webMethods EntireX

EntireX RPC Server for BS2000

Version 10.5

October 2019

WEBMETHODS

This document applies to webMethods EntireX Version 10.5 and all subsequent releases.

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

Document ID: EXX-BS2RPC-105-20220422

Table of Contents

1 About this Documentation

Document Conventions

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

Online Information and Support

Product Documentation

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

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

Product Training

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

Tech Community

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

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

Product Support

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

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

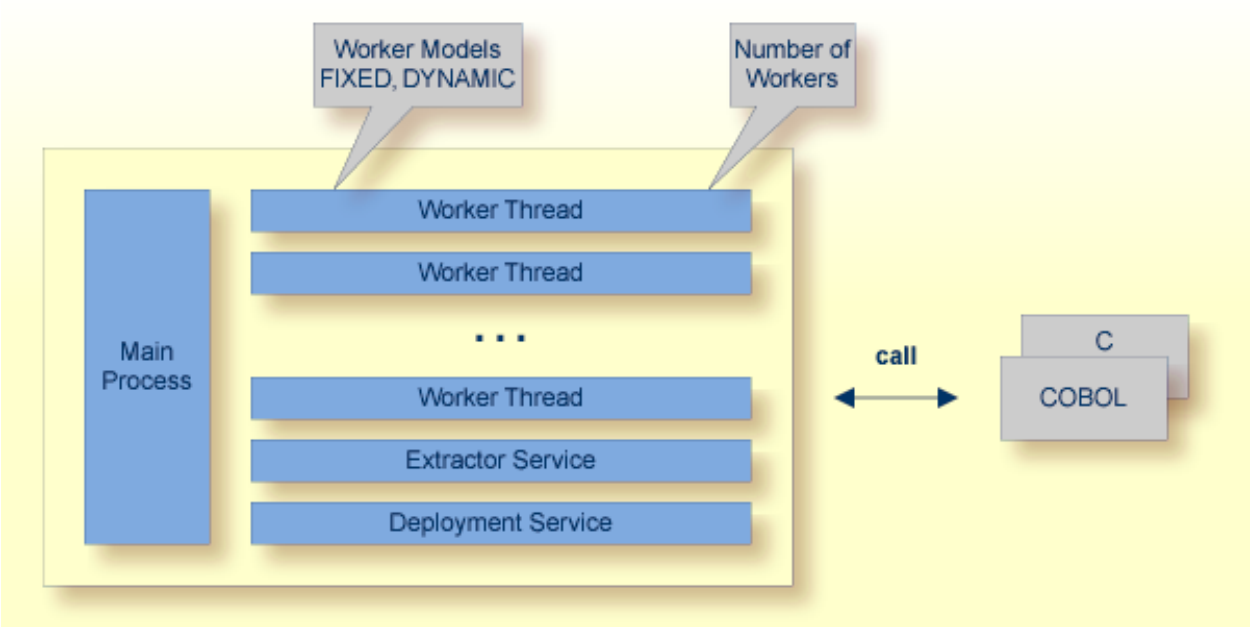
Data Protection

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

2 Introduction to the RPC Server for BS2000

The EntireX RPC Server for BS2000 allows standard RPC clients to communicate with RPC servers on the operating system BS2000. It supports the programming languages COBOL and C.

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 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.
- `DYNAMIC`
  The *dynamic* model creates worker threads depending on the incoming load of RPC requests.

For configuration and technical details, see parameter `workermodel` under *Administering the RPC Server for BS2000*.

