Administering the Batch RPC Server

The EntireX z/OS Batch RPC Server allows standard RPC clients to communicate with RPC servers on the operating system z/OS running in batch mode. It supports the programming languages COBOL, PL/I and C.

This chapter covers the following topics:

- Customizing the RPC Server
- Configuring the RPC Server
- Locating and Calling the Target Server
- Using SSL or TLS with the RPC Server
- Starting the RPC Server
- Stopping the RPC Server
- Activating Tracing for the RPC Server

Customizing the RPC Server

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

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

Configuration File

The name of the delivered example configuration file is `CONFIG` (see source library EXP970.SRCE). The configuration file is specified as a DD definition with a user-defined DD name in the *Started Task JCL*. The configuration file contains the configuration for the 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*.

IBM LE Runtime Options

Depending on the feature the Batch 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
Trap abends of called RPC server programs	`ABTERMENC(RETCODE)` (1)	Required to also trap the LE abends within a server program.
Level of information if called RPC server program terminates by unhandled condition	`TERMTHDACT(UADUMP)` (1)	Forces a U4039 system dump for abends not trapped by the server.
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:
(1) Set internally by the Batch 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 `EXPSRVB` (see EntireX job library EXX970.JOBS). The started task contains the following:

- the target server libraries of the called COBOL or PL/I server
- the configuration file used; see *Configuration File*; specified as a DD definition with a user-defined DD name as RPC server startup argument CFG:

`CFG=DD:`*`ddname`*

Example using the DD name `CONFIG`:

```
CFG=DD:CONFIG
```

- 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
`brokerid`	`localhost`	Broker ID used by the server. See *Using the Broker ID in Applications*. Example: `brokerid=myhost.com:1971`	R
`ceeoptions`		Allows you to change IBM's LE runtime options. This parameter is deprecated. See *IBM LE Runtime Options* for how to set the LE runtime options.	O
`class`	`RPC`	Server class part of the server address used by the server. The server address must be defined as a service in the broker attribute file (see *Service-specific Attributes* under *Broker Attributes*). Case-sensitive, up to 32 characters. Corresponds to `CLASS`. Example: `class=MyRPC`	R

Parameter	Default	Values	Req/ Opt
codepage	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 *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`	O

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	BROKER	Define the broker stub to be used. See *Administering Broker Stubs* for available stubs. Example: `etblnk=broker`	O

extractor	NO	The extractor service is a prerequisite for remote extractions. See *Extractor Service*.
		
		`extractor=YES|NO`
		
		Example:
		`extractor=yes`
			O

Parameter	Default	Values	Req/ Opt

impersonation	NO	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 *Impersonation*.
		
		`impersonation= NO | YES | AUTO [, sameuser | , anyuser ]`
		
		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 *RPC client user ID*. The Batch RPC Server validates the *RPC client user ID/password* 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
		
		• a secure broker (validation must be against the correct mainframe security repository where the user IDs are defined) and option `sameuser`
		or
		• your own security implementation (option `anyuser` is supported for compatibility reasons if you need different broker and server user IDs - the customer-written security implementation must validate the RPC client using the *RPC client user ID*)
		
		sameuser The Batch RPC Server checks whether the *broker client user ID* matches the *RPC client user ID*. This is the default if AUTO is used.
		
		anyuser The *RPC client user ID* is used for impersonation. The *broker client user ID* is ignored.
		
		Notes:
		
		1. EntireX supports two user ID/password pairs: a *broker client user ID/password* pair and an (optional) *RPC user ID/password* pair sent from RPC clients to the RPC server.
		2. With EntireX Security, the *broker client user ID/password* pair is checked. The *RPC user ID/password* 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.
		3. RPC clients send the (optional) *RPC user ID/password* pair in the same way as specifying the Natural user ID/password pair for a Natural RPC Server. See for example *Using Natural Security* for applications under C | COBOL | PL/I | Web Services | SOAP/XML | Java.
		4. If the RPC client does not specify the optional *RPC user ID/password pair*, the *broker client user ID* is inherited to the *RPC user ID* and thus used for impersonation by the Batch RPC Server.
		
