Writing Simple Applications with the Java Wrapper

This chapter covers the following topics:

- Required Steps
- Java Wrapper Constructors
- Generated Java Wrapper Methods

Required Steps

Interaction with the Java Wrapper occurs through instantiating objects of different classes, invoking their methods and manipulating their inner state. The basic steps for writing a client are listed below. For details, see the examples delivered with EntireX (*Delivered Java Wrapper Examples*). Methods and properties to interact with the EntireX Broker are completely inherited from the EntireX Java ACI. The EntireX Java ACI also contains the class `RPCService` used as the superclass by the generated Java Wrapper classes.

Basic Steps:

- Instantiate a Broker object.

  One object instance represents one session to an EntireX Broker on your network. If you want to work with multiple EntireX Brokers or with multiple sessions, create one object for each session to an EntireX Broker.

- Use the Broker object to log the application on to EntireX Broker.
- Instantiate the generated Java Wrapper object (see *Java Wrapper Constructors*).
- Use the Java Wrapper methods (see *Generated Java Wrapper Methods*) to call the server programs and access their parameters.

Java Wrapper Constructors

Two constructors are available for the generated Java Wrapper class:

- public Example (Broker broker)
- public Example (Broker broker, String serverAddr)

public Example (Broker broker)

This constructor requires an instantiated Broker object only. The server address used is specified in the properties of the IDL file. Each generated Java Wrapper class has two public static String constants which contain the default values of the Broker and the server as set in the properties of the IDL file. For example:

```
public static final String DEFAULT_BROKERID = "localhost";
public static final String DEFAULT_SERVER = "RPC/SRV1/CALLNAT";
```

A Java Wrapper object using the default settings may be instantiated with the following coding:

```
Broker broker = new Broker(Example.DEFAULT_BROKERID, "UserId");
Example myExample = new Example(broker);
```

public Example (Broker broker, String serverAddr)

This constructor requires an instantiated Broker object and the server address. A Java Wrapper object can be instantiated with the following coding:

```
Broker broker = new Broker("localhost", "UserId");
Example myExample = new Example(broker, "RPC/MYRPC/CALLNAT");
```

Generated Java Wrapper Methods

EntireX Interface Object Version Information

To get the version information of the generated interface object, use the method `getStubVersion ()`. It is implemented in the RPC client and server interface objects. The method returns a version string.

Example:

```
"EntireX RPC for Java Interface Object Version=8.2.0, Patch Level=0"
```

Application Identification

The application identification is sent from the application 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 trace information.

For the Java Wrapper these values are:

- **Application name**
  `ANAME=Java Runtime`
- **Node name**
  `ANODE=<host name>`
- **Application type**
  `ATYPE=Java`
- **Version**
  `AVERS=8.2.0.0`

The application is allowed to set the application name with the method `Broker.setApplicationName(String)`.

See `setApplicationName(java.lang.String)` of class `Broker` for more information.