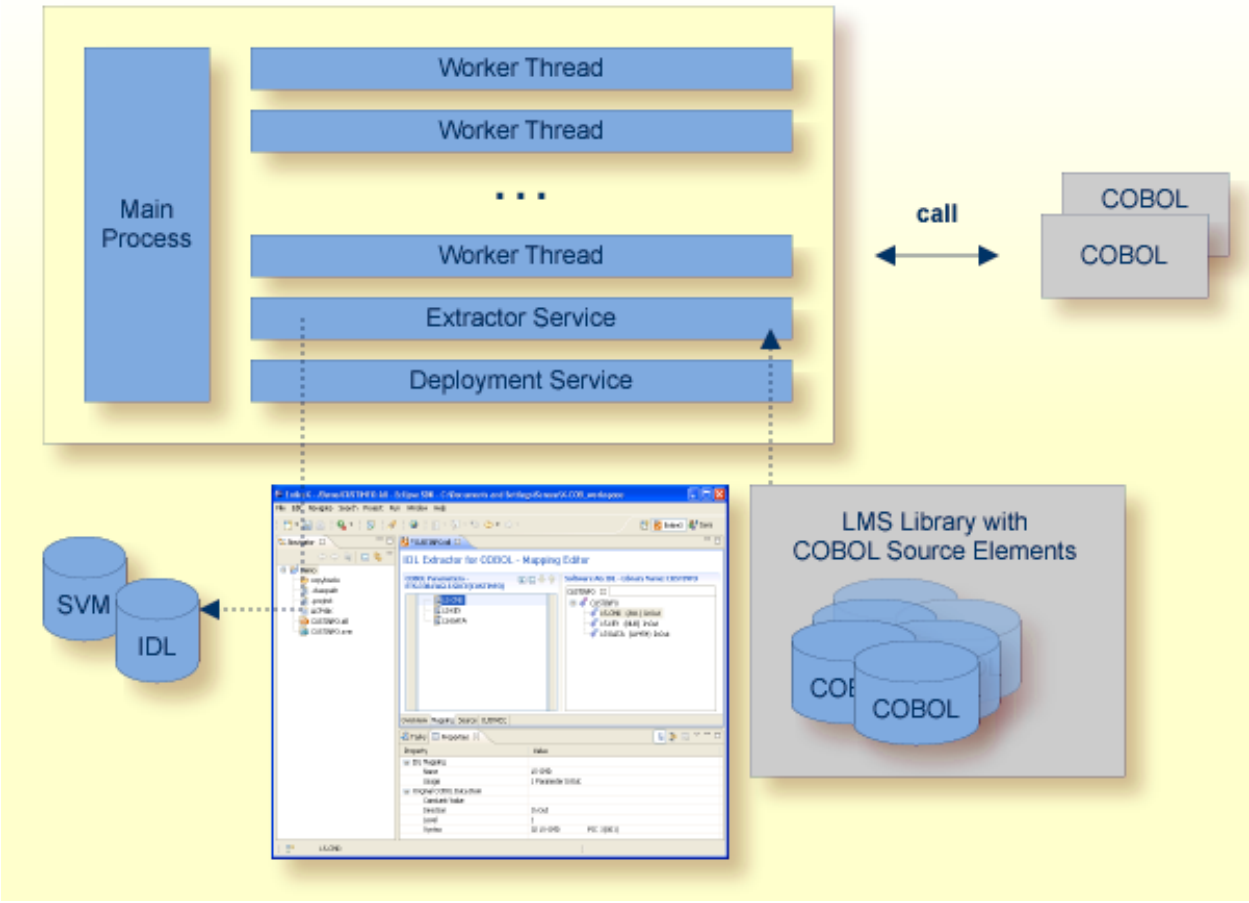
Inbuilt Services

RPC Server for BS2000 provides the following services for ease-of-use:

- Extractor Service
- Deployment Service

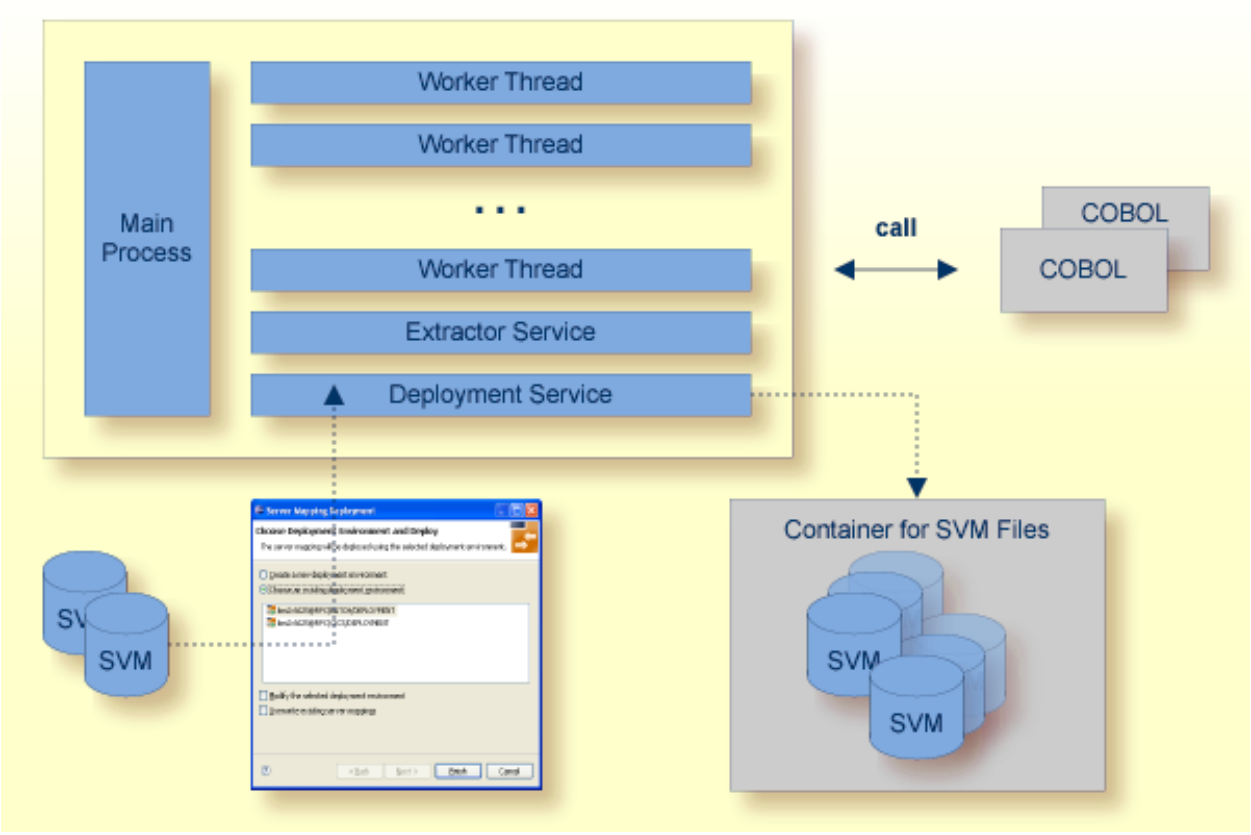
Extractor Service

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



Deployment Service

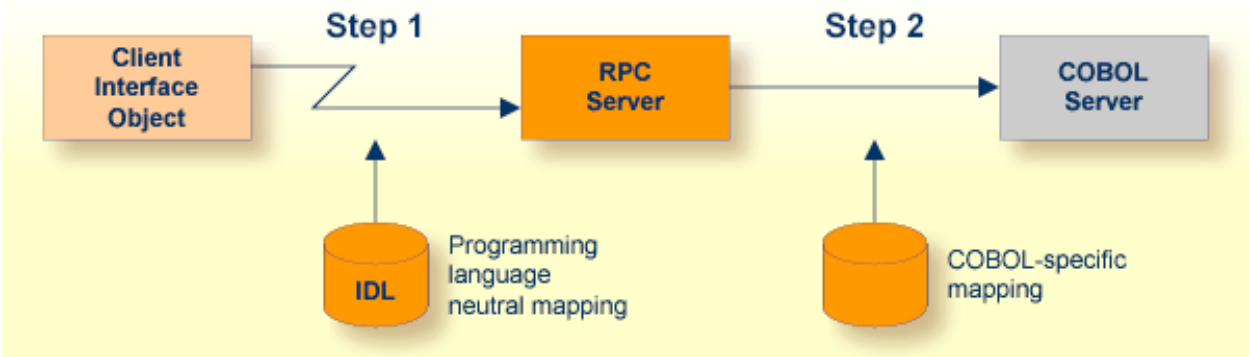
The Deployment Service allows you to deploy server-side mapping files (Designer files with extension .svm) interactively using the *Server Mapping Deployment Wizard*. On the RPC server side, the server-side mapping files are stored in a server-side mapping container (ISAM file). See ***Server-side Mapping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



Usage of Server Mapping Files

There are many situations where the RPC Server for BS2000 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 (*Software AG Designer* files with extension .svm) and *client*-side mapping files (Designer 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***.

3 Administering the RPC Server for BS2000

The EntireX RPC Server for BS2000 allows standard RPC clients to communicate with RPC servers on the operating system BS2000. It supports the programming languages COBOL and C.

