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*.

This chapter covers the following topics:

- Customizing the RPC Server with a Configuration File
- Configuring the RPC Server
- Locating and Calling the Target Server
- Starting the RPC Server
- Stopping the RPC Server
- Activating Tracing for the RPC Server

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*. 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* under *Broker Attributes*). 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*, the correct codepage (locale string) must be provided. This means it must: • follow the rules described under *Locale String Mapping* • 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	

Parameter	Default	Values	Req/ Opt
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 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*) 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. Example: runoption=<option> runoption=<option>	O

Parameter	Default	Values	Req/ Opt
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* under *Broker 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* under *Broker Attributes*. 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. ***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*.	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. Example: timeout=300	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	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

Parameter	Default	Values	Req/ Opt
workermodel	SCALE,1,3,slowshrink	The Batch RPC Server can be configured to • 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)	O

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. 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.