Administration of the EntireX Java RPC Server

The EntireX Java RPC Server is an RPC server which runs Java server interface objects generated from your IDL files. This server can register an Attach Service to start several services with the same server address on demand.

Each of these services can process one call at a time. The Java RPC Server is started by a script, which you may customize. Parameters for the server are configured in a Java properties file.

This chapter covers the following topics:

- Customizing the Java RPC Server
- Using Package Names with the Java RPC Server
- Using Internationalization with Java RPC Server
- Starting the Java RPC Server
- Stopping the Java RPC Server
- Application Identification

Customizing the Java RPC Server

- Introduction
- The Properties File
- Example
- Properties and Command-line Options

Introduction

The script files that start the Java RPC Server allow command-line options as described in the table below. Alternatively, you can use System properties or a property file. The command-line option has the highest priority; the System property has second priority, and the entries of a property file have third priority.

The Java RPC Server can adjust the number of worker threads to the number of parallel requests. Use the properties `entirex.server.fixedservers`, `entirex.server.maxservers` and `entirex.server.minservers` to configure this scalability. If `entirex.server.fixedservers=yes`, the number of servers specified in `entirex.server.minservers` is started and the server can process this number of parallel requests. If `entirex.server.fixedservers=no`, the number of worker threads balances between what is specified in `entirex.server.minservers` and what is specified in `entirex.server.maxservers`. This is done by a so-called attach server thread. At startup, the number of worker threads is the number specified in `entirex.server.minservers`. A new worker thread starts if the Broker has more requests than there are worker threads waiting. If more than the number specified in `entirex.server.minservers` are waiting for requests, a worker thread stops if its receive call times out. The timeout period is configured with `entirex.server.waitserver`.

The Properties File

The default name of the properties file is `entirex.server.properties`. It can be changed by assigning an arbitrary file name with a path to a Java system property with the name `entirex.server.properties`. The file is searched for in the directory of the start script.

An example for the properties file is in subfolder *config* of the installation folder.

Example

Under UNIX:

```
java -Dentirex.server.properties=rpcserver.properties -classpath <entirex.jar with path>:<path to your server
stubs> com.softwareag.entirex.aci.RPCServer
```

Properties and Command-line Options

Name	Command-line Option	Default Value	Explanation
entirex.rpcserver.packagename.entirex. rpcserver.packagename.<libraryname>=packagename<libraryname>=packagename			See *Using Package Names with the Java RPC Server*.
entirex.server.brokerid	-broker	localhost	Broker ID
entirex.server.codepage	-codepage		The codepage the server uses. Permitted values are the name of the codepages the JVM supports. See *Customizing the Java RPC Server* for details.
entirex.server.compresslevel	-compresslevel	0 (no compression)	Permitted values (you can enter the text or the numeric value): BEST_COMPRESSION 9 BEST_SPEED 1 DEFAULT_COMPRESSION -1, mapped to 6 DEFLATED 8 NO_COMPRESSION 0 N 0 Y 8
entirex.server.customclass	-customclass		This class is used for custom initialization and shutdown of the server. In addition, this class allows handling when closing a conversation and handling the termination of a worker thread. See `ServerImplementation` for more information.
entirex.server.encryptionlevel	-encryption	0	Encryption level (if Broker is version 6.1.1 or higher. Valid values: 0,1,2).
entirex.server.environment			Can be used in a user-written translation exit of the Broker. See `BrokerService`, `setEnvironment(java.lang.String)` (EntireX Java ACI).
entirex.server.fixedservers		no	If no, use attach server to manage worker threads, otherwise run minimum number of server threads.
entirex.server.logfile	-logfile		Path and name of the trace output file. Environment variables in the name are resolved only if used as command-line option.
entirex.server.maxservers		32	Maximum number of worker threads.
entirex.server.minservers		1	Minimum number of server threads.
entirex.server.monitorport	-smhport	0	The port where the server listens for commands from the System Management Hub (SMH). If this port is 0, no port is used and management by the SMH is disabled.
entirex.server.name			The name of the server.