Customizing the RPC Server

The following elements are used for setting up the RPC Server for BS2000:

- Common Runtime Environment (CRTE)
- Configuration File
- Start Procedure

Common Runtime Environment (CRTE)

When the RPC Server for BS2000 calls COBOL or C server programs, the BS2000 Common Runtime Environment (CRTE) is loaded dynamically into the corresponding address space of the worker task.

There is no need to bind the CRTE statically to the called server object modules. If this is needed for any reason, the CRTE must be linked as a subsystem. All entries must be hidden to prevent duplicates. Linking the CRTE statically will occupy resources and slow down the load time of the server object modules.

The CRTE is not delivered with this package. For a detailed description, see the *CRTE (BS2000) User's Guide*.

Configuration File

The name of the delivered example configuration file is "RPC-CONFIG". The configuration file contains the configuration for the RPC Server for BS2000. 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* in the RPC Server for BS2000 documentation
- scalability parameters
- trace settings
- etc.

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

Start Procedure

The name of the start S-procedure for the RPC Server for BS2000 is "START-RPC-SERVER". The start procedure contains the following:

- the location of the Common Runtime Environment (CRTE)
- the target server library name of the called COBOL or C server
- the configuration file used; see *Configuration File*
- 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, that is, the commands/parameters `broker=localhost` and `brok=localhost` are equivalents.

Parameter	Default	Values	Req/ Opt
`brokerid`	`localhost`	Broker ID used by the server. See *Using the Broker ID in Applications*. 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*). Case-sensitive, up to 32 characters. Corresponds to `CLASS`. Example:	R

Parameter	Default	Values	Req/ Opt
		`class=MyRPC`	
codepage	no codepage transferred	The codepage tells the broker the encoding of the data. The application must ensure the encoding of the data matches the codepage. The RPC server itself does not convert your application data. The application's data is shipped and received as given. Often, the codepage must also match the encoding used in the RPC server environment for file and terminal IO, otherwise unpredictable results may occur. By default, no codepage is transferred to the broker. It is assumed the broker's locale string defaults match. See *Locale String Mapping* If they do not match, provide the codepage here. Example: `codepage=EDF041` Enable character conversion in the broker by setting the service-specific attribute `CONVERSION` to `"SAGTRPC"`. See also *Configuring ICU Conversion* under *Configuring Broker for Internationalization* in the platform-specific Administration documentation. More information can be found under *Internationalization with EntireX*.	R

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

deployment	NO	Activates the deployment service, see ***Deployment Service***. Required to use the Server Mapping Deployment Wizard. See *Server Mapping Deployment Wizard* in the Designer documentation.	O

Parameter	Default	Values	Req/ Opt
		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	
init_exit		Initialization exit. The RPC Server for BS2000 provides user exits that allow you to plug in code during initialization and termination of RPC worker tasks. This parameter specifies the name of an executable module that is loaded and executed during initialization of each worker task. See also term_exit. Example: init_exit=myExit	O
extractor	NO	The extractor service is a prerequisite for remote extractions. See ***Extractor Service***. extractor=YES|NO Example: extractor=yes	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
marshalling	COBOL	The RPC Server for BS2000 can be configured to support either COBOL or C. See also ***Locating and Calling the Target Server***. marshalling=(LANGUAGE=COBOL|C) COBOL The RPC Server for BS2000 supports COBOL. The COBOL servers are called directly without a server interface object. The COBOL	O

Parameter	Default	Values	Req/ Opt
		server modules may be compiled as OM or LLM modules. So-called server mapping files are used to call the COBOL server correctly if one is available. See *Usage of Server Mapping Files* in the RPC Server for BS2000 documentation. `C` The RPC Server for BS2000 supports C. The modules are called using a server interface object built with the C Wrapper.	
`password`	no default	The password for secured access to the broker. Example: `password=MyPwd`	O
`restartcycles`	15	Number of restart attempts if the broker is not available. This can be used to keep the RPC Server for BS2000 running while the broker is down for a short time. A restart cycle will be repeated every 60 seconds. When the number of specified cycles is reached and a connection to the broker is not possible, the RPC Server for BS2000 stops. Example: `restartcycles=30` The server waits up to 30 minutes before it terminates due to a missing broker connection.	O
`servername`	`SRV1`	Server name part of the server address used by the server. The server address must be defined as a service in the broker attribute file. See *Service-specific Attributes*. Case-sensitive, up to 32 characters. Corresponds to `SERVER` of the broker attribute file. Example: `servername=mySrv`	R
`service`	`CALLNAT`	Service part of the server address used by the server. The server address must be defined as a service in the broker attribute file. See *Service-specific Attributes*. Case-sensitive, up to 32 characters. Corresponds to `SERVICE` attribute of the broker attribute file. Example: `service=MYSERVICE`	R

