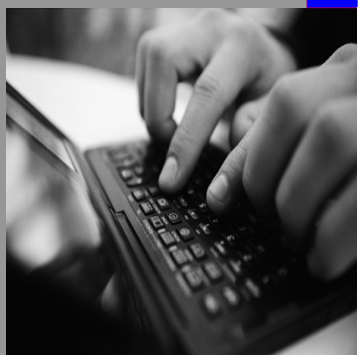


How-to Guide
SAP NetWeaver 2004s

# How To...
# Use the J2EE SOAP Adapter

Version 1.00 – August 2006

**Applicable Releases:**
SAP NetWeaver 2004s
Process Integration
Enabling Application-to-Application Processes

# Table of Contents

# 1 Scenario

You want to set up the SOAP adapter for sending messages to a Web service or providing a Web service for receiving messages.

# 2 Introduction

Web services are a standard format for exchanging data. The standard is based on XML and consists of a SOAP envelope with a header and a body. The structure of the message and the information about the connection parameters are stored in a special XML file. The structure of this file is described with the Web service definition language (WSDL). This file is usually generated by the Web service provider (receiver of the message) and can be uploaded by the Web service consumer (sender of the message).

The XI system allows you to create a WSDL file from an interface description and upload a WSDL file. The message exchange is performed by the SOAP adapter, which can transform a SOAP message to an XI message and back.

When working with the SOAP adapter you do not need to worry about the SOAP envelope, as the SOAP adapter adds this envelope for outbound messages and removes it for inbound messages. The SOAP body is identical to the payload of the XI message. Therefore, the payload of the XI message must be a valid XML.

# 3 The Step By Step Solution

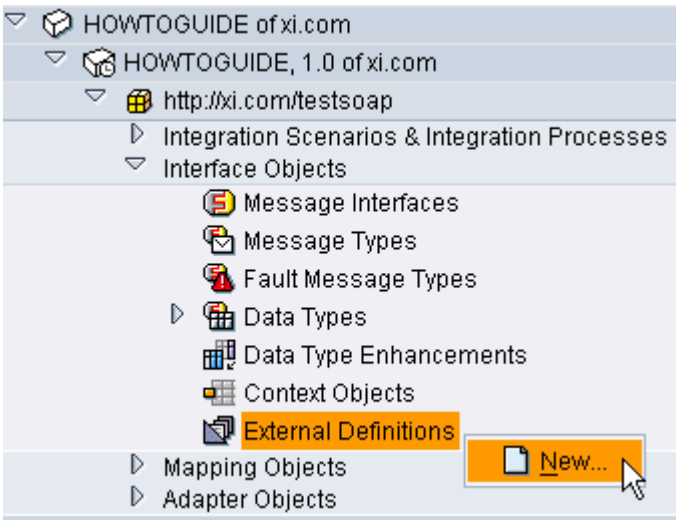
First, we will look at the basic settings of the SOAP adapter and see how to work with a WSDL file for exchanging the message structure. After that we will look at special topics of the SOAP adapter. The examples in this guide are from different scenarios and are not intended to be a step by step solution. Therefore, we will not describe configuration steps or mapping steps here.

## 3.1 The Receiver SOAP Adapter

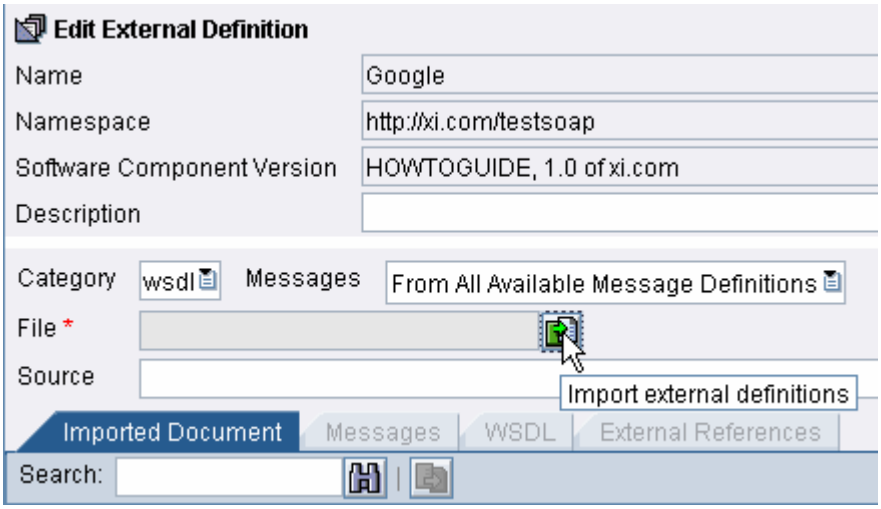
The receiver SOAP adapter connects XI with an external Web service. To call the Web service, you need to define a message interface that represents the structure of the request and response of the Web service. You can create this message interface easily with the help of a WSDL file that represents the Web service. If this is not possible, you must create the message interface based on data types that you define yourself.

