

Writing Advanced Applications with the XML/SOAP Wrapper

This chapter covers the following topics:

- XML/SOAP Listener
 - Natural Logon or Changing the Library Name
 - Using RPC Compression
 - Using Conversational RPC
 - Using Natural Security
 - Using Compression
 - Using EntireX Security
 - HTTP Proxy Settings
 - XML/SOAP RPC Server with HTTP Basic Authentication
 - XML/SOAP Listener with HTTP Basic Authentication and UsernameToken Authentication for EntireX Authentication
 - Using SSL or TLS with the XML/SOAP RPC Server
 - Using Internationalization with EntireX XML Components
 - Null Value Suppression
 - User-specified Settings
 - Map Fault to IDL Parameter
 - Whitespace Handling
-

XML/SOAP Listener

With the XML/SOAP Listener you can define parameters inside the payload of a message. We recommend this approach rather than HTTP parameters. Define the setting in the SOAP header and under the first tag of XML document as follows:

SOAP Documents

```
...
<soap-env:SOAPHeader>
  <exx:EntireX xmlns:exx="urn:com.softwareag.entirex.xml.rt">
    <!--tags with parameter setting e.g: -->
    <exx-natural-library>libraryname</exx-natural-library>
    <exx-natural-security>true</exx-natural-security>
```

```
    </exx:EntireX>
...
</soap-env:SOAPHeader>
...
```

XML Documents

```
<root-tag>
  <exx:EntireX xmlns:exx="urn:com.softwareag.entirex.xml.rt">
    <!--tags with parameter setting e.g: -->
    <exx-natural-library>libraryname</exx-natural-library>
    <exx-natural-security>true</exx-natural-security>
  </exx:EntireX>
...
</root-tag>
```

Natural Logon or Changing the Library Name

The library name sent with the RPC request to the EntireX RPC or the Natural RPC Server is specified in the IDL file. See `library-definition` under *Software AG IDL Grammar*. When the RPC is executed, this library name can be overwritten.

XML/SOAP Wrapper (Java API)

> To overwrite the library

- An EntireX XML/SOAP Wrapper client (Java API) must call the `setLibraryName` method.

> To force the library to be considered by Natural RPC Servers

- Call the `setNaturalLogon` method.

XML/SOAP Listener

> To overwrite the library

- Use the parameter `exx-natural-library`.

> To force the library to be considered by Natural RPC Servers

- Set the parameter `exx-natural-security` to "true".



Warning:
Natural RPC Servers and EntireX RPC Servers behave differently regarding the library name.

See also *Natural Logon or Changing the Library Name* under *Common Features of Wrappers and RPC-based Components*.

Using RPC Compression

EntireX and Natural RPC support a feature called RPC compression to reduce network traffic. The default for compression is on. See also *RPC Compression* under *Common Features of Wrappers and RPC-based Components*.

XML/SOAP Wrapper (Java API)

> To switch compression on and off

- Use the `setCompression` method of the class `XMLRPCService` inherited from class `RPCService`.

> To check the current compression setting

- Use the `getCompression` method of the class `XMLRPCService` inherited from class `RPCService`.

XML/SOAP Listener

> To switch compression on and off

- Use the parameter `exx-compression`. Possible values: True, False.

Using Conversational RPC

It is assumed that you are familiar with the concepts of conversational and non-conversational RPC. See also *Conversational RPC* under *Common Features of Wrappers and RPC-based Components*.

For conversational RPC you need an instantiated conversation object. See `Conversation`. Conversational RPC is enabled by passing a reference to this object to the method `setConversation`. See `setConversation`. Different stubs can participate in the same conversation if they use the same instance of a `Conversation` object. An RPC conversation is terminated by calling either the `closeConversation` method or the `closeConversationCommit` method for one stub.

XML/SOAP Wrapper (Java API)

> To enable conversational RPC

- Create a `Conversation` object and set this with `setConversation` on the wrapper object.

Different wrapper objects can participate in the same conversation if they use the same instance of a conversation object.

> To abort a conversational RPC communication

- Call the `closeConversation` method.

> To close and commit a conversational RPC communication

- Call the `closeConversationCommit` method.

XML/SOAP Listener

Conversations can only be used in connection with sessions. If the session is interrupted, the conversation will be deleted.

➤ To use conversational RPC

- Use the parameter `exx-conv` with the value OPEN.

➤ To continue conversational RPC

- Pick up the parameter `exx-sessionID` in response and set the parameter as HTTP parameter or in the same way as in the response document inside the request document.

➤ To abort a conversational RPC communication

- Use the parameter `exx-conv` with the value BACKOUT.

➤ To close and commit a conversational RPC communication

- Use the parameter `exx-conv` with the value COMMIT.

See also *XML Tester for Conversational RPC*.



Warning:
Natural RPC Servers and EntireX RPC Servers behave differently when ending an RPC conversation.

See also *Conversational RPC* under *Common Features of Wrappers and RPC-based Components*.

Using Natural Security

A Natural RPC Server may run under Natural Security to protect RPC requests. See also *Natural Security* under *Common Features of Wrappers and RPC-based Components*.

XML/SOAP Wrapper (Java API)

➤ To authenticate an EntireX XML/SOAP Wrapper client (Java API) against Natural Security

- Specify a user ID and password in the `logon` method of class `Broker`.

If different user IDs and/or passwords are used for EntireX Security and Natural Security, use the methods `setRPCUserId` or `setRPCPassword` to set the user IDs and/or passwords for Natural Security.

➤ To force an EntireX XML/SOAP Wrapper client (Java API) to log on to a specific Natural library

1. Call the `setLibraryName` method.
2. Call the `setNaturalLogon` method.

See also *Natural Logon or Changing the Library Name*.

XML/SOAP Listener

➤ To authenticate against Natural Security

- Specify the parameters `exx-userID` and `exx-password`.

If a different user ID or password is used for EntireX Security and Natural Security, use the parameters `exx-rpc-userID` and `exx-rpc-password` to set the user ID or password for Natural Security.

➤ To force a logon to a specific Natural library

1. Use the parameter `exx-natural-library`.
2. Set the parameter `exx-natural-security` to `True`.

See also *Natural Logon or Changing the Library Name*.

Using Compression

Java-based EntireX applications (including applications using classes generated by the Java Wrapper) may compress the messages sent to and received from the broker. There is a general way to enable compression using broker ID, and another way that depends on whether you use the XML/SOAP Wrapper or the XML/SOAP Listener.

- Using Broker ID
- XML/SOAP Wrapper (Java API)
- Using `setCompressionLevel()`
- XML/SOAP Listener

Using Broker ID

You may append the keyword `compresslevel` with one of the values below to the Broker ID.

Examples

- `localhost:1971?compresslevel=BEST_COMPRESSION`
- `localhost?poolsize=4&compresslevel=9`

Both examples set the compression level to 9.

XML/SOAP Wrapper (Java API)

Using `setCompressionLevel()`

Set the compression level with the method `setCompressionLevel()` as an integer or a string argument.

You can use the constants defined in class `java.util.zip.Deflater`.

If the string

- starts with Y, compression is switched on with level 6,
- starts with N, compression is switched off (level 0).

Permitted values are the integers 0 - 9 and the corresponding strings:

Compression	Level
BEST_COMPRESSION	9
BEST_SPEED	1
DEFAULT_COMPRESSION	6
DEFLATED	8
NO_COMPRESSION	0

XML/SOAP Listener

> To set the compression level

- Use the parameter `exx-compressLevel`. The values are described in the section above (*XML/SOAP Wrapper (Java API)*).

Using EntireX Security

Java-based EntireX applications that require security can use the security services offered by EntireX Security. See also *Overview of EntireX Security*

Use the methods for security, which are included in class `Broker`. See `Broker`. The two alternatives using security are:

- using EntireX Security
- using your own security implementation

XML/SOAP Wrapper (Java API)

To use EntireX Security, call `Broker.useEntireXSecurity()` for a `Broker` object. You can set the encryption level with this call and you can enable the auto mode. The encryption level allows the values `ENCRYPTION_LEVEL_NONE`, where the message is not encrypted, `ENCRYPTION_LEVEL_BROKER`, where the message is encrypted on the way to the EntireX Broker, and `ENCRYPTION_LEVEL_TARGET`,

where the message is encrypted the whole way to the target. The auto mode specifies that the Broker object uses the EntireX Security as needed by the EntireX Broker. If the EntireX Broker uses security, the EntireX Security object is used by the Broker object. The method `useEntireXSecurity()` must be called before the first call of `logon()`, which has to use a password. The security object cannot change during a session with the EntireX Broker.

To use your own security implementation, implement the interface `BrokerSecurity`. This implementation must have an accompanying security exit for the EntireX Broker. See *Using Sample Security Exits for Broker Security*. Call the methods `setSecurity()` with the security object and set encryption level or auto mode in the same way as the `useEntireXSecurity()` methods.

XML/SOAP Listener

The parameter `exx-use-security` (true, false) is responsible for EntireX Security. Set the encryption level with the required parameter `exx-encryption-level` (0,1,2).

HTTP Proxy Settings

If the target server of your Web service has to be reached through a firewall, set and adjust to your needs the following properties:

- `-Dhttp.proxySet=true`
- `-Dhttps.proxySet=true`
- `-Dhttp.proxyHost=httpprox.mydomain.org`
- `-Dhttps.proxyHost=sslprox.mydomain.org`
- `-Dhttp.proxyPort=8080`
- `-Dhttps.proxyPort=443`
- `-Dhttp.nonProxyHosts=*mydomain.org|localhost`
- `-Dhttps.nonProxyHosts=*mydomain.org|localhost`
- `-Dhttp.proxyUser`
- `-Dhttps.proxyUser`
- `-Dhttp.proxyPassword`
- `-Dhttps.proxyPassword`

Add the proxy settings to the start script.

XML/SOAP RPC Server with HTTP Basic Authentication

The XML/SOAP RPC Server uses basic authentication for a Web service if the configuration contains the attribute `basicAuthentication` block in `<TargetServer>`. Basic authentication is used for all calls associated with defined XMM files for the `<TargetServer>`.