Parameter	Default	Values	Req/ Opt
svm	PREFERRED	Usage of server mapping files. See ***Server-side Mapping Files in the RPC Server***. SVM=PREFERRED|NO PREFERRED This setting is to support COBOL server programs that do not have server-side mapping, plus COBOL server programs built with a server-side mapping file. If you use server-side mapping files, the server-side mapping container must be installed and configured. See *Step 1: Define a Server-side Mapping Container* in the BS2000 Installation documentation. There are also client-side mapping files that do not require configuration here; see *Server Mapping Files for COBOL* in the Designer documentation. NO Server-side mapping files are not used. Example for BS2000: SVM=NO See also ***Usage of Server Mapping Files***.	O
term_exit		Termination exit. The RPC Server for BS2000 provides user exits that allow you to plug in code during initialization and termination of RPC worker tasks. This parameter specifies the name of an executable module that is loaded and executed during termination of each worker task. See also init_exit. Example: term_exit=myExit	O
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 and worker model DYNAMIC. Example: timeout=300	O
tracedestination	ERXTrace.*nnn*.log	Trace output is written to SYSOUT. See also ***Activating Tracing for the RPC Server***.	O

Parameter	Default	Values	Req/ Opt

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	The user ID for access to the broker. The default ERX-SRV will be used if this parameter is omitted or specified without a value: "userid=". Example: `userid=MyUid`	O

workermodel	SCALE,1,3,slowshrink	The RPC Server for BS2000 can be configured to
		
		■ use a DYNAMIC worker model, which adjusts 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 RPC Server for BS2000.
		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. This allows you to define a certain number of threads - not
			O

Parameter	Default	Values			Req/ Opt
			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. The *thru* value restricts the maximum number of all worker threads concurrently.		
			`slowshrink`	Default. 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.	
			`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

Target server programs are loaded dynamically, using the BS2000 BLSLIB chain. The target server library name needs to be set up as `PROGRAM-LIB` in the parameter declaration section of the START-RPC-SERVER S-procedure, see ***Start Procedure***. Different mechanisms are used depending on the language:

- COBOL

- C

COBOL

The approach used to derive the COBOL object 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 COBOL object module name of the RPC server is derived from this mapping. In this case the IDL program name can be different to the COBOL object 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 COBOL object module name of the RPC server (the IDL library name is ignored).

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

To use the RPC Server for BS2000 with COBOL

1 Make sure that all target server programs called as RPC servers

- are COBOL object modules
- use COBOL calling conventions

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

```
marshalling=COBOL
```

C

To use the RPC Server for BS2000 with C

1 Make sure that all target server programs called as RPC servers

- are C object modules
- use C calling conventions

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

```
marshalling=C
```

See *Scenario III: Writing a New C Server* in the RPC Server for BS2000 documentation.

Starting the RPC Server

To start the RPC Server for BS2000

- Use the following SDF command:

```
/ENTER-PROCEDURE *LIB(LIB=EXP105.JOBS,ELE=START-RPC-SERVER), -
/JOB-NAME=RPCMAIN,LOG=*NO
```

Stopping the RPC Server

To stop the RPC Server for BS2000 from a privileged user ID

- Enter the command:

```
/INFORM-PROGRAM MSG='STOP',JOB-IDENTIFICATION=*TSN(TSN=tsn)
```

where *tsn* is the task number associated with the RPC Server for BS2000 main task (in the example above the TSN of `RPCMAIN`)

All other tasks that were created as a result of starting the RPC Server for BS2000 will be stopped automatically.