### 3.1.1 Uploading the WSDL

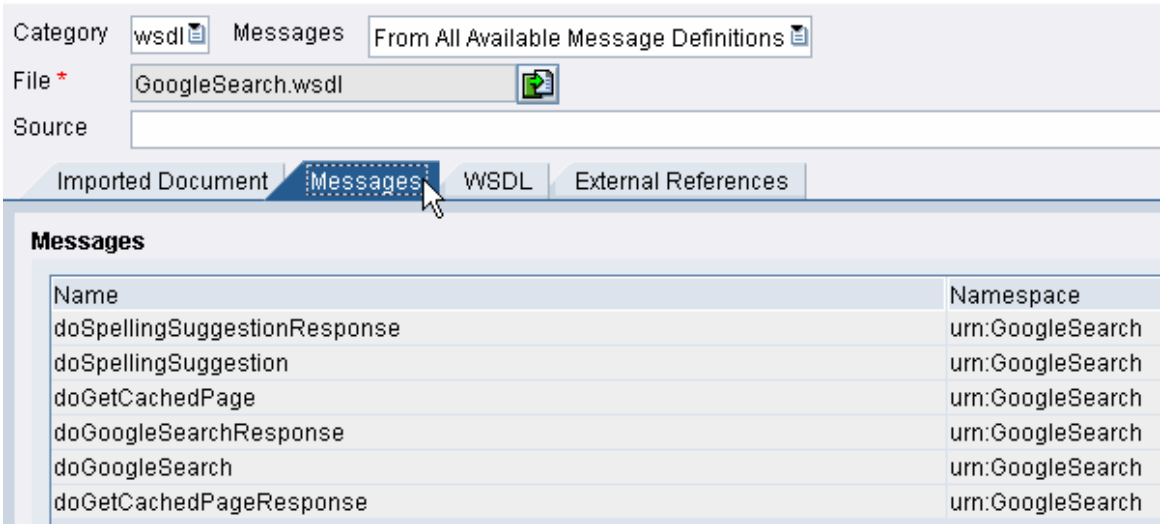
Since the WSDL describes the message interface, you upload the WSDL into the Integration Repository. Go to your *Software Component Version* and open the required *Namespace*. Create an *External Definition*:




In the external definition object, choose *Category* **`wsdl`** and click *Import external definitions*:

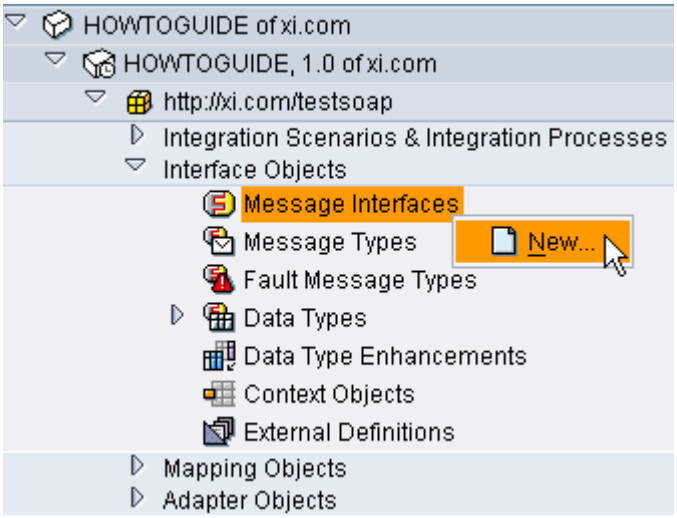



After importing the WSDL file, you can view the included messages on the *Messages* tab page:

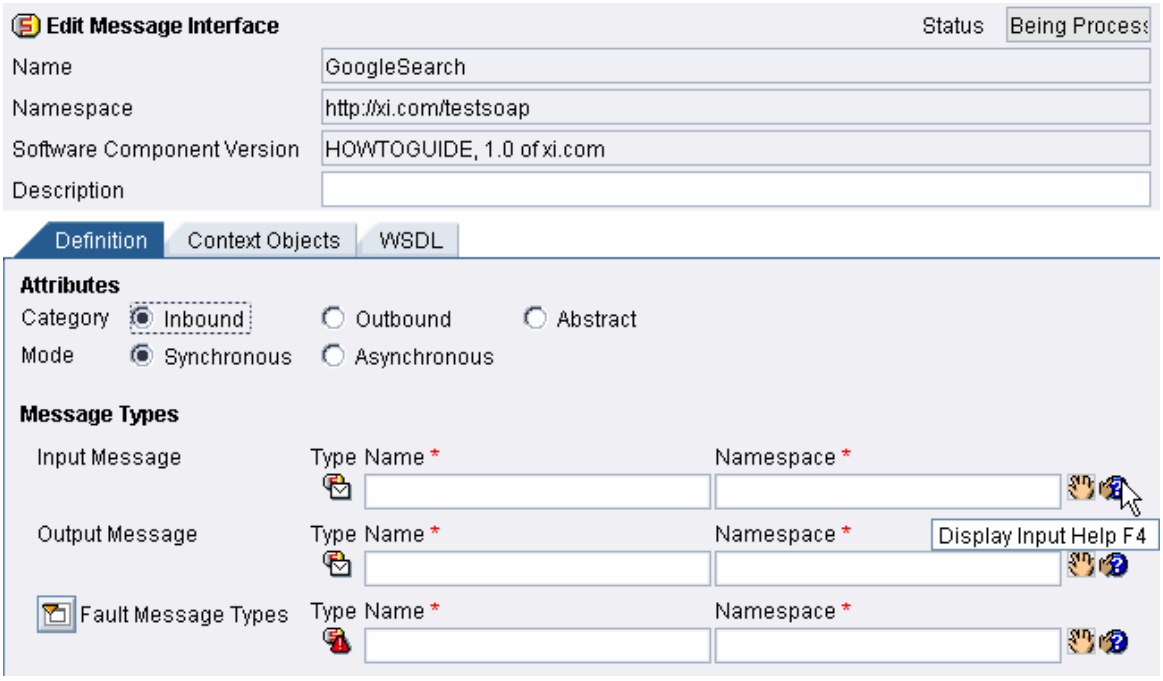


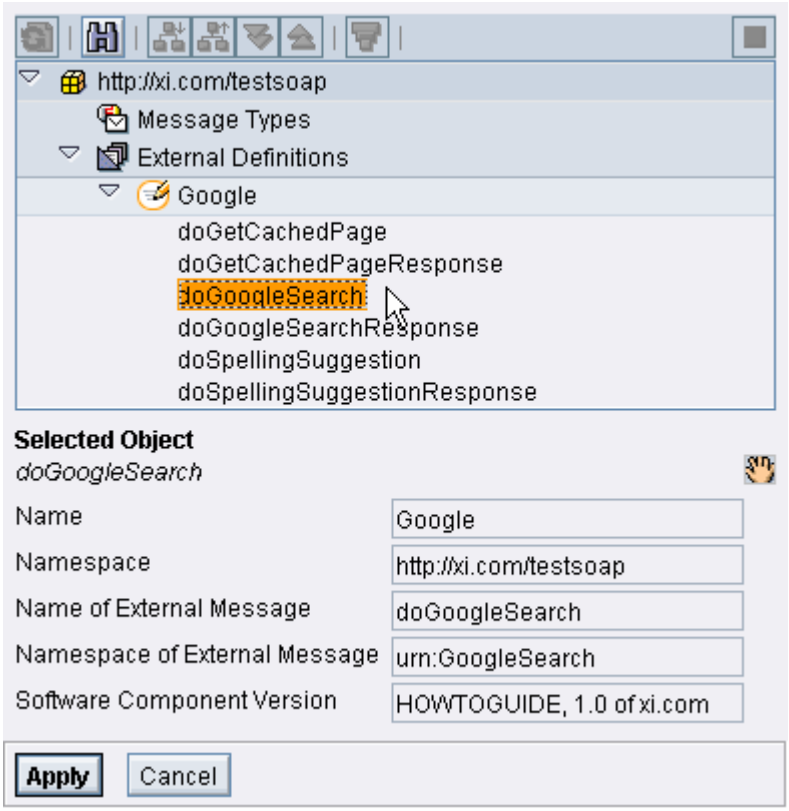

The namespaces of the messages are part of the WSDL description and can differ from the namespace of the external definition object.

Now you can create a message interface corresponding to the message types from the external definition. You need the message interface for routing the message to the Web service.




In the message interface object, select the message types of the external definition object by using the input help:






After assigning the output and input message, *save* and *activate* your work. Now you can use the message interface for routing and mapping.