Basic authentication can be used with fixed credentials or credentials set from the client application:

- If `<TargetServer>` contains attributes `user` and `password`, these settings are used for basic authentication.
- Otherwise the client application must provide the credentials: Enable Natural logon and set RPC user ID and RPC password.

See *Configuration File for the XML/SOAP RPC Server* under UNIX | Windows.

XML/SOAP Listener with HTTP Basic Authentication and UsernameToken Authentication for EntireX Authentication

The XML/SOAP Listener allows you to use the user credentials from the incoming request by means of Basic Authentication or UsernameToken. The same credentials are used for EntireX Broker authentication and (Natural) RPC Server authentication. This means you need to make some settings for the EntireX Web service in Web Service Wizard and Configuration Editor.

Note:

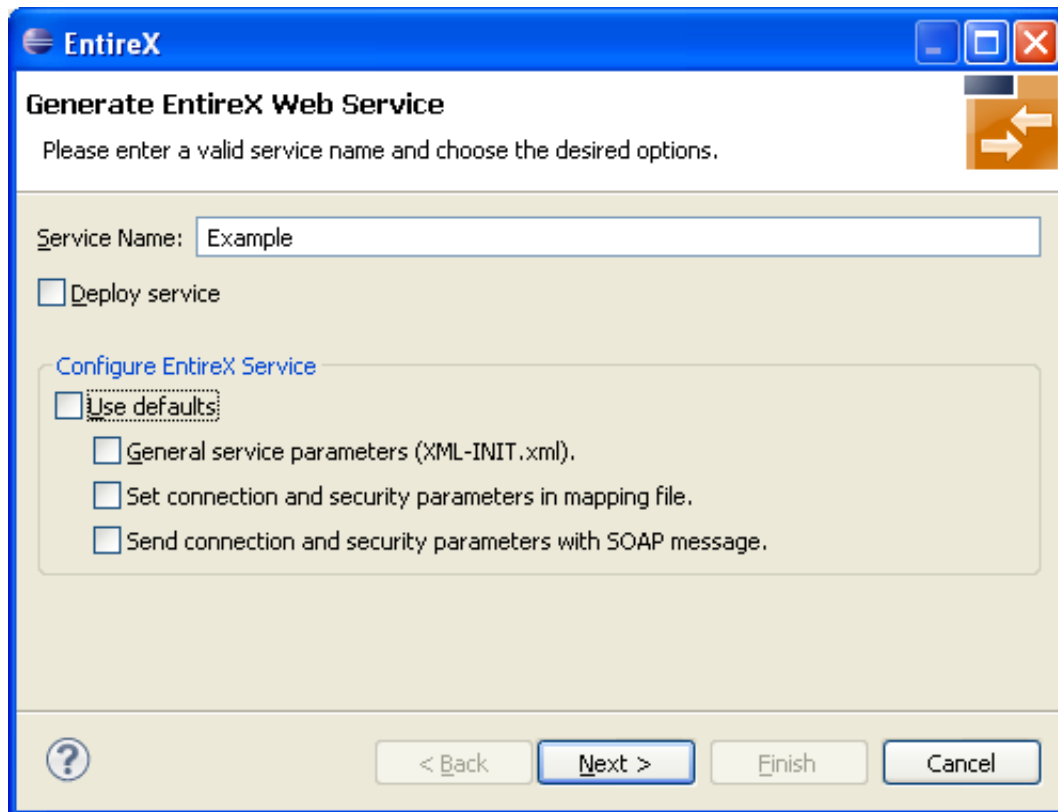
UsernameToken is part of WS-Security. See WS-Security UsernameToken Specification. See also *Example: Setting up an EntireX Client to Consume a Secured Web Service* in the IDL Extractor for WSDL documentation.

The priority of credentials settings is as follows:

1. `exx-userID`, `exx-password`, `exx-rpc-userID`, `exx-rpc-password` (highest priority)
2. UsernameToken
3. Basic Authentication (lowest priority)

➤ To use the XML/SOAP Listener with Basic Authentication and UsernameToken Authentication

1. Select an IDL file or XMM file.
2. Choose **Web Service > Generate Web Service...**
3. Disable check box **Use Defaults**.



EntireX

Generate EntireX Web Service

Please enter a valid service name and choose the desired options.

Service Name:

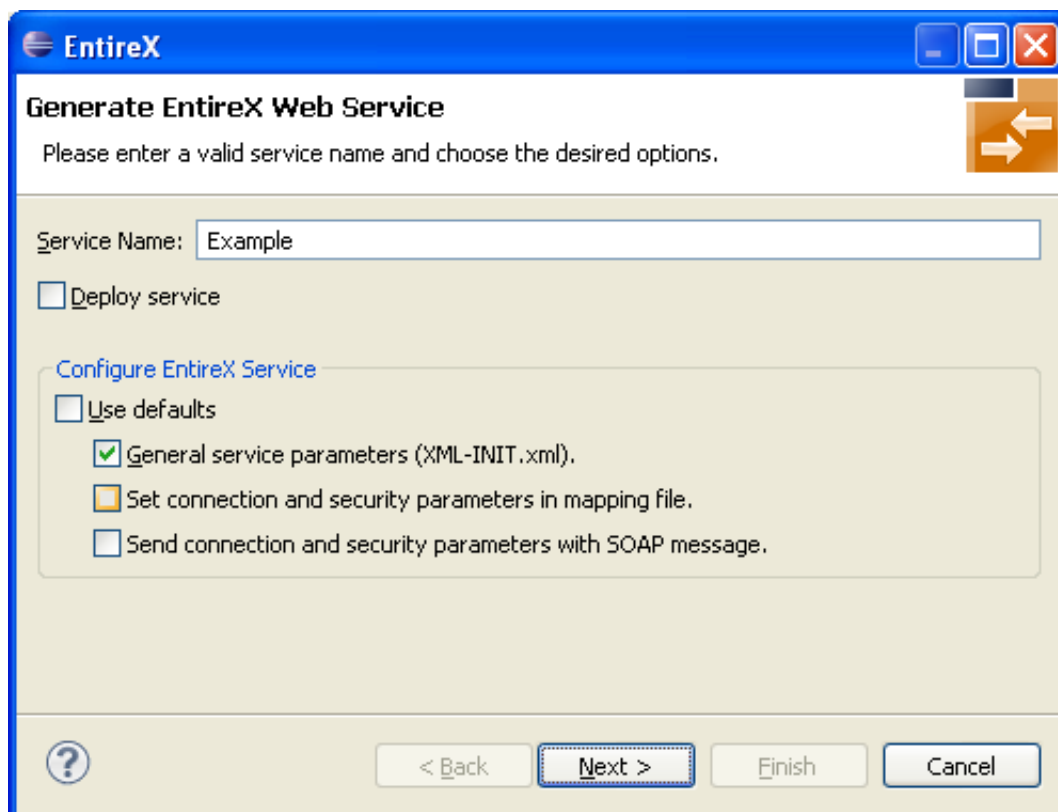
Deploy service

Configure EntireX Service

Use defaults

- General service parameters (XML-INIT.xml).
- Set connection and security parameters in mapping file.
- Send connection and security parameters with SOAP message.

4. Enable at least **General service parameters...**



EntireX

Generate EntireX Web Service

Please enter a valid service name and choose the desired options.

Service Name:

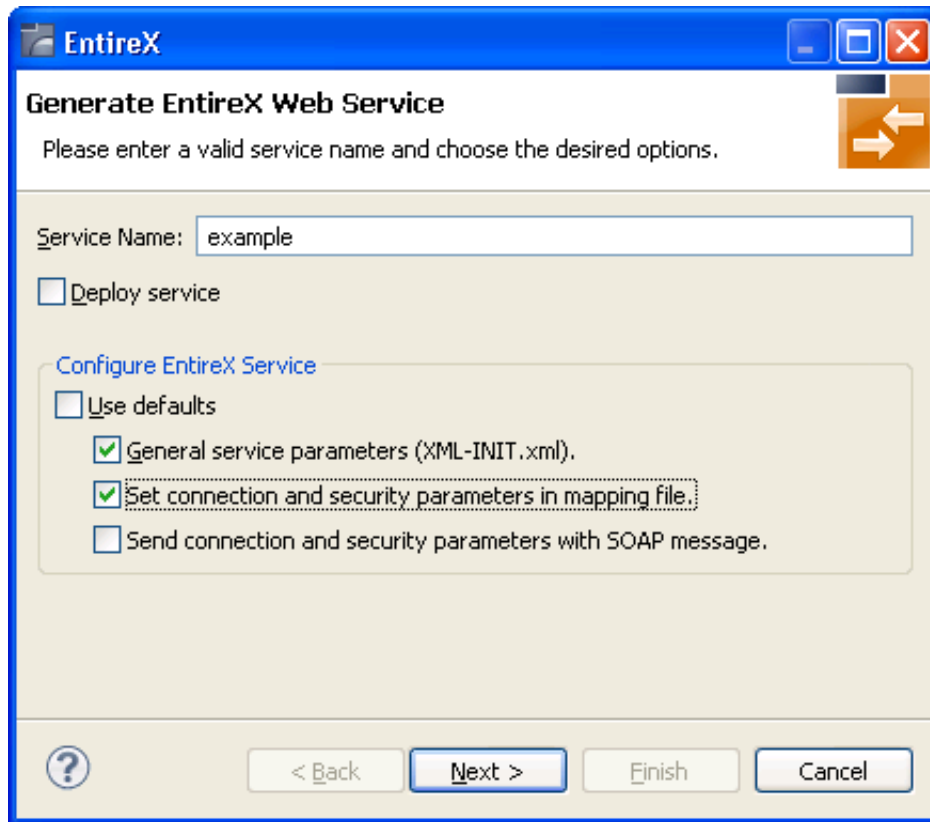
Deploy service

Configure EntireX Service

Use defaults

- General service parameters (XML-INIT.xml).
- Set connection and security parameters in mapping file.
- Send connection and security parameters with SOAP message.

