Reference - HTTP and Java Interface

This chapter covers the following topics:

- Client Using the Java Interface
- The Java Interface
- The HTTP Interface

Client Using the Java Interface

See also the delivered example `XMMInvoker` and `XMLRPCService` (EntireX XML/SOAP Runtime).

Step 1: Writing the Client Program

To write the client program

1. Initialize the `Broker` Object

   ```
   Broker broker = new Broker(brokerID, userID);
   // if a logon should be issued
    broker.logon(password);
   ```

2. Initialize the `XMLRPCService` Object

   ```
   XMLRPCService service;
   service = new XMLRPCService(xmmfilename);//constructor with XMM
   ```

   or

   ```
   XMLRPCService service;
   service = new XMLRPCService(broker, serverName, xmmfilename);
   ```

3. Initialize the `Conversation` Object

   For one `XMLRPCService`, only one conversation may be used at a time.

   ```
     private Conversation conv;
     conv = new Conversation(service);
   ```

4. Assign the incoming XML document:

   ```
   String xmlDocument = "<?xml version="1.0" encoding="iso8859-1"?>" +
   "<CALC Operation=\"-\" Operand_1=\"1000000611\"
   Operand_2=\"1000000288\"></CALC>";
   ```

5. Invoke the Service

   Non-conversational

   ```
   try
   {
     String result = service.invokeXML(xmlDocument);
   }
   ```

```
catch (XMLException e)
{
   // Error handling ...
}
catch (BrokerException e)
{
  // Error handling ...
}
catch (Exception e)
{
  // Error handling ...
}
```

Conversational

```
try
{
  service.setConversation(conv);
  String result = service.invokeXML(xmlFromFile(filename));
  service.closeConversationCommit();
}
catch (XMLException e)
{
   // Error handling ...
}
catch (BrokerException e)
{
  // Error handling ...
}
catch (Exception e)
{
  // Error handling ...
}
```

The string result contains the returned document. It should be:

```
<CALC Function_result="899" />
```

Step 2: Running the Client Program

The definition of the Java classpath must include

- the generated XMM file, see *Generating an XMM File*
- the *entirex.jar* file (delivered in the directory *<EntireX Home>/classes*)
- the files for an XMLStreamParser, for example the delivered *wstx-asl.jar* and *stax-api.jar* (in the directory *<EntireX Home>/classes*)

The Java Interface

Class XMLRPCService

`XMLRPCService` extends `com.softwareag.entirex.aci.RPCService`. See `XMLRPCService` (EntireX XML/SOAP Runtime).

The HTTP Interface

The HTTP interface supports HTTP `POST` Request and HTTP `GET` Request. The following HTTP headers and parameters are available:

Header/Parameter	Direction	Description
`exx-brokerID`	IN	If this attribute is set, it overwrites both the properties of the XMM file and the settings of the servlet initialization.
`exx-service`	IN	The service name is the triple set of server class/server name/service. If this attribute is set, it overwrites both the properties of the XMM file and the settings of the servlet initialization.
`exx-userID`	IN	The userID specified here is used for calling the broker.
`exx-password`	IN	The password specified here is used for calling the broker.
`exx-use-security`	IN	Possible values: `true` | `false`. This optional parameter determines whether EntireX Security is used. If the parameter is *not* defined for the XML/SOAP Listener / XML Tester (Quick test), the runtime determines if EntireX Security is required or not. If this parameter is defined, the behavior is fixed depending on value. See *EntireX Security for EntireX Broker*.
`exx-encryption-level`	IN	Possible values: `0` | `1`| `2`. See `ENCRYPTION-LEVEL`, class `Broker` and method `setSecurity`.
`exx-rpc-userID`	IN	The RPC user ID specified here is used for Natural Security. If no RPC user ID and no RPC password is defined, the values of `exx-userID` and `exx-password` are used for these values.
`exx-rpc-password`	IN	The RPC password specified here is used for Natural Security. If no RPC user ID and no RPC password is defined, the values of `exx-userID` and `exx-password` are used for these values.
`exx-natural-security`	IN	Possible values: `true` | `false`. Determines whether Natural Security is used. See *Using Natural Security* in the Java Wrapper documentation.
`exx-natural-library`	IN	The Natural library. Applicable only if `exx-natural-security` is "`true`". See *Using Natural Security* in the Java Wrapper documentation.
`exx-use-codepage`	IN	Determines the translation processing of the EntireX Broker. Valid values: `true` | `false` | <`character encoding`>. If a character encoding is set, this character encoding is used for RPC message. See method `useCodePage` and `setCharacterEncoding` in the documentation on class `BrokerService` (EntireX Java ACI).

Header/Parameter	Direction	Description
`exx-compresslevel`	IN	Sets the compression level. See *Using Compression* under *Writing Advanced Applications - EntireX Java ACI*.
`exx-compression`	IN	Possible values: `true` |`false`. See *Using Compression*.
`exx-xml-sessionID`	IN OUT	This HTTP header is returned from the servlet to the client application. If the client returns it to the servlet with the preceding call, the same session will be used. For conversations, the `exx-conv` parameter is also required.
`exx-xml-info`	OUT	In this HTTP header additional information is returned.
`exx-xml-error`	OUT	In this HTTP header error information is returned.
`exx-conv`	IN	Possible values: `OPEN` | `COMMIT` | `BACKOUT`. Conversations can only be used in connection with sessions. If the session is interrupted, the conversation is deleted.
`exx-reliable`	IN	Possible values: `AUTO-COMMIT` | `OFF`. See *Reliable RPC for XML/SOAP Wrapper*.