### 3.1.2 Creating a Message Interface Without WSDL

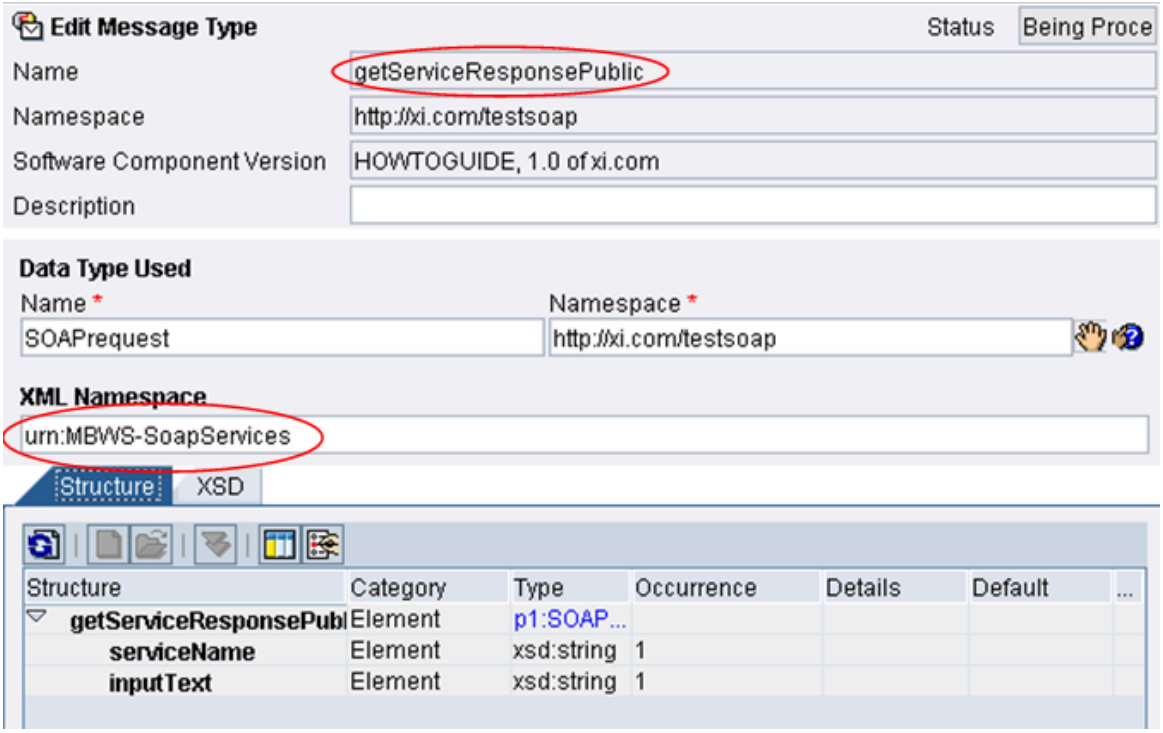
For some scenarios it is not possible to provide a WSDL file. In this case you need to create a message interface according to the message structure of the server. The best way to achieve this is by using an example SOAP message for the request and the response message.

```
<?xml version="1.0" encoding="UTF-8" ?>
- <soap:Envelope xmlns:n="urn:MBWS-SoapServices"
    xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  - <soap:Body soap:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
    - <n:getServiceResponsePublic>
        <serviceName xsi:type="xs:string">StockQuote</serviceName>
        <inputText xsi:type="xs:string">SAP</inputText>
      </n:getServiceResponsePublic>
    </soap:Body>
  </soap:Envelope>
```

Since the SOAP adapter creates the SOAP envelope you have to extract the SOAP body. Usually, Web services ignore declarative attributes (xsi:type), therefore you can delete them. The namespace declaration of the root tag must be applied. The XI message payload should be the following message:

```
<?xml version="1.0" encoding="UTF-8" ?>
<n:getServiceResponsePublic xmlns:n="urn:MBWS-SoapServices">
  <serviceName>StockQuote</serviceName>
  <inputText>SAP</inputText>
</n:getServiceResponsePublic>
```

Create a data type, message type, and message interface according to this structure. In the message type, apply the correct name and namespace of the message:



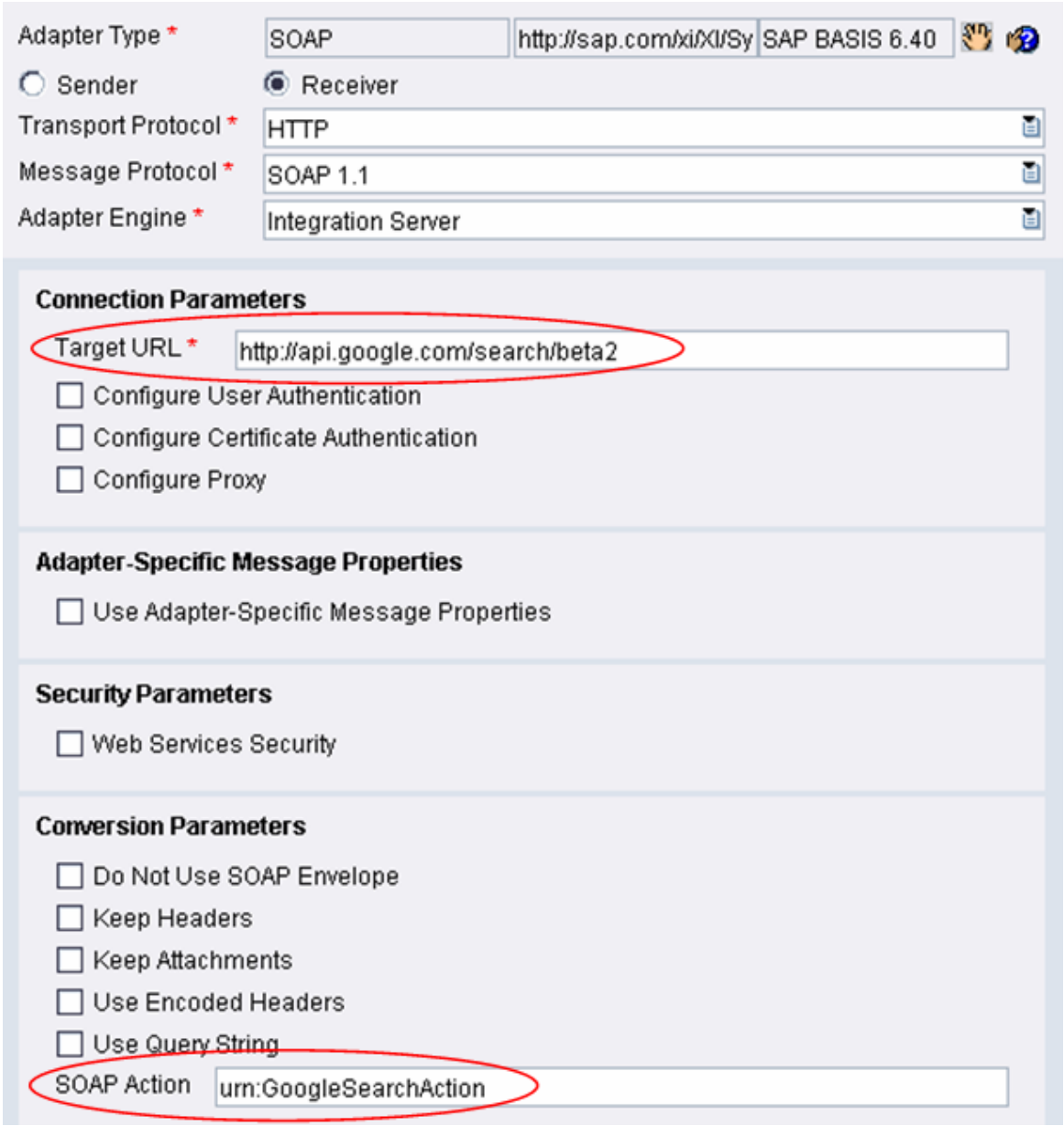

### 3.1.3 Creating a SOAP Receiver Channel

To call the Web service, you create a communication channel with type **`SOAP`** and direction **`receiver`** in the Integration Directory. The obligatory parameters in the configuration are *Target URL* and *SOAP action*. You get the values you have to enter here from the WSDL file.
You find the target URL at the tag *soap:address* and the SOAP action at the tag *soap:operation*:

```
    - <output>
        <soap:body use="encoded" namespace="urn:GoogleSearch"
          encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" />
      </output>
    </operation>
  - <operation name="doGoogleSearch">
      <soap:operation soapAction="urn:GoogleSearchAction" />
    - <input>
        <soap:body use="encoded" namespace="urn:GoogleSearch"
          encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" />
      </input>
    - <output>
        <soap:body use="encoded" namespace="urn:GoogleSearch"
          encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" />
      </output>
    </operation>
  </binding>
  <!--  Endpoint for Google Web APIs   -->
- <service name="GoogleSearchService">
  - <port name="GoogleSearchPort" binding="typens:GoogleSearchBinding">
      <soap:address location="http://api.google.com/search/beta2" />
    </port>
  </service>
</definitions>
```

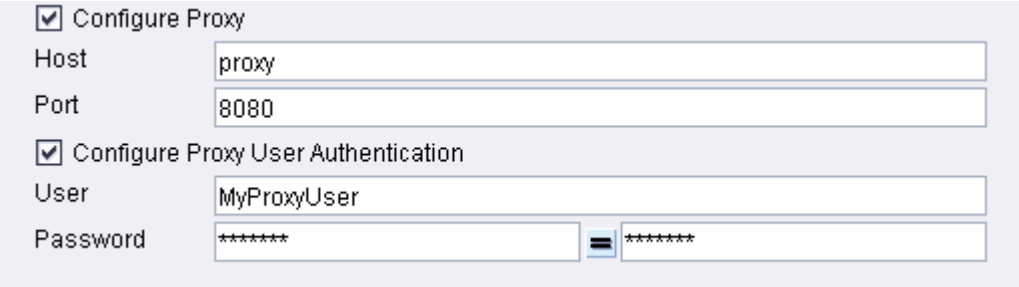
If more than one message type is described in the WSDL, several SOAP actions might also be defined. Check for the correct operation name. If no SOAP action is defined in the WSDL, you can leave the parameter in the receiver channel empty.