5. If using EntireX Security or Natural Security, enable **Set connection and security...** too.



6. Press **Next**.
7. Enable the required authentication. In this example, both possibilities of web service authentication are enabled.

EntireX

Generate EntireX Web Service

Please configure xml-init.xml for the service.

Location Transparency Config:

Init Location Transparency:

Default Wait Time:

Behaviour of Non-Conversation Calls: nonConv-with-logoff

User Name Token: enable

Basic Authentication: enable

? < Back Next > Finish Cancel

8. Press **Next**.
9. The page with XMM settings appears if it was selected before (step 5). Enable the required security (EntireX Security and/or Natural Logon).

EntireX

Generate EntireX Web Service

Please configure the Mapping file.

File: example.xmm

Broker ID: localhost:1971

Server Address: RPC/SRV1/CALLNAT

NATURAL Library:

RPC User ID:

RPC Password:

User ID:

Password:

Use Security: (not used)

NATURAL Logon: (not used)

Encryption level: (not used)

Compression Level: (not used)

Logical Broker ID:

Logical Service:

Logical Set Name:

Use Codepage:

< Back Next > Finish Cancel

10. Press **Next** and follow the wizard.

11. After generating the web service archive (extension "aar"), open the generated AAR file with the Configuration Editor (e.g. with double click).

For more information on the Configuration Editor see *Configuring Web Services*.

Using SSL or TLS with the XML/SOAP RPC Server

Using HTTPS with XML/SOAP RPC Server requires setting Java properties and changing the protocol from http to https in the configuration file. This section covers the following topics:

- SSL or TLS Settings
- Sample Start Script
- Configuration File Settings

See also *Configuration File for the XML/SOAP RPC Server* under UNIX | Windows.

SSL or TLS Settings

➤ To configure SSL communication for the JRE

- Set the following properties:
 - **-Djavax.net.ssl.keyStore=<filename-without-blanks>**
Here we keep the certificate and the private signing key of our client application, which is the EntireX XML/SOAP RPC Server.
 - **-Djavax.net.ssl.keyStorePassword=<you-should-know-it>**
The password that protects the keystore.
 - **-Djavax.net.ssl.keyStoreType=pkcs12**
If not jks (default).
 - **-Djavax.net.ssl.trustStore=<filename-without-blanks>**
Here we keep the trusted certificate of the Web service host or the certificate of its signing (issuing) certificate authority.
 - **-Djavax.net.ssl.trustStorePassword=<you-should-know-it>**
The password that protects the truststore.
 - **-Djavax.net.ssl.trustStoreType=**
If not jks (default).

For more information about Java and SSL, see your Java documentation (JSSE documentation).

Sample Start Script

```
set CLASSPATH=.;\classes\entirex.jar;..\WS-Stack\lib\wsstack-client.jar

set PROXYSETTINGS=-Dhttps.proxySet=true
-Dhttps.proxyHost=sslproxy.mydomain
-Dhttps.proxyPort=443
-Dhttps.nonProxyHosts="localhost"

set SSL=-Djavax.net.ssl.keyStore=C:\myKeystore.p12
-Djavax.net.ssl.keyStorePassword=myKeystorePassword
-Djavax.net.ssl.keyStoreType=pkcs12
-Djavax.net.ssl.trustStore=C:\myTrustStore.jks
-Djavax.net.ssl.trustStorePassword=myTruststorePassword

java -classpath %CLASSPATH% %SSL% %PROXYSETTING% com.softwareag.entirex.xml.rt.XMLRPCServer
```

For the changes that are required to the start script, see your Java documentation (JSSE documentation).

Configuration File Settings

Specify the fully qualified host name as TargetServer. The host name has to match the CN (Common Name) item of the host certificate.

```
<?xml version="1.0" encoding="iso-8859-1" ?>
<EntireX
xmlns="http://namespaces.softwareag.com/entirex/xml/runtime/configuration" version="8.0"
>
  <XmlRuntime Version="1">
    <TargetServer name="https://targethost:8080/entirex/xmlrt">
      <xmms>
        <exx-xmm name="yourFile1.xmm" />
        <exx-xmm name="yourFile2.xmm" />
      </xmms>
    </TargetServer>
  </XmlRuntime>
</EntireX>
```

Using Internationalization with EntireX XML Components

XML components support Conversion and Translation for Internationalization. If you choose Conversion (which is recommended), the correct codepage must be sent as locale string to the EntireX Broker matching the encoding of the data sent. This codepage must also be a codepage supported by the broker, see *Locale String Mapping* for information on how the broker derives the codepage from the locale string. For Translation and more details on Conversion, see *Internationalization with EntireX*.

To enable EntireX XML components to send a codepage as locale string to the EntireX Broker, they must be prepared as described below:

- XML/SOAP Wrapper (Java API)
- XML/SOAP Listener
- XML/SOAP RPC Server

XML/SOAP Wrapper (Java API)

➤ **To enable use of the encoding of an incoming XML document for Broker communication**

1. Call `useCodePage(true)` on the `XMLRPCService` object. The XML/SOAP Wrapper will then use the codepage retrieved from the XML document to send data to EntireX Broker.
2. Use a stream-oriented method of `XMLRPCService` to transfer the data to XML/SOAP Wrapper.

XML/SOAP Listener

➤ **To enable use of the encoding of an incoming XML document for broker communication**

- Activate the parameter `exx-use-codepage` (true/false). The XML/SOAP Wrapper will then use the codepage retrieved from the incoming XML document to send data to the EntireX Broker.

XML/SOAP RPC Server

The encoding for broker communication is defined by the parameter `codepage`.

- The locale string for broker communication can be overridden for a broker version 7.2.x and above. Instead of using the default encoding of the JVM, the given encoding is used.
- It can be forced for a broker version 7.1.x and below.
- It does not change the default encoding of your Java virtual machine.
- We recommend using UTF-8 as the file encoding for your JVM and the value `LOCAL` to send the default encoding of the JVM to the broker, i.e start the XML/SOAP RPC Server with `-Dcodepage=LOCAL` and `-Dfile.encoding=utf-8`. See also *Using the Abstract Codepage Name LOCAL* for more information.

The encoding for the outgoing XML document is determined by the IDL to XML Mapping. See *Defining the XML Encoding*.

Null Value Suppression

This section covers the following topics:

- Introduction
- Default Setting for Null Value Suppression
- Definition and Examples of Null Value Suppression Mode
- Default Definition of Null Value

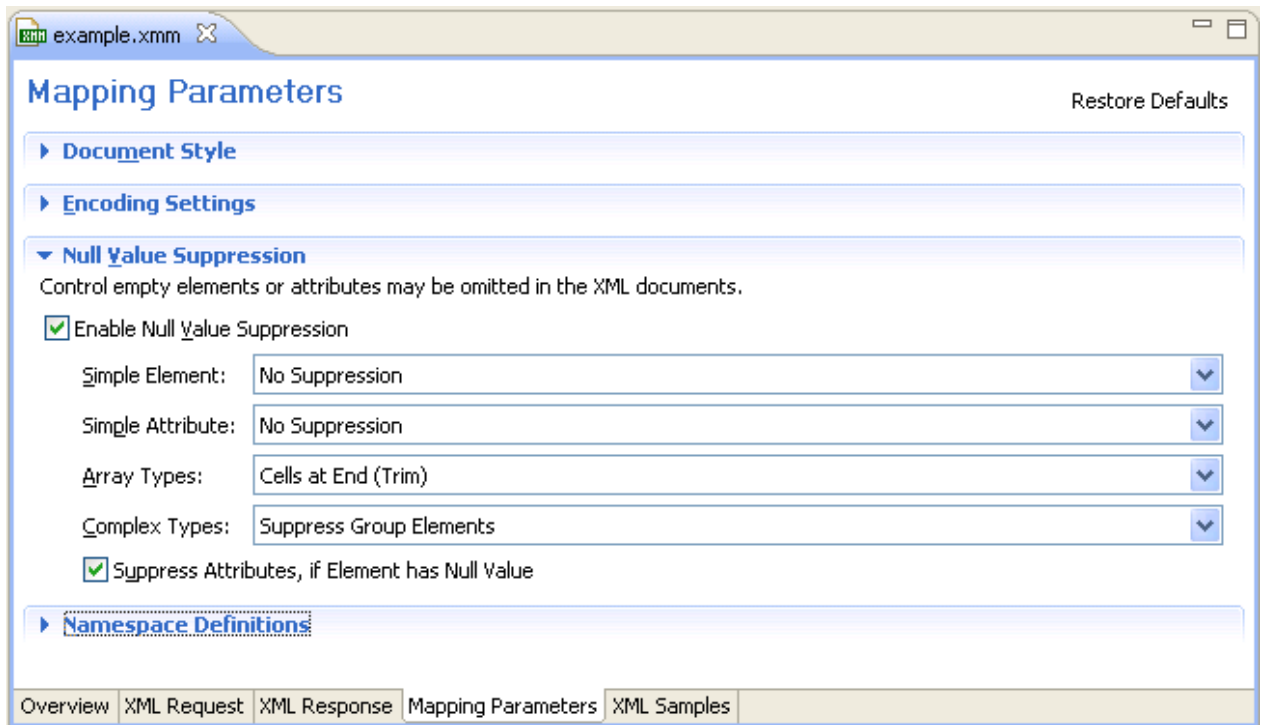
Introduction

The EntireX XML/SOAP Runtime introduced a feature called *null value suppression* to reduce to amount of data transferred between Web client and Web service. Null value suppression (NVS) allows you to hide tags or attributes with a specified value (so-called null value).

The different types of NVS are explained below. The EntireX XML Mapping Editor provides two ways of setting/modifying the NVS type:

- using a general setting on tab **Mapping Parameter**
- on each element or attribute in the definition of request/response on tab **XML Request/ XML Response**

For several data types, a null value is defined by default. See *Default Definition of Null Value*. To change one of these null values, open the **XML Request/XML Response** tab, select the element/attribute and modify the property of the null value manually. For this modification use the **Properties** view or the **Properties** dialog (open with the context menu).