To stop the RPC Server for BS2000 from an operator console

- Enter the command:

```
/INTR tsn,STOP
```

where *tsn* is the task number associated with the RPC Server for BS2000 main task (in the example above the TSN of RPCMAIN)

All other tasks that were created as a result of starting the RPC Server for BS2000 will be stopped automatically.

To stop the RPC Server for BS2000 from a non-privileged user ID

- Use S-procedure STOP-RPC-SERVER in EXP105.JOBS.

Startup Parameter	Description	Default

BROKER-ID	Depending on the communication method, the broker ID can be specified in two different formats:
	■ **TCP Transport Method**
	`ip:port:TCP`
	where *ip* is the address or DNS host name,
	port is the port number that EntireX Broker is listening on, and
	TCP is the protocol name
	■ **NET Transport Method**
	`ETBnnn:SVCmmm:NET`
	where *nnn* is the ID under which EntireX Broker is connected to the Adabas ID table,
	mmm is the SVC number under which the Adabas ID table can be accessed, and
	NET is the protocol name
		none

CLASS	The class name under which the RPC server is registered at the EntireX Broker.	RPC

Startup Parameter	Description	Default
SERVER	The server name under which the RPC server is registered at the EntireX Broker.	SRV1
SERVICE	The service name under which the RPC server is registered at the EntireX Broker.	CALLNAT
USERID	If EntireX Broker is running with EntireX Security, a user ID needs to be supplied	none
PASSWORD	If EntireX Broker is running with EntireX Security, a password needs to be supplied	none
EXX-JOBS	EntireX Broker jobs library	EXX105.JOBS
EXX-LIB	EntireX Broker module library	EXX105.LIB
WAL-MOD	WAL module library	WAL842.MOD

Set the broker ID in the `PARAMETER-DECLARATION` section and enter following command:

```
/CALL-PROCEDURE (EXP105.JOBS, STOP-RPC-SERVER)
```

Activating Tracing for the RPC Server

To switch on tracing for the RPC server

- Set the parameter `TRACELEVEL` in S-element `RPC-CONFIG` in `EXP105.JOBS`.

  To evaluate the return codes, see *Error Messages and Codes*.

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 (Designer files with extension .cvm) and server-side mapping files (Designer 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 Designer documentation.

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

Server-side Mapping Files in the RPC Server

Under BS2000, server-side mapping corresponds to lines of Designer files with extension .svm. See *Server Mapping Files for COBOL*. The mapping information is stored as records within one ISAM file, the server-side mapping container. This container contains all server-side mapping entries from all Designer files with extension .svm. The unique key of the ISAM 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 (ISAM 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 (ISAM file) to the RPC server. See configuration parameter `svm`.

Deploying Server-side Mapping Files to the RPC Server

Deploy a server-side mapping file (Designer file with extension .svm) with the Server Mapping Deployment Wizard. See *Server Mapping Files for COBOL* in the Designer documentation.

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

1 Make sure the 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** COBOL > Deploy/Synchronize Server Mapping and call the Deployment Wizard. See *Server Mapping Deployment Wizard* in the Designer documentation.

Undeploying Server-side Mapping Files from the RPC Server

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

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

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

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

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

Change Management of Server-side Mapping Files

Under BS2000, change management for an ISAM file (server-side mapping container, see ***Server-side Mapping Files in the RPC Server***) is similar to change management for an ordinary file. All updates to the ISAM file done after a backup must be kept.

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

List Deployed Server-side Mapping Files

Use the command `SHOW-FILE` to list the contents of the server-side mapping container. See ***Server-side Mapping Files in the RPC Server***.

```
SHOW-FILE <server-mapping-file>
```

where `<server-mapping-file>` is the server-side mapping container (ISAM file) containing all server-side mapping information.

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

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

The timestamp can be found on the same offset in the records in the server-side mapping container (ISAM 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***.

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 Designer. If you have not used server-side mapping, we recommend you use client-side mapping. See *Server Mapping Files for COBOL*.

