`workermodel`	`SCALE,1,3,slowshrink`	The Batch RPC Server can be configured to	O

Parameter	Default	Values	Req/ Opt
		■ adjust the number of worker threads to the current number of client requests: `workermodel=(SCALE,*from*,*thru* [,slowshrink | fastshrink])` ■ use a fixed number of worker threads: `workermodel=(FIXED,*number*)`	

`FIXED`	A fixed `number` of worker threads is used by the Batch RPC Server.	
`SCALE`	The number of worker threads is adjusted to the current number of client requests. With the `from` value, the minimum number of active worker threads can be set. The `thru` value restricts the maximum number of worker threads.	
	`slowshrink`	The RPC server stops all worker threads not used in the time specified by the `timeout` parameter, except for the number of workers specified as minimum value. This is the default if `SCALE` is used.
	`fastshrink`	The RPC server stops worker threads immediately as soon as it has finished its conversation, except for the number of workers specified as minimum value.

Example:
`workermodel=(SCALE,2,5)`

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 Batch RPC Server environment. Different mechanisms are used depending on the language:

- COBOL
- C

COBOL

The approach used to derive the z/VSE module name for the RPC server depends on whether server mapping is used or not. See ***Usage of Server Mapping Files*** for an introduction.

1. If the RPC client sends a client-side type of server mapping with the RPC request, this server mapping is used first.
2. If no server mapping is available from step 1 above, and if server-side type of server mapping is used, the IDL library and IDL program names are used to form a key to locate the server mapping in the server-side mapping container. If a server mapping is found, this is then used.
3. If a server mapping is available from step 1 or 2 above, the z/VSE module name of the RPC server is derived from this mapping. In this case the IDL program name can be different to the z/VSE 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*.
4. If no server mapping is used at all, the IDL program name is used as the z/VSE module name of the RPC server (the IDL library name is ignored).

To use the Batch RPC Server with COBOL

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

- are compiled with IBM's Language Environment (see **LE/VSE V1R4 Programming Guide** for more information)
- use COBOL calling conventions
- can be called dynamically ("fetched") from any Language Environment program
- are accessible through the Batch RPC Server JCL `LIBDEF` chain.

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

```
marshalling=COBOL
```

3 Configure the parameter `svm` depending on whether server-side mapping files are used or not. See ***Usage of Server Mapping Files***.

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

C

The approaches needed to derive the names for the Batch RPC Server are more complex for C, for the following reasons:

- the limitation of characters per (physical) member name
- 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.

You need to restrict yourself to short IDL library names.

To use the Batch RPC Server with C

- Configure the parameter `marshalling` for C, for example

```
marshalling=C
```

See *Using the C Wrapper for the Server Side (z/OS, UNIX, Windows, BS2000/OSD, IBM i)*.

Starting the RPC Server

To start the Batch RPC Server

- Run the job `RPCRPC.J`.

Stopping the RPC Server

To stop the Batch RPC Server

- Use the console command `STOP`. For example:

  ```
  task_id STOP
  ```

  Or:

  Use the System Management Hub. This method ensures that the deregistration from the Broker is correct.

Activating Tracing for the RPC Server

To activate tracing for the Batch RPC Server

1. Set the parameter `tracelevel`.
2. Dynamically change the trace level with the operator command

   ```
   port_number TRACELEVEL=tracelevel
   ```

   See the table below for supported trace levels.

   The `TRACELEVEL` command without `tracelevel` option will report the currently active trace.