Default Setting for Null Value Suppression

Data Type	Suppression Mode
Simple Elements	No Suppression
Simple Attributes	No Suppression
Array Types	Cells at end (trim)
Complex Types	Suppress group elements

Tip:

The default setting for elements and attributes changed with version 7.3 from "Suppress Element/Attribute" to "No Suppression". The null value suppression for elements and attributes can be set independently. The null value suppression for Complex Types was introduced with version 7.3.

Definition and Examples of Null Value Suppression Mode

If there is a significant difference between pure XML and SOAP for a null value suppression mode, two examples are introduced.

Suppression Mode	Valid for
<i>No Suppression</i>	Elements, attributes
<i>Elements</i>	Elements
<i>Attributes</i>	Attributes
<i>Cells at End (Trim)</i>	Elements inside an array definition
<i>All Empty Cells</i>	Elements inside an array definition
<i>Suppress Group Elements</i>	Elements inside a group definition
<i>Depends On Element</i>	Attributes

No Suppression

The element or attribute is always present in document. The minimum and maximum occurrence of element/attribute must be set to one.

Example 1

XML document	Displayed XML Document
<pre><prog> <integer>0</integer> </prog></pre>	<pre><prog> <integer>0</integer> </prog></pre>

Elements

An element is hidden if

- the element value is equal to null value
- all attributes of the element can be suppressed
- the element has only subelements that can be suppressed

The minimum occurrence of element must be zero, and the maximum occurrence of element must be one.

Example 2

XML document	Displayed XML Document
<pre><prog> <gr> <integer>0</integer> </gr> </prog></pre>	<pre><prog /></pre>

Attributes

An attribute with null value is hidden.

The minimum occurrence of attribute must be zero, and the maximum occurrence of attribute must be one.

Example 3

XML document	Displayed XML Document
<code><prog integer="0" name="Henry" /></code>	<code><prog name="Henry" /></code>

Cells at End (Trim)

All elements of array that fulfills the assertion of "Suppress Element/Attribute" are suppressed if its index is higher than the highest index of the non-suppressed element.

The minimum occurrence of elements must be lower than the maximum occurrence, and the maximum occurrence of elements must be the maximum number of elements or unlimited.

XML document	Displayed XML Document
<pre> <prog> <array> <integer>0</integer> <integer>1</integer> <integer>0</integer> <integer>2</integer> <integer>0</integer> <integer>0</integer> </array> </prog> </pre>	<pre> <prog> <array> <integer>0</integer> <integer>1</integer> <integer>0</integer> <integer>2</integer> </array> </prog> </pre>

All Empty Cells

All elements of array that fulfills the assertion of NVS_FIELD are suppressed.

The minimum occurrence of elements must be lower than the maximum of occurrence and maximum occurrence of elements must be maximum number of elements or unlimited.

Example 5a

XML document	Displayed XML Document
<pre> <prog> <array> <integer>0</integer> <integer>1</integer> <integer>0</integer> <integer>2</integer> <integer>0</integer> <integer>0</integer> </array> </prog> </pre>	<pre> <prog> <array> <integer>1</integer> <integer>2</integer> </array> </prog> </pre>

Example 5b (for SOAP documents the XML/SOAP Runtime creates position attributes)

SOAP document	Displayed SOAP Document
<pre><prog> <array> <integer>0</integer> <integer>1</integer> <integer>0</integer> <integer>2</integer> <integer>0</integer> <integer>0</integer> </array> </prog></pre>	<pre><prog> <array> <integer SOAP-ENC:position="[1]">1</integer> <integer SOAP-ENC:position="[3]">2</integer> </array> </prog></pre>

Suppress Group Elements

The suppression mode allows you to suppress group information if - and only if - all elements of the group can be suppressed.

The minimum occurrence of elements must be zero.

Example 6

XML document	Displayed XML Document
<pre><prog> <person> <firstname>Henry</ firstname > <lastname>Miller</ lastname > <someInformation>2</ someInformation > </person> <person> <firstname></ firstname > <lastname></ lastname > <someInformation>0</ someInformation > </person> <person> <firstname>John</ firstname > <lastname>Miles</ lastname > <someInformation>0</ someInformation > </person> </prog></pre>	<pre><prog> <person> <firstname>Henry</ firstname > <lastname>Miller</ lastname > <someInformation>2</ someInformation > </person> <person/> <person> <firstname>John</ firstname > <lastname>Miles</ lastname > <someInformation>0</ someInformation > </person> </prog></pre>

Depends On Element

Attribute of the element is visible if the element does not have the null value.

The minimum occurrence of attribute and associated element must be zero, and the maximum occurrence must be one.

Example 7

XML document	Displayed XML Document
<pre><prog type="integer">0 <parm type="integer">0</parm> </prog></pre>	<pre><prog /></pre>

Default Definition of Null Value

Data Types	Null Value
String	Empty String
Integer	0
Floating Point	0.0
Numeric	0.0
Time	No default definition
Date	No default definition
Binary	No default definition
Boolean	False

User-specified Settings

For further settings, use the method `setUserProperty` in `XMLRPCService` (EntireX XML/SOAP Runtime).

Map Fault to IDL Parameter

- Introduction
- Example
- Testing the Fault Mapping

Introduction

The XML/SOAP RPC Server maps the values of IDL parameters to XML/SOAP documents and vice versa. If the Web service responds with a fault document, this information is mapped to an error and returned to RPC client normally. With the optional feature **Map Fault to IDL Parameter** you can map values from a normal response and also from a fault document response. This means that no RPC error is returned to the RPC client; instead the fault information is contained in the IDL file. An RPC error is returned to the client only if internal error processing problems occurred within the XML/SOAP RPC Server. This feature is available for SOAP and XML documents.

Example

Note:

This example illustrates the feature **Map Fault to IDL Parameter**. Other features mentioned here, such as renaming parameters or assigning a prefix/namespace to a parameter, are described elsewhere.

Sample IDL File

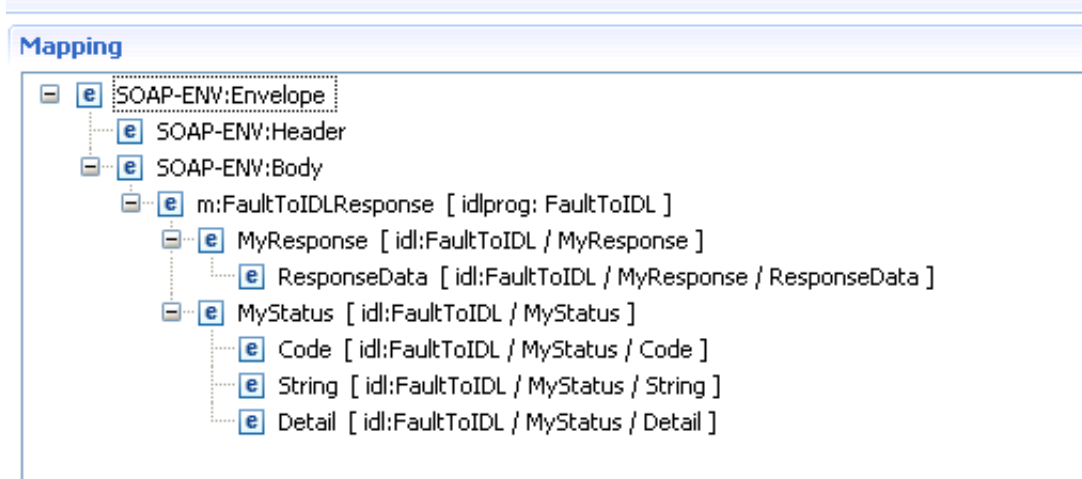
```
/* Sample IDL file
library 'Demo' is
  program 'FaultToIDL' is
    define data parameter
      1 MyRequest In
      2 RequestData (AV)
      1 MyResponse Out
      2 ResponseData (AV)
      1 MyStatus Out ** parameters for fault information
      2 Code (AV)
      2 String (AV)
      2 Detail (AV)
    end-define
```