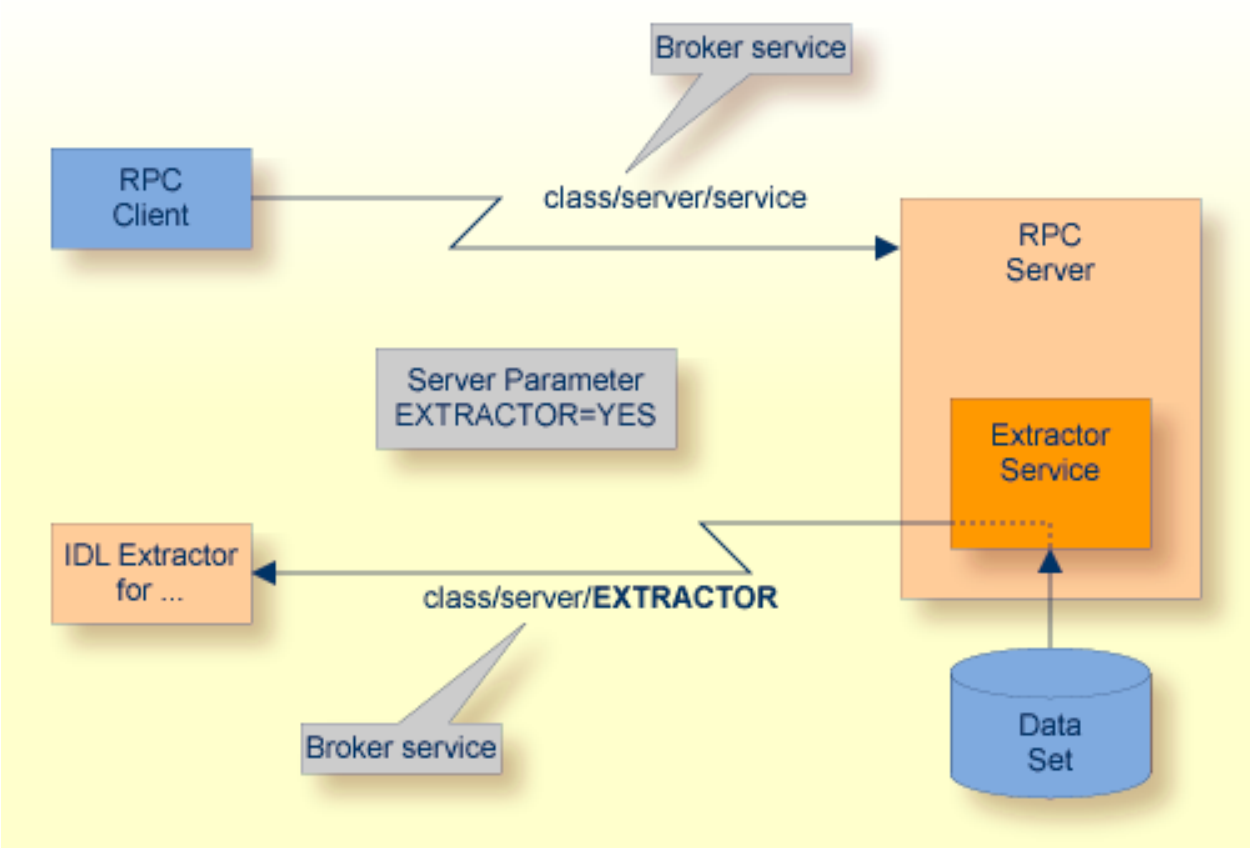
You can also call COBOL servers generated or extracted with previous versions of EntireX mixed with a COBOL server that requires server-side mapping. All EntireX RPC servers are backward compatible.

5 Extractor Service

Introduction

The extractor service

- provides access to LMS libraries
- 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* 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*.

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*.

Enabling the Extractor Service

To enable the extractor service

1 Set the RPC Server for BS2000 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 section *EntireX Security under z/OS*.
- For a UNIX or Windows broker, see *Authorization Rules* in the platform-independent administration documentation.
- Not applicable to a BS2000 broker.

Disabling the Extractor Service

To disable the extractor service

- Set the RPC Server for BS2000 parameter `extractor=no`. See `extractor` under *Configuring the RPC Server*. The RPC Server for BS2000 will not register the extractor service in the broker.