Name	Command-line Option	Default Value	Explanation
entirex.server.password	-password		The password for secured access to the Broker. The password is encrypted and written to the property `entirex.server.password.e`. To change the password, set the new password in the properties file (default is *entirex.server.properties*). To disable password encryption, set `entirex.server.passwordencrypt=no`. Default for this property is yes.
entirex.server.properties	-propertyfile	entirex.server.properties	The file name of the property file.
entirex.server.restartcycles	-restartcycles	15	Number of restart attempts if the Broker is not available. This can be used to keep the Java RPC Server running while the Broker is down for a short time.
entirex.server.security	-security	no	no/yes/auto/Name of BrokerSecurity object.
entirex.server.serveraddress	-server	RPC/SRV1/CALLNAT	Server address
entirex.server.serverlog	-serverlog		Name of the file where start and stop of worker threads is logged. Used by the Windows RPC Service.
entirex.server.userid	-user	JavaServer	The user ID for the Broker for RPC. See entirex.server.password .
entirex.server.verbose	-verbose	no	Verbose output to standard output yes/no.
entirex.server.waitattach		600S	Wait timeout for the attach server thread.
entirex.server.waitserver		300S	Wait timeout for the worker threads.
entirex.timeout		20	TCP/IP transport timeout. See *Setting the Transport Timeout* under *Writing Advanced Applications - EntireX Java ACI*.
entirex.trace	-trace	0	Trace level (1,2,3).

Using Package Names with the Java RPC Server

A package name can be specified when the server is generated.

The Java RPC Server can handle server programs with package names if the package name of each library is configured in the properties of the server. For each library the property `entirex.rpcserver.packagename.<library>` has the value of the package.

Example for the library Example (as in *example.idl*):

```
entirex.rpcserver.packagename.example=my.package
```

The library name must be lowercase.

Using Internationalization with Java RPC Server

It is assumed that you have read the document *Internationalization with EntireX* and are familiar with the various internationalization approaches described there.

With the parameter codepage for the Java RPC Server you can

- override the encoding used for the payload sent to / received from the broker. Instead of using the default encoding of the JVM, the given encoding is used. Using this method does not change the default encoding of your JVM.
- force a locale string to be sent if communicating with broker version 7.1.x and below. You can use the abstract codepage name LOCAL to send the default encoding of the JVM to the broker. See *Using the Abstract Codepage Name LOCAL*.

EntireX Java components use the codepage configured for the Java virtual machine (JVM) to convert the Unicode (UTF-16) representation within Java to the multibyte or single-byte encoding sent to or received from the broker by default. This codepage is also transferred as part of the locale string to tell the broker the encoding of the data if communicating with a broker version 7.2.x and above.

To change the default, see your JVM documentation. On some JVM implementations, it can be changed with the `file.encoding` property. On some UNIX implementations, it can be changed with the `LANG` environment variable.

Which encodings are valid depends on the version of your JVM. For a list of valid encodings, see Supported Encodings in your Java documentation. The encoding must also be a supported codepage of the broker, depending on the internationalization approach.

Starting the Java RPC Server

To start the Java RPC Server

- Use a shell script in the subfolder *bin* of the installation directory.

On UNIX, the shell script is named *jrpcserver.bsh*.

If the Java interpreter is not called "java", change the call to "java".

- You can set the environment variable JAVA_HOME for the location of the Java interpreter.
- Set the classpath to "entirex.jar" and the path to the generated proxies.
- The Java RPC Server accepts parameters. See column Command-line options in table above.

Stopping the Java RPC Server

To stop the Java RPC Server

- Use the function **Deregister a Service** or **Deregister a Server** of the System Management Hub. This method ensures that the deregistration from the Broker is correct.

Application Identification

The application identification is sent from the RPC server to the Broker. It is visible with Broker Command and Info Services.

The identification consists of four parts: name, node, type, and version. These four parts are sent with each Broker call and are visible in the trace information.

For the Java RPC Server these values are:

Identification Part	Value
Application name:	ANAME=Java RPC Server
Node name:	ANODE=<*host name*>
Application type:	ATYPE=Java
Version:	AVERS=8.2.0.0