IDL-XML Mapping

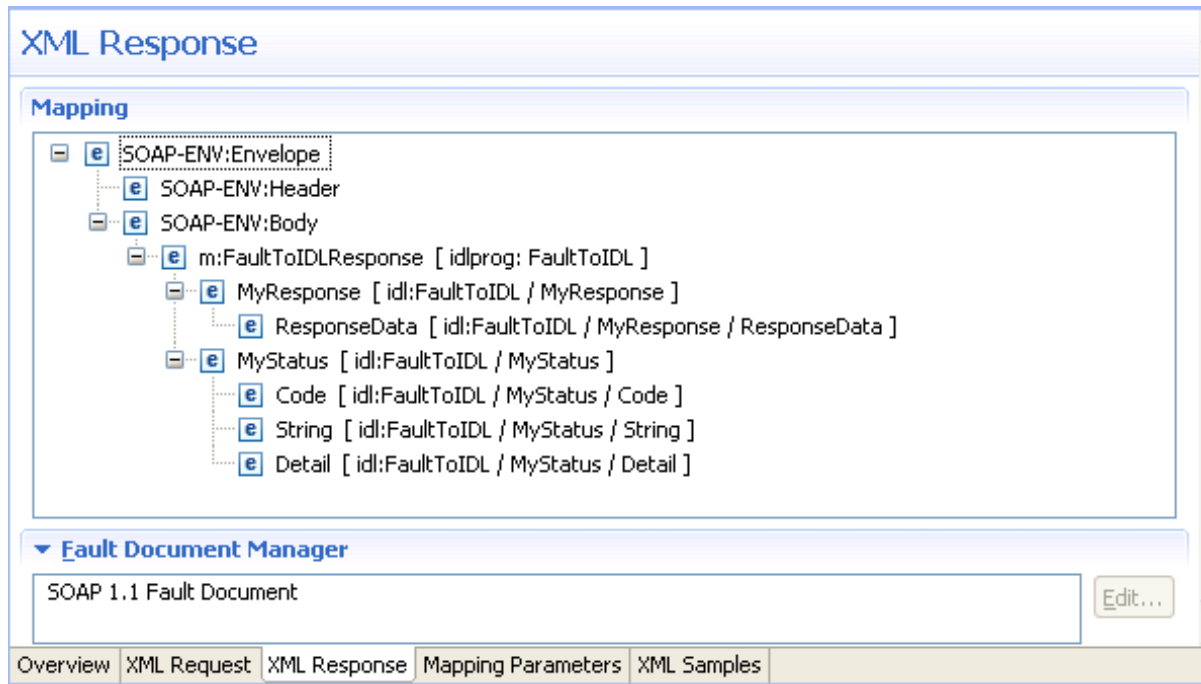
> To map fault items to IDL

1. In the XML Mapping Editor, generate a (SOAP) mapping and select the response tab.

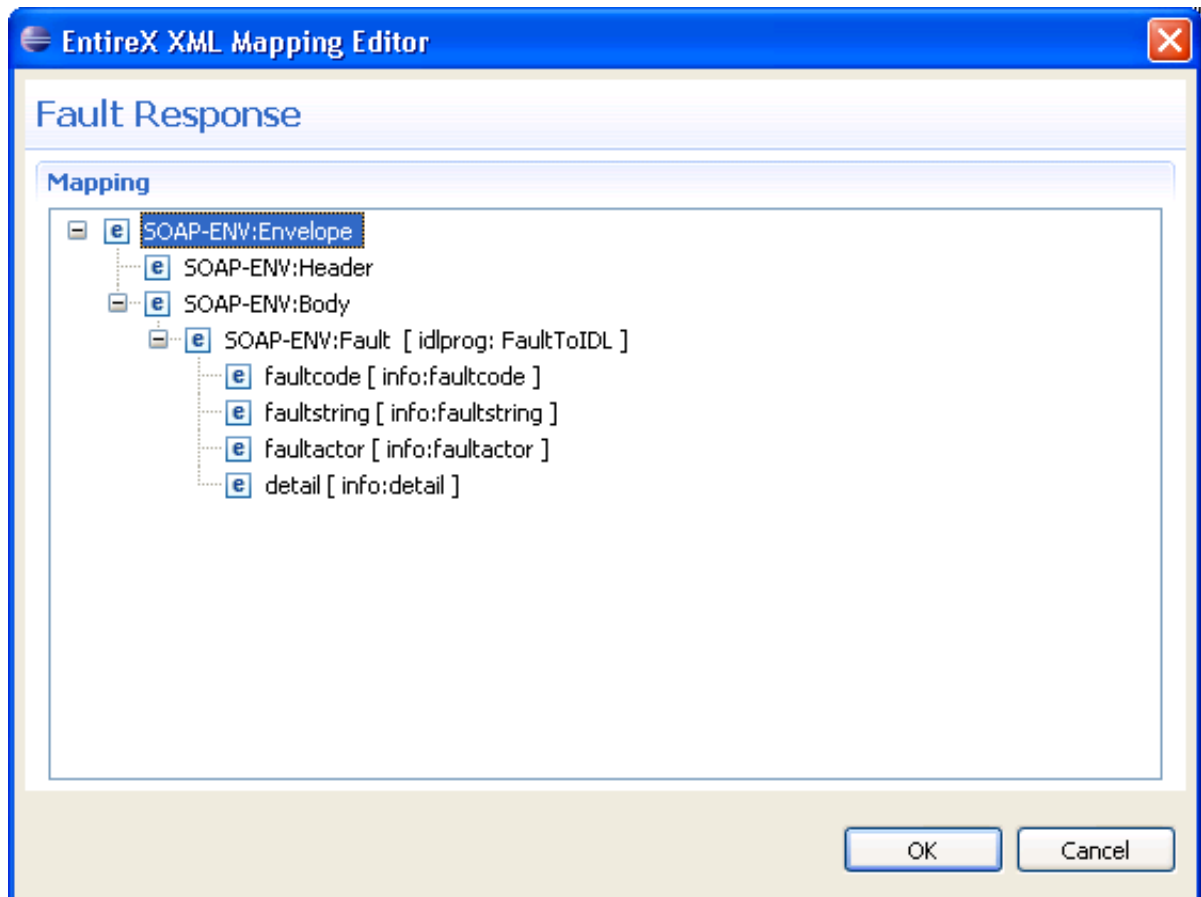
XML Response



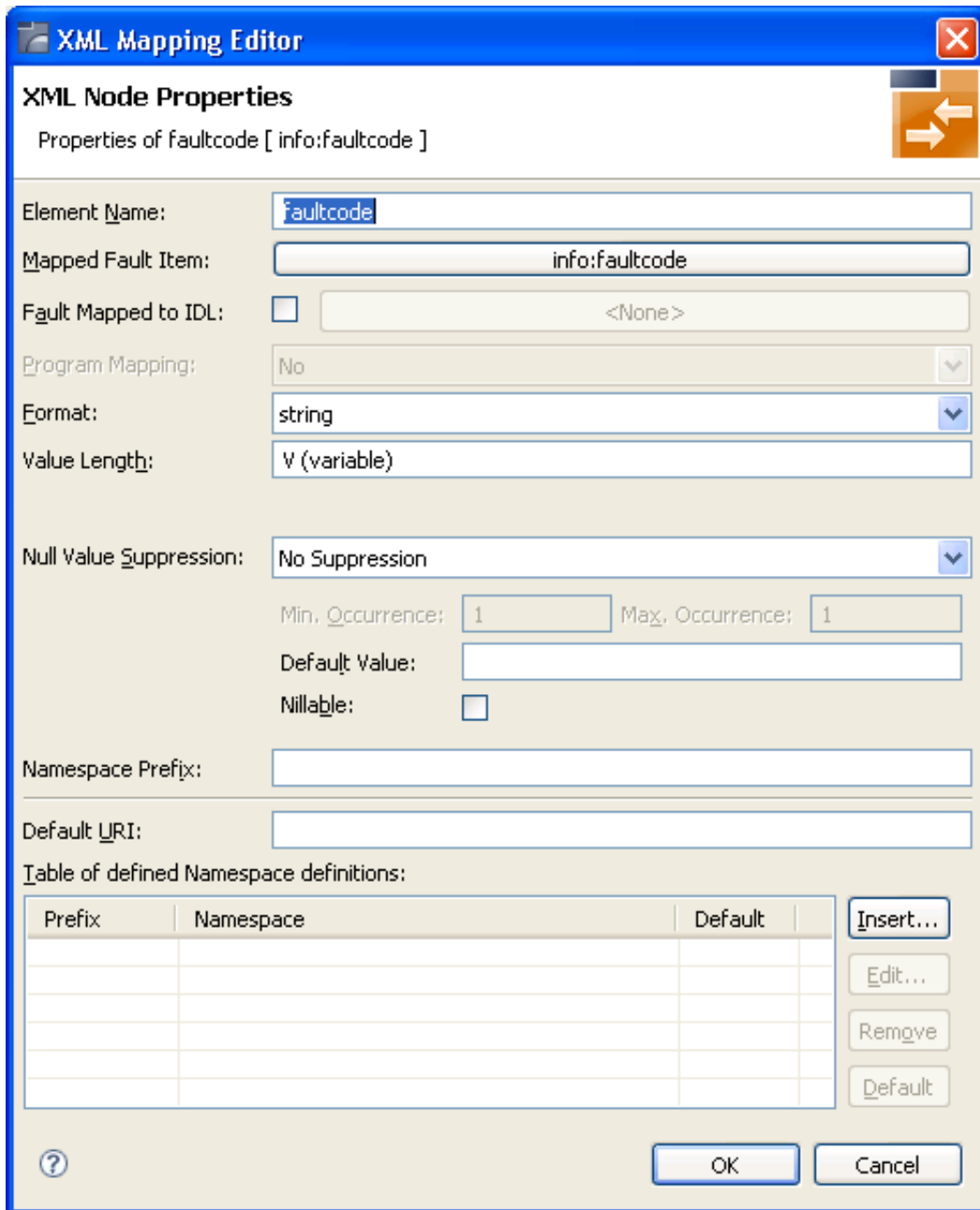
2. Remove the parameter MyStatus, including its children, because the regular response will not contain these parameters and the corresponding IDL parameters will be used for fault information later.