For the receiver SOAP adapter you need at least the following parameters:




If your Web service requires logon data, select the *Configure User Authentication* checkbox and fill in the corresponding fields:


If the Web service is outside your system landscape and you need to address a proxy server, select the *Configure Proxy* checkbox and fill in the corresponding fields:




The other parameters are described below.

## 3.2 The Sender SOAP Adapter

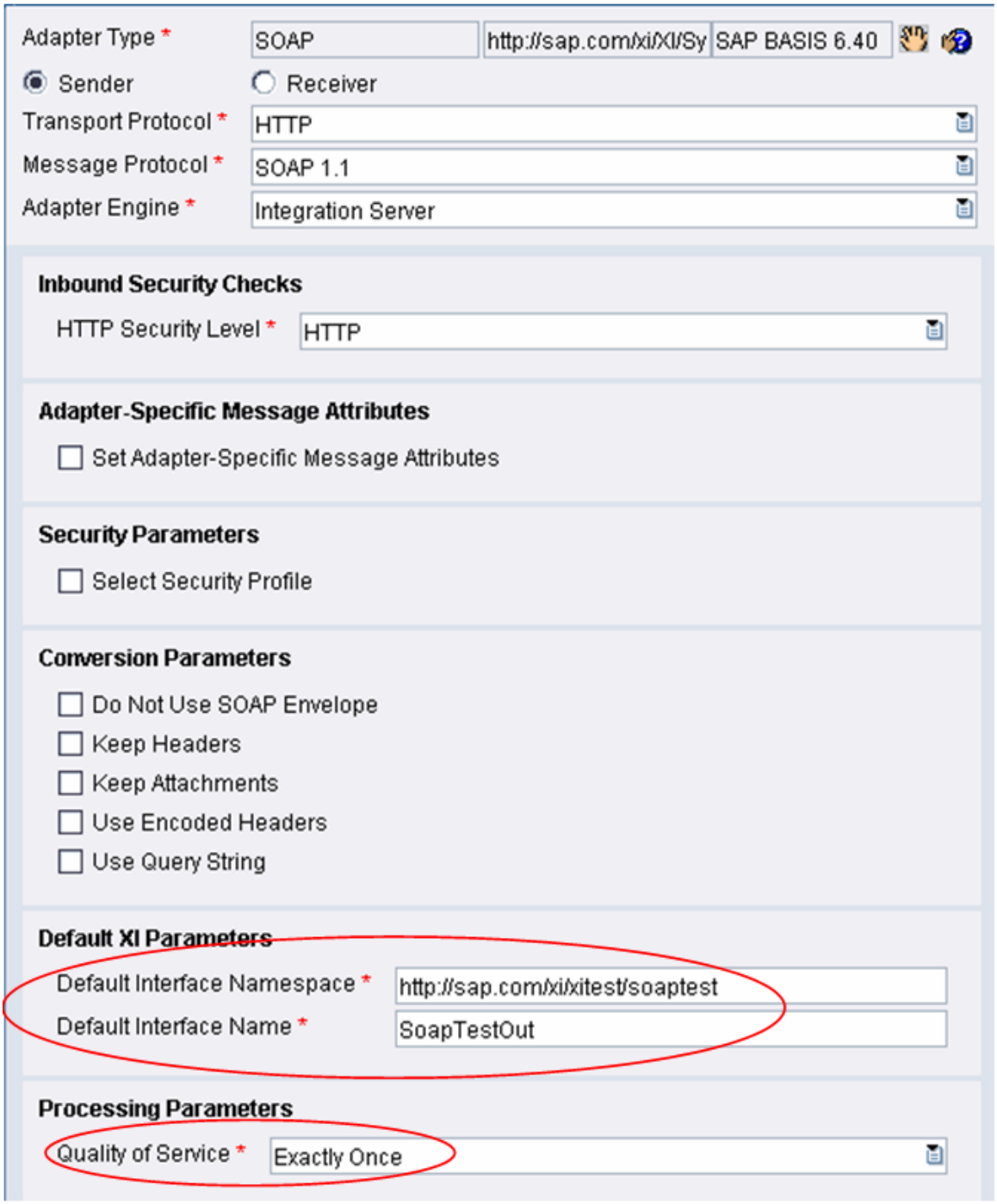
Let us assume that we have already created a message interface that we will use for the SOAP communication. We will start with the definition of the communication channel and then generate a WSDL file. The WSDL file can be used for testing with a test tool but also to generate Web service clients.

