Writing Web Service Client Applications

This chapter covers the following topics:

- Web Service Clients
- Configuring Advanced Web Service Clients
- Example: Setting up an EntireX Client to Consume a Secured Web Service

---

Web Service Clients

EntireX, in conjunction with the Software AG Common Web Services Stack (WSS), provides development and runtime functionality to support EntireX RPC clients consuming (or calling) Web services. The relevant products parts are:

- IDL Extractor for WSDL to generate an XML/SOAP mapping from the service's WSDL.
- XML Mapping Editor to adapt the mapping file if necessary. See *XML Mapping Editor*.
- EntireX XML/SOAP RPC Server, acting as the EntireX Web service runtime, to deploy the XML mapping file into and perform the Web service call. See *Administering the EntireX XML/SOAP RPC Server* under UNIX | Windows.
- Web Services Stack client runtime, which handles the underlying SOAP, WS-Policy and message transport. See the separate Software AG Common Web Services Stack documentation.

For each Web service client that is deployed in XML/SOAP RPC Server, a special configuration is required. The default name of the configuration file is *entirex.xmlrpcserver.configuration.xml*. Example configuration:

```
...
<TargetServer name="http://localhost:10010/wsstack/services/example">
  <xmms>
    <exx-xmm
      name="C:\MyWorkspace\Example\example.xmm"
      wsdl="http://localhost:10010/wsstack/services/example?wsdl"
      service="example"
      port="EXAMPLESOAP11Port"
      soapVersion="1.1"
      repository="C:\SoftwareAG\WS-Stack\repository" />
  </xmms>
</TargetServer>
...
```

where	`code`	is the XMM mapping file for the service
	`wsdl`	is the reachable URL for the WSDL file of the service. This WSDL file can contain additional WS-Policy information for the service that is supported by the Web Services Stack
	`service`	is the service name inside the WSDL file of the service
	`port`	is the port inside the WSDL file. This information is needed when the WSDL file can contain more than one port. The value in this example is the default number; this number can be changed during installation
	`soapVersion`	can be "1.1" or "1.2"
	`repository`	is the client "repository" of the Web Services Stack (containing the *conf* and *modules* subfolders)

Configuring Advanced Web Service Clients

A Web Services Stack client using advanced Web services functionality (WS-Security, WS-ReliableMessaging, etc.) requires the following configuration data:

- A "repository" containing configuration files and extension modules (.mar files). The "repository" is a folder containing subfolders *conf* and *modules*.
- The *conf* folder contains the client's global configuration file *axis2.xml* for the Web Services Stack engine.
- The modules folder contains modules for WS-* extensions, for example
  - addressing-1.4.mar if WS-Addressing is used
  - rampart-1.4.mar if WS-Security is used
  - rahas-1.4.mar if WS-Trust is used
- If WS-Security is used, an additional security configuration file *wsclientsec.properties* is required. The name and location of this file can be configured in the client's global configuration file *axis2.xml*, using the `securityConfigFile` property.

Here is an example of a client security configuration file *wsclientsec.properties*:

```
USERNAME=client
ENCRYPTION_USER=service
PASSWORD_CALLBACK_HANDLER_CLASS=com.softwareag.wsstack.test.PasswordCallbackHandler
KEYSTORE_FILE_ENCRYPT=client.jks
KEYSTORE_TYPE_ENCRYPT=jks
KEYSTORE_PASSWORD_ENCRYPT=apache
KEYSTORE_FILE_SIGN=client.jks
KEYSTORE_TYPE_SIGN=jks
KEYSTORE_PASSWORD_SIGN=apache
KEYSTORE_SSL_LOCATION=clientKS.jks
SSL_KEYSTORE_TYPE=jks
SSL_KEYSTORE_PASSWORD=apache
TRUSTSTORE_SSL_LOCATION=clientKS.jks
TRUSTSTORE_SSL_TYPE=jks
TRUSTSTORE_SSL_PASSWORD=apache
```

The USERNAME specifies the user name that the Web service client uses to authenticate itself with the Web service. It corresponds to Alias as described for the service configuration. ENCRYPTION_USER and PASSWORD_CALLBACK_HANDLER correspond accordingly. The example Java password callback handler from above can also be used for the client. The Web Services Stack provides some default password callback handlers that can be instantly used without needing to write a custom one. For example com.softwareag.wsstack.pwcb.ConfigFilePasswordCallbackHandler, which uses a simple user configuration file users.xml. See the separate WS-stack documentation for more details.

A Web Services Stack client that connects to a Web Service that requires advanced Web Services policies (which are attached to the service's WSDL as policy attachment) automatically sets up and processes the necessary SOAP headers in the SOAP message exchange with the service and fills the required parameters according to the configuration information described above.