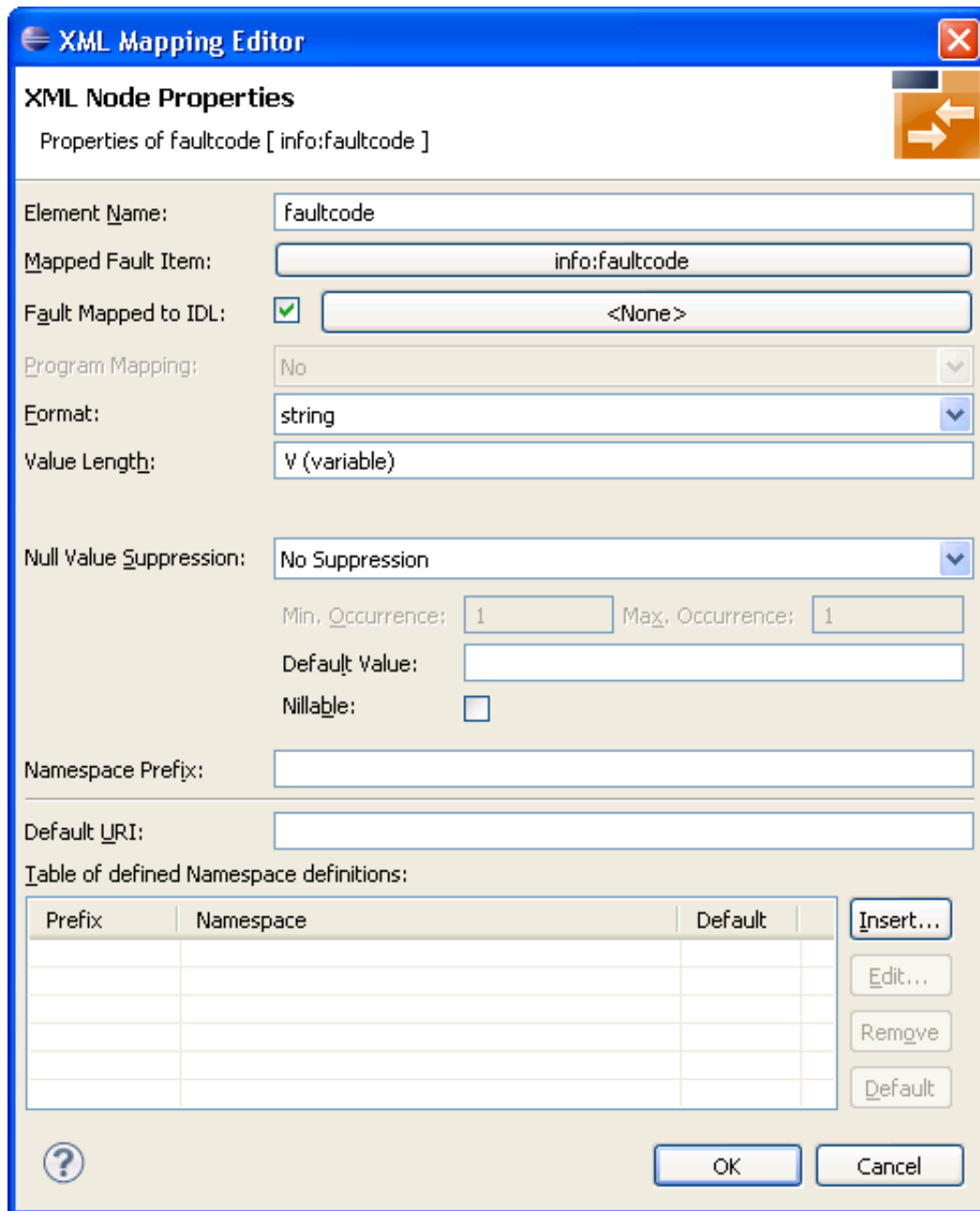
3. Open the Fault Document Manager (see bottom of opened tab **Response**) and select the document to open the following wizard:



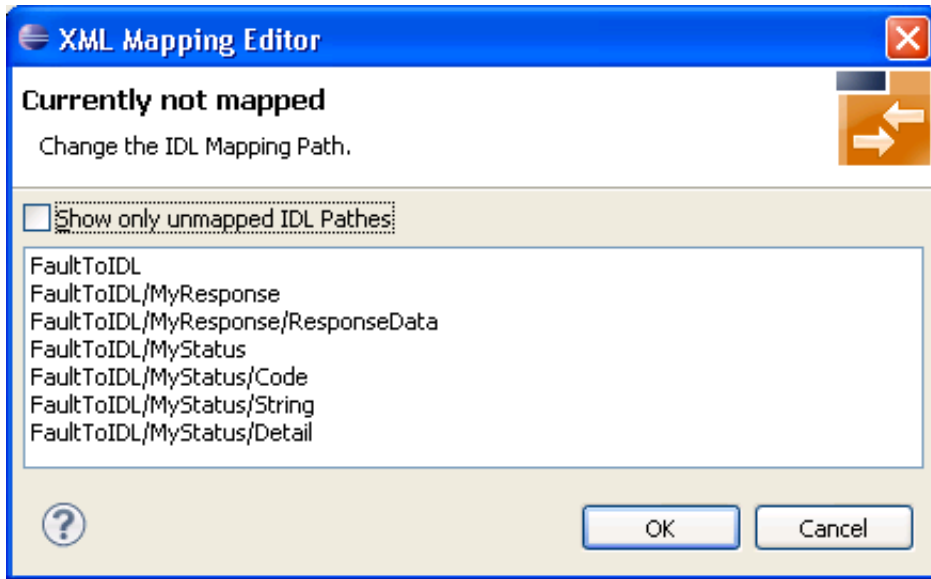
4. In the following steps, map "faultcode", "faultstring" and "detail" to IDL parameters. Fault item "faultactor" is not used in this example.
5. Select "faultcode" and open the properties shown on the following screen:



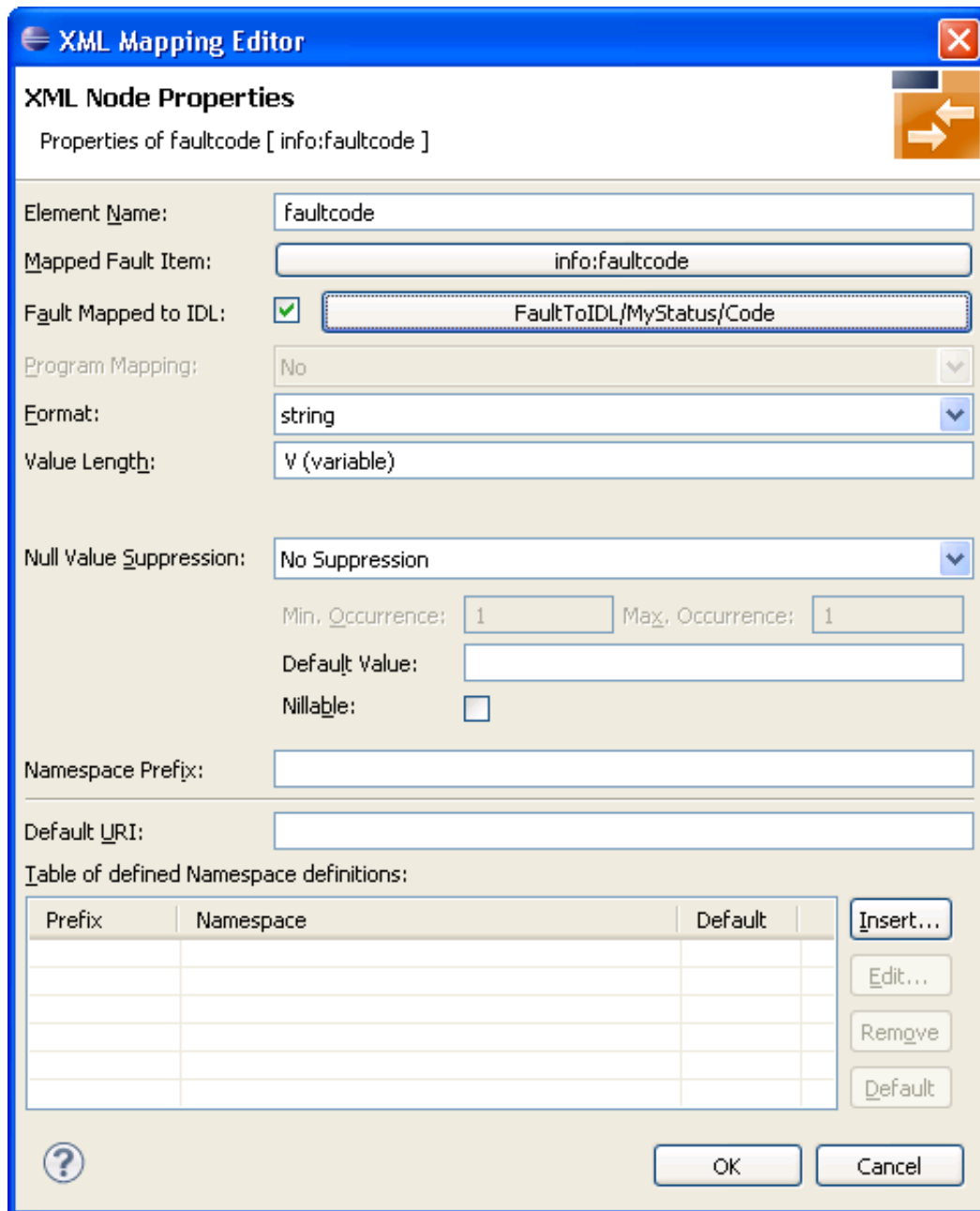
6. Check **Fault Mapped to IDL** to enable the mapping path button. In this example, a mapping path has not yet been entered and the button is labelled "<none>".



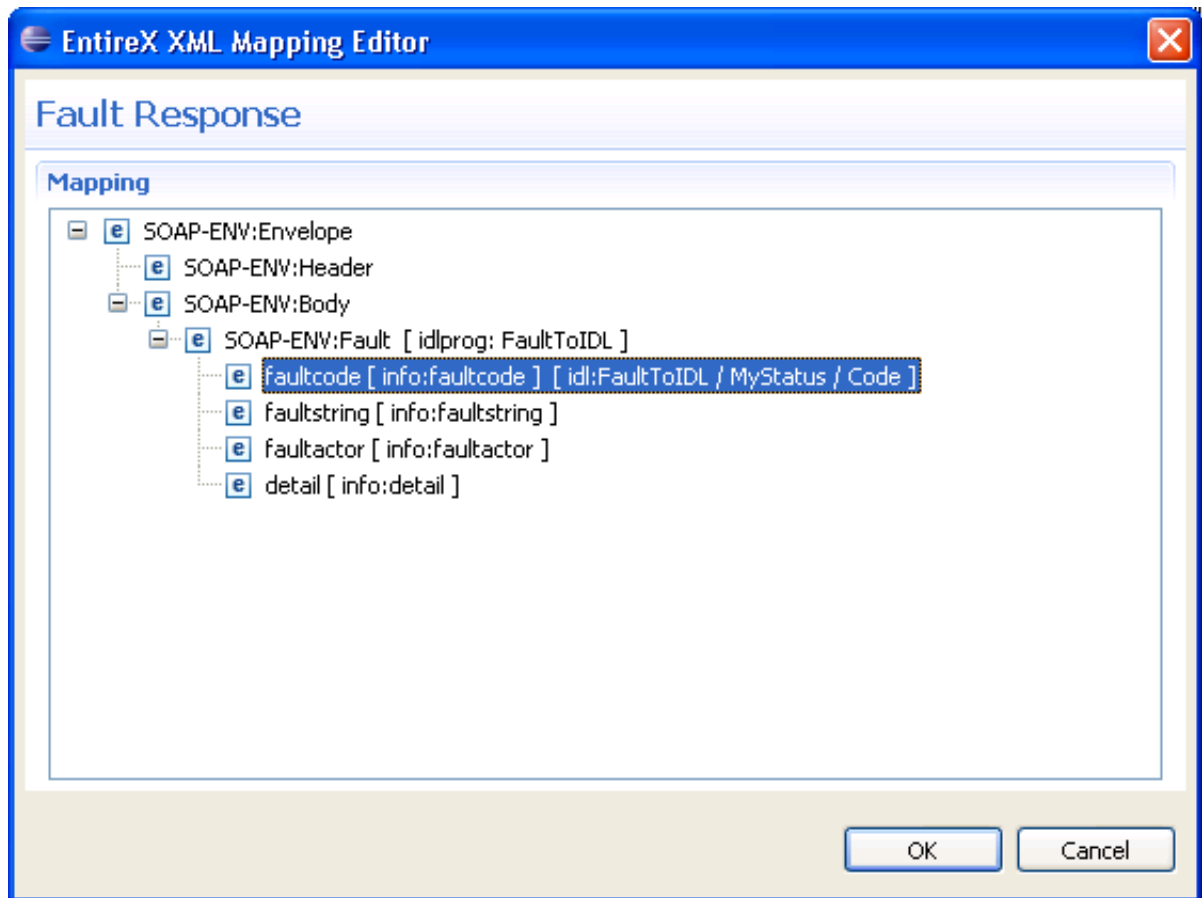
7. Press the mapping path button.