### 3.2.1 Creating a SOAP Sender Channel

When you create a SOAP sender channel you have to define the *namespace* and the *name* of a message interface. Since no input help is provided, you copy and paste the values from your Integration Repository.
Select the *Quality of Service* according to your interface type. If you are using a synchronous interface, select `Best Effort`. Otherwise, select `Exactly Once` or `Exactly Once in Order`.

If you select `Best Effort`, the Web service client will receive a response message in the SOAP body. Otherwise, the Web service client will not receive a response message if no error occurs.

For the sender SOAP adapter you need at least the following parameters:




The other parameters are described below.

After activating the SOAP adapter channel, you can send SOAP messages to the following address:

```
http://<host>:<j2ee-port>/XISOAPAdapter/MessageServlet?
channel=<party>:<service>:<channel>
```

If the SOAP adapter channel belongs to a service without party, the address is as follows:

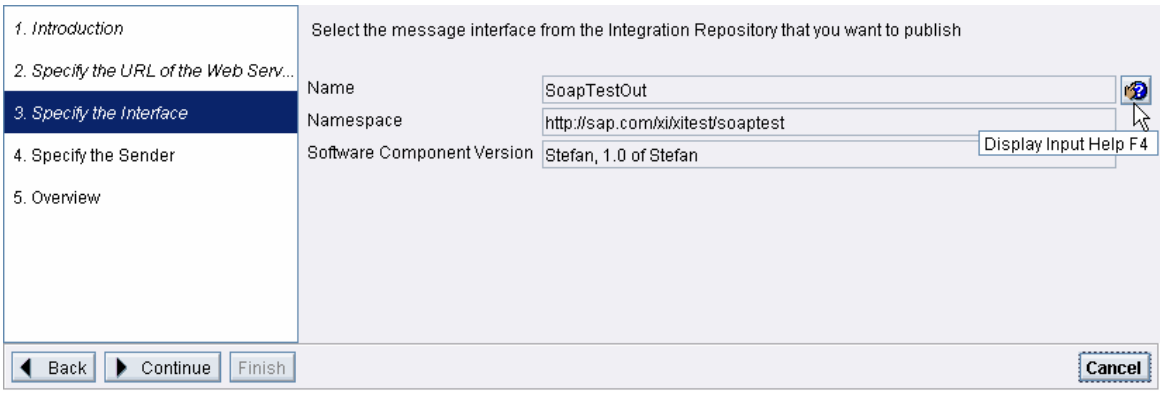
```
http://<host>:<j2ee-port>/XISOAPAdapter/MessageServlet?
channel=:<service>:<channel>
```

### 3.2.2 Creating a WSDL from an Interface Definition

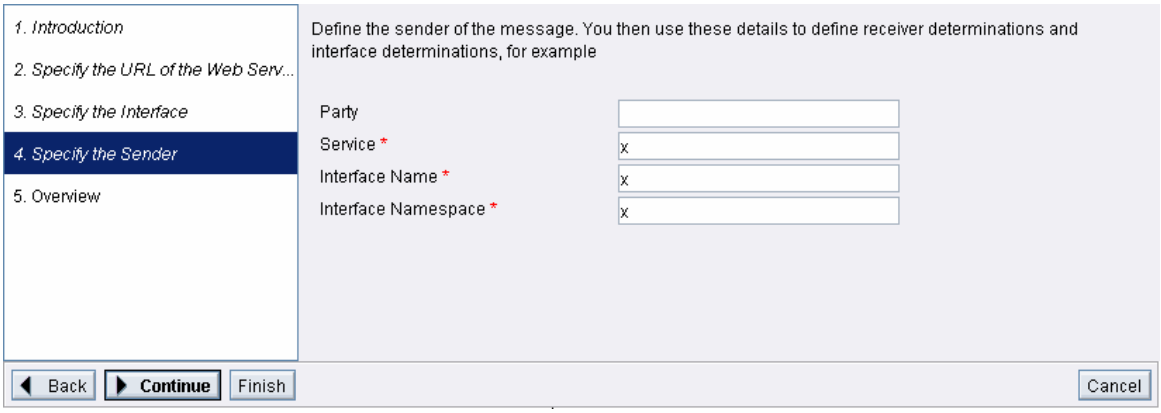
You can create a WSDL with the help of a wizard. In the Integration Directory, choose *Tools -> Define Web Service* to enter the wizard. Skip the first page by choosing *Continue*. On the second page, enter the URL mentioned above.