Example: Setting up an EntireX Client to Consume a Secured Web Service

This section describes how to set up EntireX RPC clients calling a remote Web service that has a WS-Security UsernameToken policy in effect. Two scenarios are described: one where the security policy is defined in the WSDL, and one where the policy is *not* defined.

- Setting up an EntireX RPC Server to Configure WS-Security
- Scenario 1: Service requires UsernameToken and has a Security Policy in the WSDL
- Scenario 2: Service requires UsernameToken but does not declare this in the WSDL

Setting up an EntireX RPC Server to Configure WS-Security

To set up a dedicated XML/SOAP RPC Server instance to connect EntireX RPC clients to a secured Web service, the following prerequisites apply, for example in a folder of their own in the file system:

- A startup script, jxmlserver.bat. You can copy this from <*Install-Dir*>\EntireX\bin and adapt it.
- Property file and config file, entirex.xmlrpcserver.properties and entirex.xmlrpcserver.configuration.xml. You can copy these from <Install-Dir>\EntireX\conf and adapt them.
- A WSS client repository containing the subfolders conf, modules and services. You can copy this from <*Install-Dir*>\WS-Stack\repository and adapt it

  Note:
  For this example only the addressing and rampart modules are required; delete the others.
- A WSS client security configuration properties file, wsclientsec.properties, containing at least values for USERNAME and PASSWORD_CALLBACK_HANDLER_CLASS. You can copy this from <*Install-Dir*>\EntireX\bin and adapt it.

Scenario 1: Service requires UsernameToken and has a Security Policy in the WSDL

In this scenario, the Web Services Stack runtime can use the WS-Security policy from the WSDL to determine which security headers need to be attached to the SOAP message. Follow the steps below:

To set up an XML/SOAP RPC server with defined security policy

1. Store a copy of the service's WSDL (which also includes the policy attachment) in the test folder.
2. Generate an XML/SOAP mapping file (.xmm) with the IDL Extractor for WSDL. Enter the name of the XML/SOAP RPC Server ("XMLSERVER" in this example) under **Broker Settings** on the wizard page.
3. Start the XML/SOAP RPC Server in a command window, using the `start` script.
4. Deploy the mapping file to the XML/SOAP RPC Server. Provide the location of the WSDL and specify the desired service endpoint, name and port.
5. Configure the Web Services Stack repository for the service, using the following steps:
   1. Stop the XML/SOAP RPC Server.
   2. Edit the file `entirex.xmlrpcserver.configuration.xml` and add the repository definition to the `TargetServer` section. For example:

      ```
      <TargetServer
       name="http://localhost:10010/wsstack/services/example.EXAMPLESOAP11Port/">
        <xmms>
          <exx-xmm
            name="D:\TestWS\example.xmm"
            port="EXAMPLESOAP11Port"
            wsdl="D:\TestWS1\example.wsdl"
            service="example"
            soapVersion="1.1"
            repository="repository"/>
        </xmms>
      </TargetServer>
      ```

   3. Restart the XML/SOAP RPC Server.
6. Configure security for the WSS client runtime by modifying files `wsclientsec.properties` and `users.xml`:
   - File `wsclientsec.properties`, containing the lines

     ```
     USERNAME=user
     PASSWORD_CALLBACK_HANDLER_CLASS=
       com.softwareag.wsstack.pwcb.ConfigFilePasswordCallbackHandler
     ```

     Specify the desired `username`, which should go into the `UsernameToken`. The password callback handler class is used by the WSS client runtime to inquire a password for this user. The `ConfigFilePasswordCallbackHandler` is a simple default handler delivered with Web Services Stack that reads the password of a given user from a flat file `users.xml`. You can write a custom password callback handler for other methods to acquire passwords.

- File `users.xml`. Example:

```
<?xml version="1.0" encoding="UTF-8"?>
<users>
  <user username="user" password="pass" />
  <user username="client" password="apache" />
  <user username="service" password="apache" />
  <user username="bob" password="bobPW" />
</users>
```

To test access to the remote Web service, use the XML Tester on the IDL file. See *XML Tester*.

Scenario 2: Service requires UsernameToken but does not declare this in the WSDL

For this scenario, perform the steps as described above. Because the WSDL does not contain a security policy stating that `UsernameToken` is required, perform this additional step:

- Explicitly tell the Web Services Stack client runtime about the `UsernameToken` required by the service. Edit *<SuiteInstallDir>/profiles/CTP/workspace/wsstack/repository/conf/axis2.xml*, uncomment the `rampart` module and add the `OutFlowSecurity` parameters:

```
<module ref="rampart"/>
<parameter name="OutflowSecurity">
  <action>
    <items>UsernameToken</items>
    <user>user</user>
    <passwordType>PasswordText</passwordType>
    <passwordCallbackClass>
    com.softwareag.wsstack.pwcb.ConfigFilePasswordCallbackHandler
    </passwordCallbackClass>
  </action>
</parameter>
```

  where *`user`* is a valid user name for authentication.