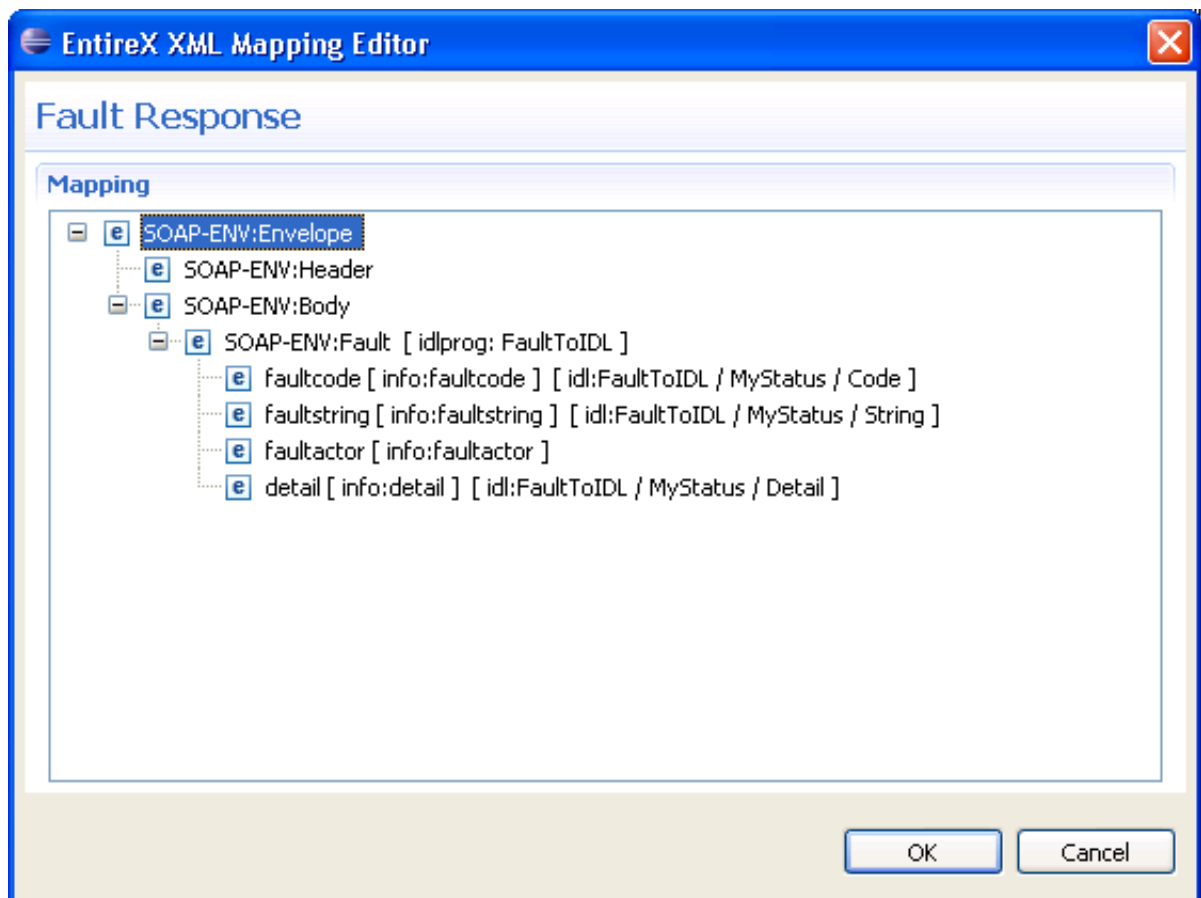
8. Select the path to IDL parameter, for example "FaultToIDL/MyStatus/Code" and choose **OK** to display the following screen:



9. Choose **OK**.



- Repeat the steps above to select the fault items "faultstring" (path to IDL parameter e.g. "FaultToIDL /MyStatus/String") and "detail" (path to IDL parameter e.g. "FaultToIDL /MyStatus/Detail").

**Note:**

The fault item "info:detail" contains the complete document fragment enclosed by an associated tag (in this example tag <detail>).

11. Choose **OK** to save the IDL-XML mapping.

In subsequent steps you need to

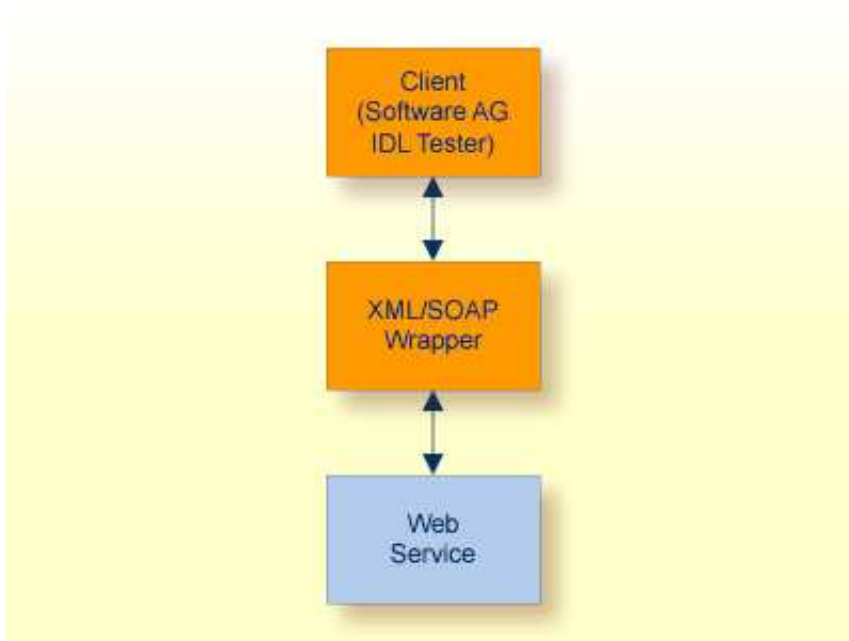
- Set up the XML/SOAP RPC Server with this XMM.
- Set up a new Web service or use an existing one.

Testing the Fault Mapping

As a quick test, implement a Web service that behaves as follows:

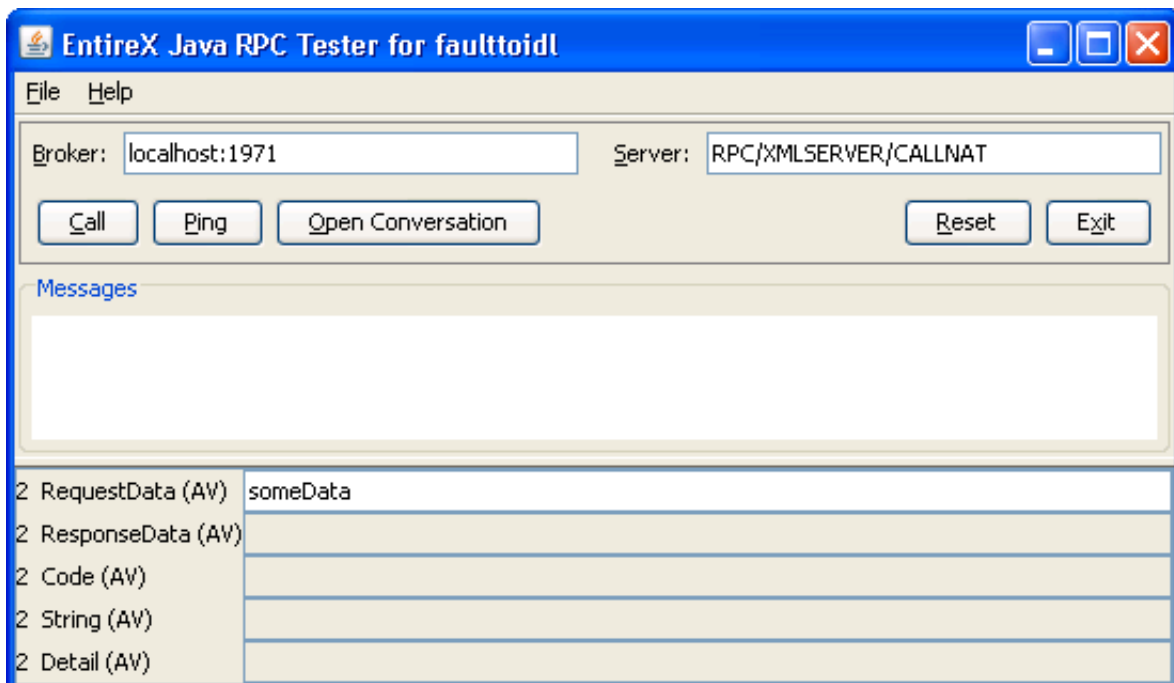
- If the "requestData" field contains any data except "throwException", the field "responseData" in the response document is set to a concatenation of the string "Receiving:" and the value of field "requestData". See Request 2).
- If the "requestData" field contains "throwException", the Web service responds with a SOAP fault.