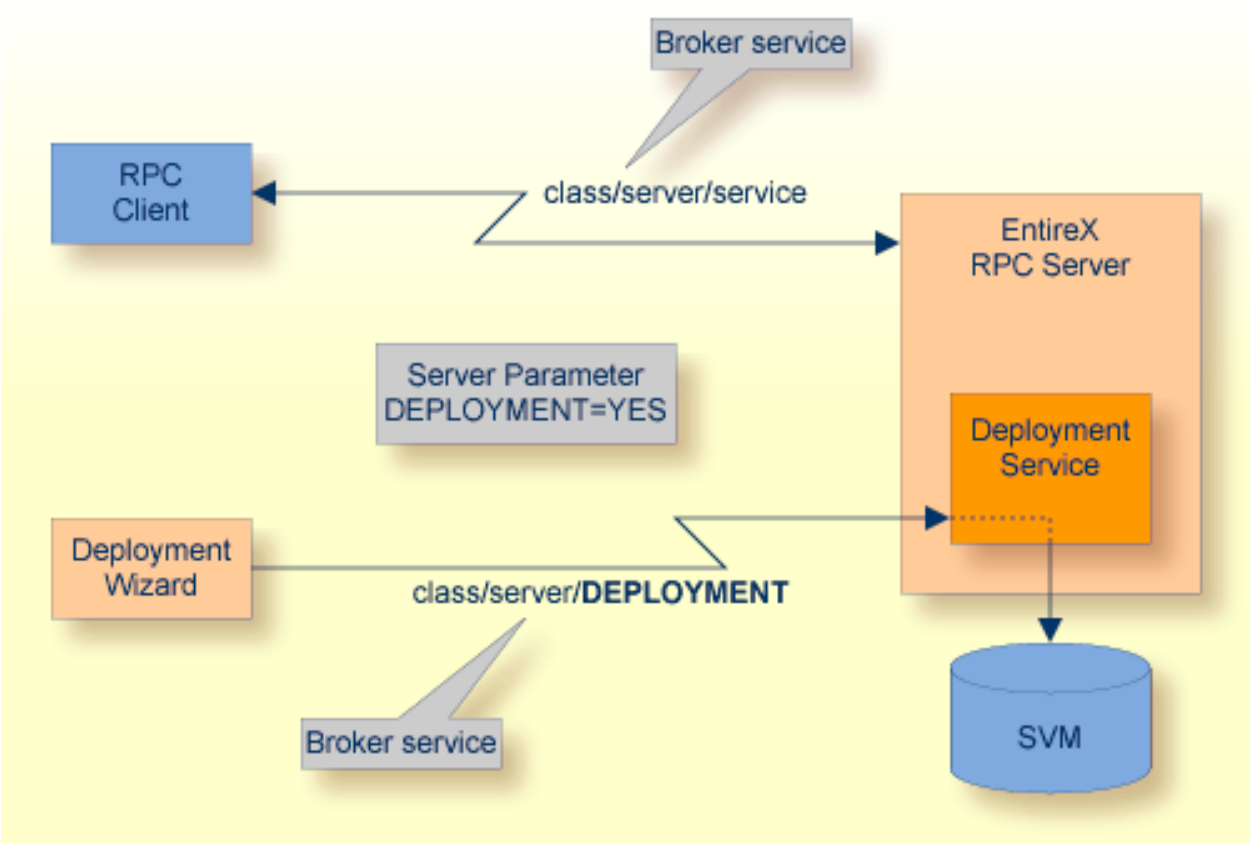
6 Deployment Service

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* 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 Designer 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 an RPC Server for BS2000, the server-side mapping container (ISAM file) must be installed and configured. See *Step 1: Define a Server-side Mapping Container* in the BS2000 Installation documentation.

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

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

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

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

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

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

- For a z/OS broker, see *Resource Profiles in EntireX Security*.

- For a UNIX or Windows broker, see *Authorization Rules*.
- Not applicable to a BS2000 or z/VSE broker.

Disabling the Deployment Service

To disable the deployment service

- Set the RPC Server for BS2000 parameter `deployment=no`. See `deployment` under ***Configuring the RPC Server***.

  The RPC Server for BS2000 will not register the deployment service in the broker.

7 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 Designer documentation.

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

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

See *Client and Server Examples for BS2000* 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 Designer 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 Designer documentation
- generate an XML mapping file (XMM) and use the XML Tester for verification; see *EntireX XML Tester* in the XML/SOAP Wrapper documentation

See *Client and Server Examples for BS2000* in the COBOL Wrapper documentation for COBOL RPC Server examples.

C Scenarios

Scenario III: 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, IBM i)*.

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

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