		Example:
		`impersonation=auto,anyuser`
		
		Using impersonation requires additional installation steps. See *Using z/OS Privileged Services*.
			O

Parameter	Default	Values	Req/ Opt

library	no default	`library = search-logic [- library] where search-logic is one of FIX(dllname) | PREFIX(prefix) | PREFIX()`
		
		This parameter applies to programming language C only. Do not set if other programming languages for RPC server are used.
		
		FIX(*dllname*) The IDL library name coming from the RPC client is ignored, thus long IDL library names can be used. You have to define the DLL names for all client interface objects and RPC servers.
		
		PREFIX(*prefix*) The IDL library name coming from the RPC client is used to form the DLL name. As *prefix* you can define any character. If an RPC client sends, for example, "SYSTEM" as the IDL library name and "D" is defined as *prefix*, the DLL name derived is "DSYSTEM". This configuration restricts the IDL library names to max. 7 characters.
		
		PREFIX() The IDL library name coming from the RPC client is used as DLL name. This configuration restricts the IDL library names to max. 8 characters.
		
		Example PREFIX configuration (this configuration matches the standard names produced by the C Wrapper):
		`library=PREFIX(D)-PREFIX()`
		
		Example FIX configuration:
		`library=FIX(MYSTUBS)-FIX(MYRPCS)`
			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, PL/I or C. See also *Locating and Calling the Target Server*. `marshalling=(LANGUAGE=COBOL|PLI [flavor=ENTERPRISE | MVS] | C)` **COBOL** Server supports COBOL. The COBOL servers are called directly without a server interface object. So-called server mapping files are used to call the COBOL server correctly if one is available. See *Usage of Server Mapping Files*. **PLI** Server supports PL/I 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 `flavor` can be omitted. **ENTERPRISE** Enterprise compiler z/OS. This is the default if PL/I is used. See prerequisites for PL/I Wrapper. **MVS** Server supports PL/I Server compiled with older IBM compiler PL/I MVS & VM V1R1 and above. See prerequisites for PL/I Wrapper. **C** Server supports C. The modules are called using a server interface object built with the *C Wrapper*.	O
password	no default	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

return_code	NO	Enable application-specific errors.
		`return_code=(NO|YES)`
		
		NO No tests of COBOL special register `RETURN-CODE` for application-provided error.
		
		YES After execution of the RPC server, tests COBOL special register `RETURN_CODE` for application provided error. See *Aborting RPC Server Customer Code and Returning Error to RPC Client*.
		
		Example:
		`return_code=yes`
			O

Parameter	Default	Values	Req/ Opt
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
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
ssl file	no default	Set the SSL parameters. See *Using SSL or TLS with the RPC Server* for examples and more information.	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 DD name ERXSVM. If this DD 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 *Install the Server-side Mapping Container for a Batch RPC Server (Optional)* under *Installing the EntireX RPC Servers under z/OS*. 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| ddname`
		
		no No server-side mapping files are used.
		ddname DD name of the server-side mapping container in the started task JCL of the Batch RPC Server.
		
		Example:
		`svm=MYSVM`
		
		For the example above, define the DD name MYSVM in the started task JCL of the Batch RPC Server (see *Started Task JCL*) as
		
		`//MYSVM DD DISP=SHR,DSN=<svm.cluster>`
		
		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

tracedestination	DD:ERXTRACE	The name of the destination file for trace output.
		
		`tracedestination=DD:`*ddname*, where *ddname* is the name of the trace file.
		