Test Scenario

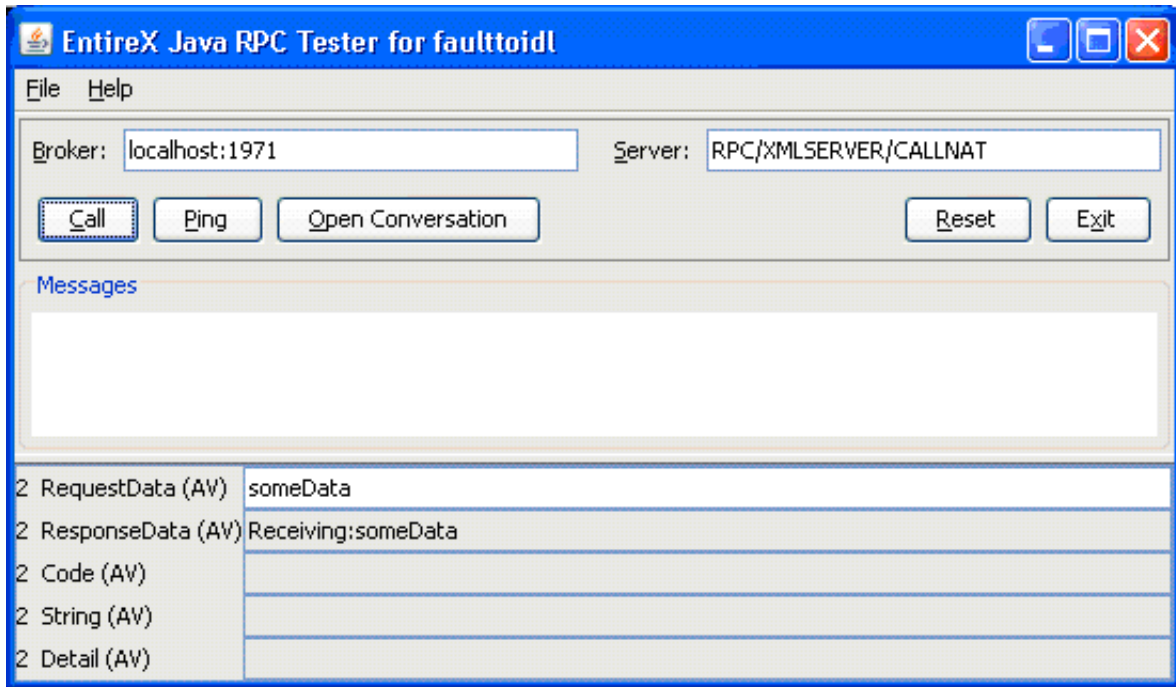


Request 1 (Expecting Normal Response)

The following screen illustrates a request that expects a normal response:

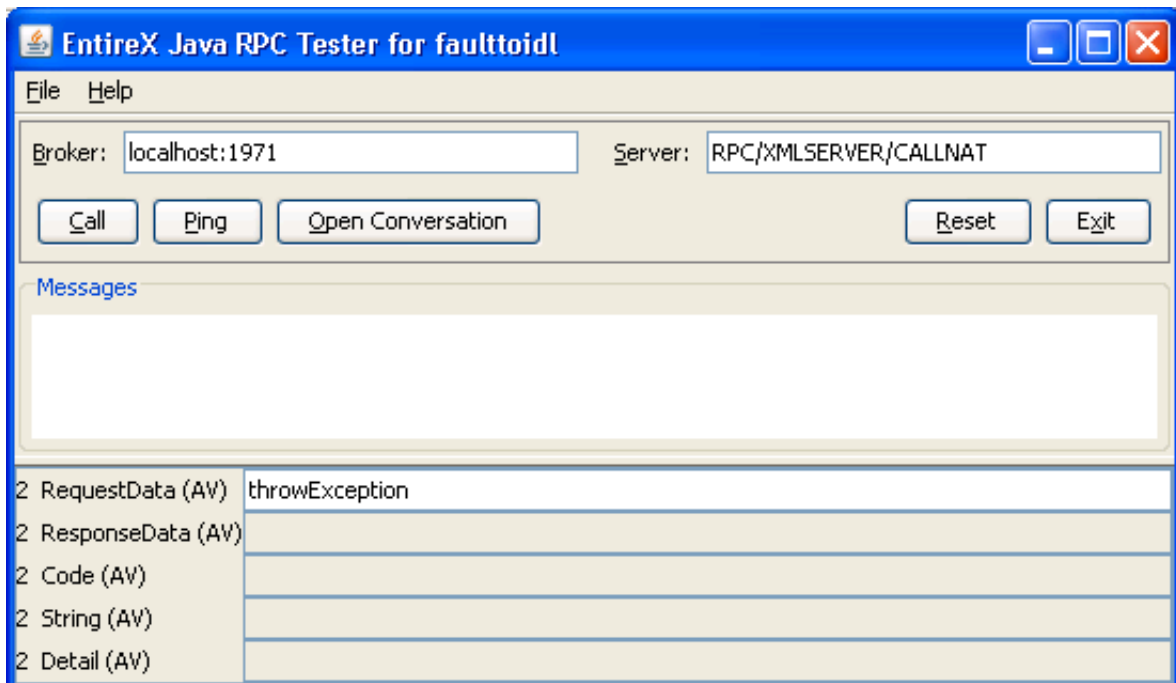


The following response is received (field responseData is filled):

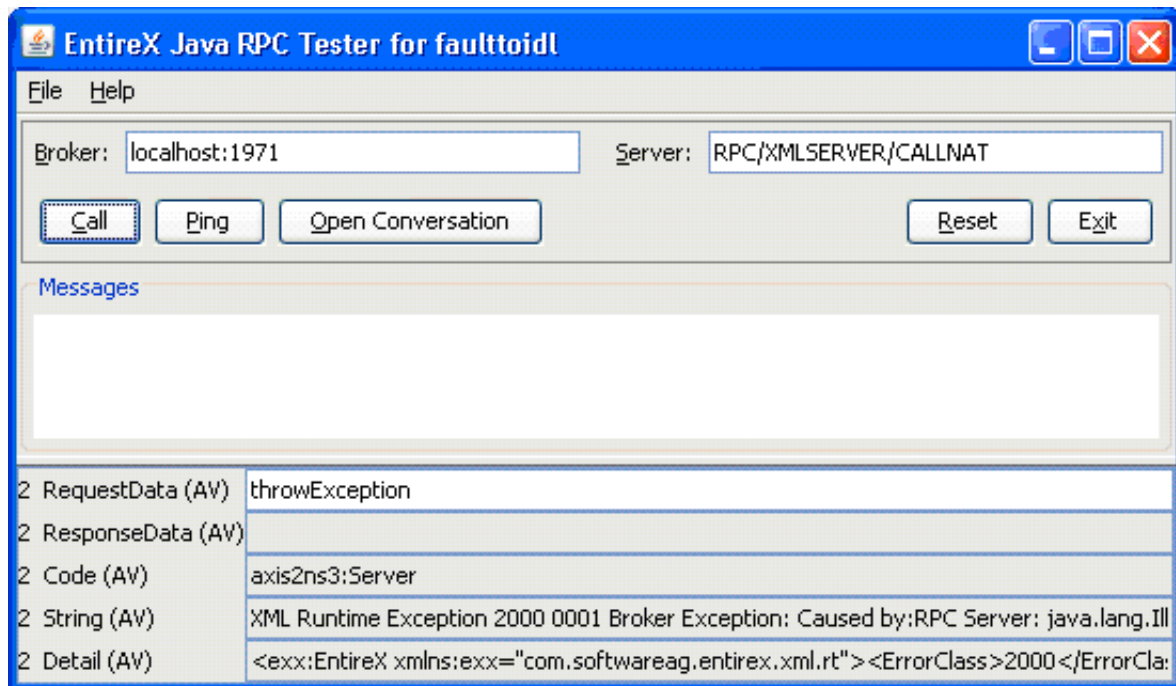


Request 2 (Expecting Fault Document)

The following screen illustrates a request where a fault document from the Web service is expected:



The following fault information is provided:



Whitespace Handling

The XML/SOAP Runtime trims whitespace in values by default. The whitespace handling is also determined by defining attribute `xml:space` (see XML specification) on element(s). The attribute `xml:space` has the higher priority and is inherited from children of the element recursively.

This section covers the following topics:

- Attribute `xml:space`
- Changing the Default for Whitespace Handling

Attribute `xml:space`

The attribute `xml:space` can be added in the XML Mapping Editor. Select an element and add new child, select the checkbox for `xml:space` and choose **OK**. Depending on the application, perform these steps for the request and/or response document definition. The attribute is added with value "preserve". If another value is required, open the properties on the attribute and change the default value.

Note:

The XML/SOAP Runtime only supports the value "preserve" for attribute `xml:space`, all other values disable the preserving of whitespace (that is, whitespace is trimmed).

Changing the Default for Whitespace Handling

The steps required to keep whitespace as default depend on the EntireX component:

- **XML/SOAP Listener**
Add the following line to file `axis2.xml` in WS-Stack Web application:

```
<parameter name="exx-xml-space">preserve</parameter>
```

- **Java API (XMLRPCService)**

Set the user property `entirex.sdk.xml.runtime.xmlspace`:

```
XMLRPCService rpcService = new XMLRPCService(...);  
...  
rpcService.setUserProperty("entirex.sdk.xml.runtime.xmlspace", "preserve");
```

- **XML/SOAP RPC Server**

Add the following line to properties file of the XML/SOAP RPC Server:

```
entirex.sdk.xml.runtime.xmlspace=preserve
```