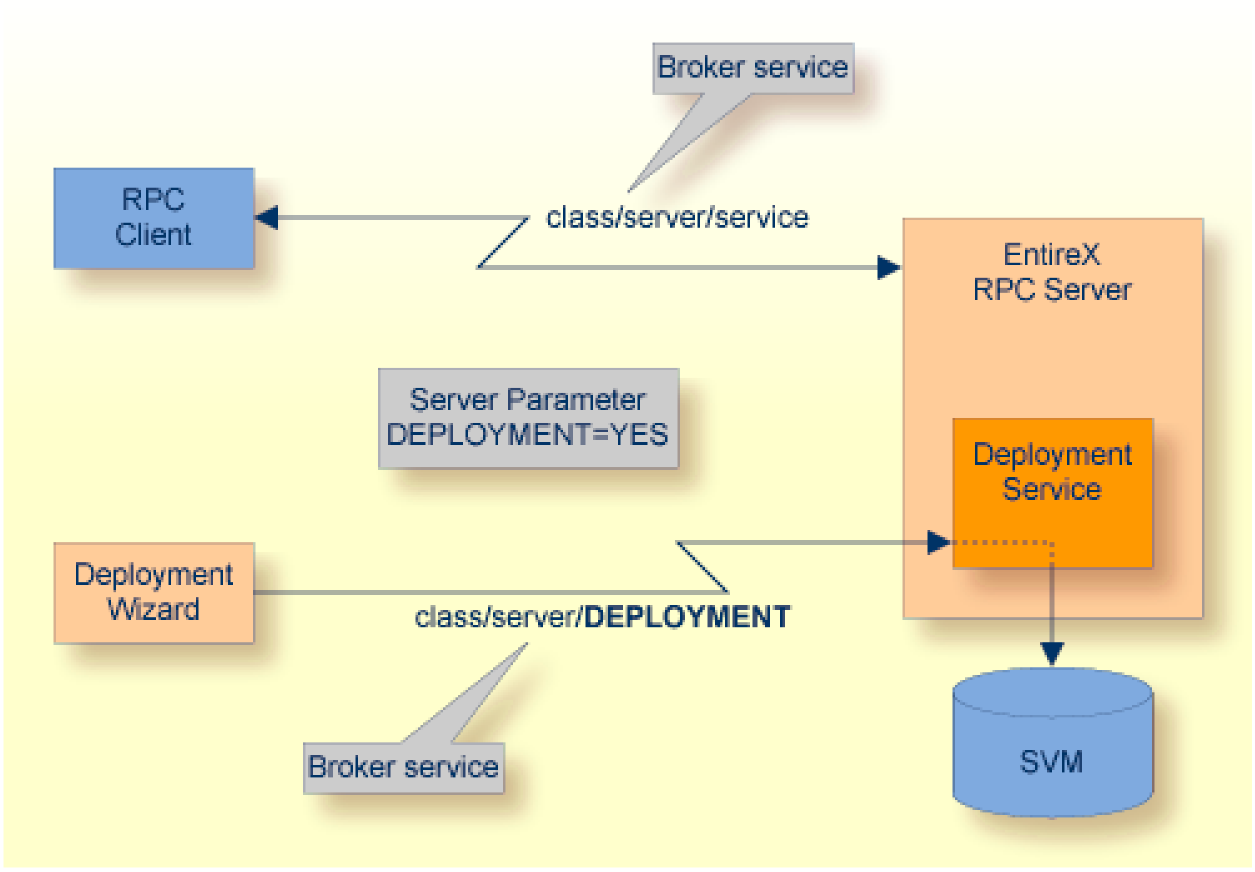
3 Deployment Service

Introduction

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

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

You need to configure the deployment service only when server-side mapping files are used. There are also client-side server mapping files that do not need configuration here; see *Server Mapping Files for COBOL* in the EntireX Workbench documentation.



Scope

The deployment service is used in conjunction with the

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

See also ***Deploying Server-side Mapping Files to the RPC Server***.

The deployment service uses the same class and server names as defined for the EntireX RPC server, and `DEPLOYMENT` as the service name, resulting in *`class`*`/`*`server`*`/DEPLOYMENT` as the broker service. Please note `DEPLOYMENT` is a service name reserved by Software AG. See broker attribute `SERVICE`.