		Example:
		`tracedestination=DD:MYTRACE`
		
		The DD name MYTRACE must be defined in the started task of the Batch RPC Server (see *Started Task JCL*):
		
		`//MYTRACE DD DISP=SHR,DSN=<rpctrace-file>`
			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

traceoption	None	Additional trace option if trace is active. **None** No additional trace options. **STUBLOG** If `tracelevel` is `Advanced` or `Support`, the trace additionally activates the broker stub log. **NOTRUNC** Normally if a data buffer larger than 8 KB is traced, the buffer trace is truncated. Set this option to write the full amount of data without truncation. **Note:** This can increase the amount of trace output data dramatically if you transfer large data buffers. Example: `traceoption=(STUBLOG,NOTRUNC)`	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
- PL/I
- C
- Assembler (IBM 370)

COBOL

The approach used to derive the z/OS 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/OS module name of the RPC server is derived from this mapping. 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*.
4. If no server mapping is 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 Batch 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 Batch 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 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*.

PL/I

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 Batch RPC Server with PL/I

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 PL/I calling conventions
   - can be called dynamically ("fetched") from any Language Environment program
   - are accessible through the Batch RPC Server started task JCL `STEPLIB` concatenation. See *Started Task JCL*.
2. Configure the parameter `marshalling` for PL/I, for example `marshalling=PLI`.

See also *Scenario III: Calling an Existing PL/I Server* or *Scenario IV: Writing a New PL/I Server*.

C

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

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

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 Batch RPC Server with C

1. Make sure all dynamic-link libraries (DLLs) called as RPC servers and client interface objects are accessible through the Batch RPC Server started task JCL `STEPLIB` concatenation. See *Started Task JCL*.
2. Configure the parameter `marshalling` for C, for example `marshalling=C`.
3. Configure the parameter *`library`* either with the `FIX` configuration or `PREFIX` configuration, depending on how you have built your DLLs. See *Using the C Wrapper for the Server Side (z/OS, 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 Batch RPC Server with Assembler

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

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

Using SSL or TLS with the RPC Server

The Batch 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 `POSIX(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 Batch RPC Server connects to is prepared for SSL/TLS connections as well. See the following sections:
   - *Running Broker with SSL or TLS Transport* under z/OS | UNIX | Windows
   - *Settting up and Administering the Broker SSL Agent* under UNIX | Windows
   - Direct RPC in the EntireX Adapter documentation under http://documentation.softwareag.com > *webMethods Product Line*

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 `POSIX(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 Batch 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 Batch RPC Server connects to is prepared for SSL/TLS connections as well. See the following sections:
   - *Running Broker with SSL or TLS Transport* under z/OS | UNIX | Windows
   - *Settting up and Administering the Broker SSL Agent* under UNIX | Windows
   - Direct RPC in the EntireX Adapter documentation under http://documentation.softwareag.com > *webMethods Product Line*

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

To set up SSL or TLS with AT-TLS

1. Set up the Batch RPC Server for a TCP/IP connection.
2. Configure the rules for the AT-TLS policy agent the Batch RPC Server matches, for example by using the job name and remote port number the Batch 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 Batch RPC Server connects to is prepared for SSL/TLS connections as well. See the following sections:
   - *Running Broker with SSL or TLS Transport* under z/OS | UNIX | Windows
   - *Settting up and Administering the Broker SSL Agent* under UNIX | Windows
   - Direct RPC in the EntireX Adapter documentation under http://documentation.softwareag.com > *webMethods Product Line*

Starting the RPC Server

To start the Batch RPC Server

1. Modify the member `EXPSRVB` (see EntireX job library EXX970.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 EXPSRVB
   ```

   Or:
   Add the task to your system automation tool(s)

Stopping the RPC Server

To stop the Batch RPC Server

- Use the operator command `STOP`. Examples:

  ```
  /p EXPSRVB
  /f EXPSRVB,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* under UNIX | Windows.

Activating Tracing for the RPC Server

To switch on tracing for Batch RPC Server

1. Set the parameters `tracelevel` and `tracedestination`.
2. Dynamically change the trace level with the operator command

   ```
   F EXPSRVB,TRACELEVEL=tracelevel,
   ```

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

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

   ```
   F EXPSRVB,TRACELEVEL
   ```

   might reply with the operator message

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

   To evaluate the return codes, see *Component Return Codes in EntireX*.