Enabling the Deployment Service

To enable the deployment service

1 For a Batch RPC Server, the server-side mapping container (VSAM file) must be installed and configured. See *Step 1: Define a Server-side Mapping Container -* `VSAMDEF.J` *(Optional)* under *Installing the z/VSE EntireX RPC Servers*.

2 Set 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 Batch RPC Server parameter `deployment=no`. See `deployment` under ***Configuring the RPC Server***.

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

4 Server-side Mapping Files

Server mapping enables the RPC server to correctly support special COBOL syntax such as `REDEFINE`s, `SIGN LEADING` and `OCCURS DEPENDING ON` clauses, `LEVEL-88` fields, etc. If one of these elements is used, the IDL Extractor for COBOL automatically extracts a server mapping file in addition to the IDL file (interface definition language). Also, the COBOL Wrapper may generate a server mapping file for RPC server generation. The server mapping is used at runtime to marshal and unmarshal the RPC data stream. There are client-side mapping files (EntireX Workbench files with extension .cvm) and server-side mapping files (Workbench files with extension .svm). If you have not used server-side mapping, we recommend you use client-side mapping. See *Server Mapping Files for COBOL* in the EntireX Workbench documentation.

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

Server-side Mapping Files in the RPC Server

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

If *one* server requires a server-side mapping file, you need to provide this to the RPC server:

- Development environments: to deploy new server-side mapping files, see ***Deploying Server-side Mapping Files to the RPC Server***.
- Production environments: provide a server-side mapping container (VSAM file) containing all required server-side mapping files to the RPC server. See configuration parameter `svm`.

If *no* server requires server-side mapping, you can execute the RPC server without server mapping files:

- Development environments: you can disable the deployment service. See ***Disabling the Deployment Service***.
- Production environments: there is no need to provide a server-side mapping container (VSAM file) to the RPC server. See configuration parameter `svm`.

Deploying Server-side Mapping Files to the RPC Server

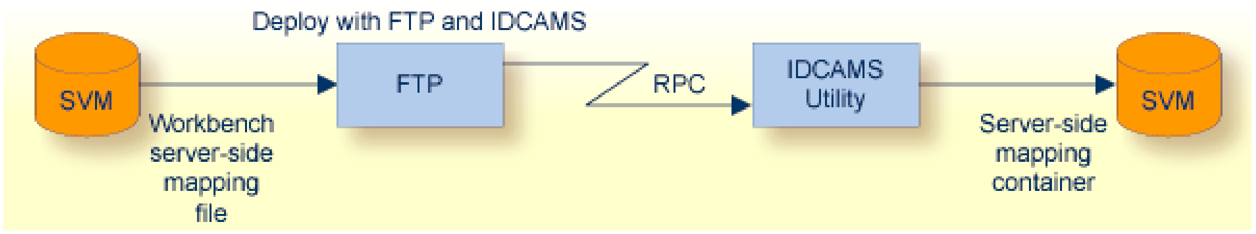
The following approaches are available to deploy a server-side mapping file (EntireX Workbench file with extension .svm; see *Server Mapping Files for COBOL*):

- Server Mapping Deployment Wizard
- FTP and IDCAMS

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

1. Make sure your RPC server is active and that the Deployment Service of the RPC server is properly configured. See *Deployment Service*.
2. From the context-menu of your IDL file, choose **COBOL > Deploy/Synchronize Server Mapping** and call the Server Mapping Deployment Wizard. See *Server Mapping Deployment Wizard* in the EntireX Workbench documentation.

To deploy a server-side mapping file using FTP and IDCAMS


Deploy with FTP and IDCAMS
SVM
Workbench server-side mapping file
FTP
RPC
IDCAMS Utility
Server-side mapping container
SVM

1. Make sure the server-side mapping container (VSAM file) is installed. See *Step 1: Define a Server-side Mapping Container - `VSAMDEF.J` (Optional)* under *Installing the z/VSE EntireX RPC Servers*.
2. Allocate a target sequential file on your mainframe.
3. Allow write access to the VSAM file mentioned above and usage of IDCAMS tools.
4. Transfer the server-side mapping file to the target host, using FTP. You have to switch to text mode and the codepage of the FTP service must be the same as the codepage (locale string) of the RPC server used.
5. Install the server mapping contained in the server-side mapping file into the server-side mapping container (VSAM file) with an appropriate IDCAMS job.

   Note: If you omit the keyword `REPLACE` or define `NOREPLACE` in the `SYSIN` data stream of IDCAMS instead, existing server mapping information is not overwritten. This protects server-side mapping records from being overwritten by duplicates.

Undeploying Server-side Mapping Files to the RPC Server

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

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

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

2 Make sure your IDL file is within an EntireX Workbench directory (folder) without the related server-side mapping file (.svm).

3 From the context-menu of your IDL file, choose **COBOL > Deploy/Synchronize Server Mapping** and call the Server Mapping Deployment Wizard. See *Server Mapping Deployment Wizard* in the EntireX Workbench documentation. Because there is no related server-side mapping file in the Workbench, all server mapping information related to the IDL file in the RPC server will be removed.

Change Management of Server-side Mapping Files

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

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

List Deployed Server-side Mapping Files

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

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

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

Server-side mapping records in the server-side mapping container correspond to lines of EntireX Workbench files with extension .svm. See *Server Mapping Files for COBOL* in the EntireX Workbench documentation. The records contain a creation timestamp at offset 276 (decimal) in the format `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 Server Mapping File Deployment

For deployment with the *Server Mapping Deployment Wizard*, use EntireX Security if the broker is running on platforms z/OS, UNIX , Windows or z/VSE. See ***Enabling the Deployment Service***.

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

Ensure that Deployed Server-side Mapping Files are not Overwritten

For IDCAMS, use the `NOREPLACE` option to disallow overwriting of duplicate server-side mapping records in the server-side mapping container (VSAM file); see ***Server-side Mapping Files in the RPC Server***. See also ***Deploying Server-side Mapping Files to the RPC Server***.

Is There a Way to Smoothly Introduce Server-side Mapping Files?

All EntireX RPC servers can be executed without server-side mapping files. See ***Server-side Mapping Files in the RPC Server***. There is no need to install the server-side mapping container if the following conditions are met:

- You do not use features that require server mapping; see *When is a Server Mapping File Required?*
- Server-side type of COBOL mapping is switched on in the EntireX Workbench. If you have not used server-side mapping, we recommend you use client-side mapping. See *Server Mapping Files for COBOL*.

You can also call COBOL servers generated or extracted with previous versions of EntireX mixed with a COBOL server that requires server-side mapping. All EntireX RPC servers are backward compatible.

5 Scenarios

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?* in the EntireX Workbench documentation.
2. Build an EntireX RPC client using any EntireX wrapper. For a quick test you can:
   - use the IDL Tester; see *EntireX IDL Tester* in the EntireX Workbench documentation
   - generate an XML mapping file (XMM) and use the XML Tester for verification; see *EntireX XML Tester* in the XML/SOAP Wrapper documentation

See *Client and Server Examples for z/VSE Batch* in the COBOL Wrapper documentation for COBOL RPC Server examples.

Scenario II: Writing a New COBOL Server

To write a new COBOL server

1. Use the *COBOL Wrapper* to generate a COBOL server skeleton and, depending on the complexity, also a server mapping file. See *When is a Server Mapping File Required?* in the EntireX Workbench documentation. Write your COBOL server and proceed as described under *Using the COBOL Wrapper for the Server Side*.
2. Build an EntireX RPC client using any EntireX wrapper. For a quick test you can:
   - use the IDL Tester; see *EntireX IDL Tester* in the EntireX Workbench documentation
   - generate an XML mapping file (XMM) and use the XML Tester for verification; see *EntireX XML Tester* in the XML/SOAP Wrapper documentation

See *Client and Server Examples for z/VSE Batch* in the COBOL Wrapper documentation for COBOL RPC Server examples.