Do not use the *Propose URL* button, as this URL would not point to the SOAP adapter sender channel.


Go to the next page by choosing *Continue*.
Select the message interface you want to use for the WSDL. Make sure that you choose an outbound interface.




Go to the next page by choosing *Continue*.
On this page you need to enter dummy values for the obligatory parameters. They are not needed when the SOAP adapter handles the request of the Web service client.




Choose *Finish*. On the next page you can see the WSDL that has been created. Choose *Save* and specify a folder and a file name to store it on your local PC. The file extension must be *wsdl*.


You can adjust the target URL to the structure mentioned above by using an editor.

The sender SOAP adapter does not require a SOAP action, but you always have to apply logon data when using the SOAP adapter. In the central adapter engine you can use a service user such as `xiappluser`; in a non-central adapter engine or a PCK you must use one of the user names assigned to security role `xi_adapter_soap_message` for component XISOAPAdapter.

### 3.2.3 Testing the Scenario Using a SOAP Client

Before you can test your scenario, you need to make all necessary configuration steps for the receiver and interface determination and activate the scenario.

If you use Altova® XML Spy® you can upload the WSDL file you have created from the Integration Directory. Since XML Spy® takes the proxy settings from the Internet browser that is installed on your PC, make sure that the proxy server is bypassed for internal calls if your Integration Engine is installed in the same network as your PC.

In the XML Spy® menu, choose *SOAP -> Create new SOAP request*. In the dialog window, select your wsdl file. In the next dialog window, choose the SOAP operation (there should only be one).
XML Spy® now creates a proposal for the SOAP request, which you can edit. You can enter valid parameters instead of the term *String*.

```
1  <SOAP-ENV:Envelope xmlns:SOAP-ENV="
   http://schemas.xmlsoap.org/soap/envelope/"
   xmlns:SOAP-ENC="
   http://schemas.xmlsoap.org/soap/encoding/" xmlns:xsi="
   http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd
   ="http://www.w3.org/2001/XMLSchema">
2    <SOAP-ENV:Body>
3      <m:SoapTest xmlns:m="http://sap.com/xi/xitest/soaptest
   ">
4        <field1>String</field1>
5        <field2>String</field2>
6      </m:SoapTest>
7    </SOAP-ENV:Body>
8  </SOAP-ENV:Envelope>
9
```

In the menu, choose *SOAP -> Send request to server*. For your first call, you have to enter your user and password to log on to the Adapter Framework. Enter the logon data for a service user such as `xiappluser`. If the request is successful you will receive a SOAP response, which is empty in an asynchronous scenario:

```
<SOAP:Envelope xmlns:SOAP="
http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP:Header/>
  <SOAP:Body/>
</SOAP:Envelope>
```

If the request is not successful, you will receive an error message:


Depending on the error code, you may get a more detailed description in the SOAP response.

```
<?xml version="1.0"?>
<!-- see the documentation -->
<SOAP:Envelope xmlns:SOAP="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP:Body>
    <SOAP:Fault>
      <faultcode>SOAP:Server</faultcode>
      <faultstring>Server Error</faultstring>
      <detail>
        <s:SystemError xmlns:s="http://sap.com/xi/WebService/xi2.0">
          <context>XIAdapter</context>
          <code>ModuleUnknownException</code>
          <text><![CDATA[
com.sap.aii.af.mp.module.ModuleException: either no channelID specified or no channel found for the specified
party, service, and channel name, MessageServlet(Version $Id:
//tc/aii/30_VAL_REL/src/_adapters/_soap/java/com/sap/aii/af/mp/soap/web/MessageServlet.java#13 $)
  at com.sap.aii.af.mp.soap.web.MessageServlet.doPost(MessageServlet.java:380)
```

If the error code is `0`, check the URL and the proxy settings in your Internet browser. To check whether you can reach the SOAP adapter, type in the URL in your browser window.

# 4 Advanced Features

## 4.1 SOAP Header Fields

The SOAP adapter creates the SOAP envelope. You cannot influence this procedure. If you need to apply special tags inside the SOAP header, the only option is to create the whole SOAP envelope during mapping, and in the case of a synchronous call, to remove the SOAP envelope in the mapping.
In the SOAP adapter channel, select *Do Not Use SOAP Envelope*.

By using this parameter, you can use the SOAP adapter to send or receive non-SOAP messages. In this case the sender SOAP adapter requires an additional parameter *nosoap=true* in the URL. For example:

```
http://<host>:<j2ee-port>/XISOAPAdapter/MessageServlet?
channel=<party>:<service>:<channel>&nosoap=true
```

## 4.2 SOAP with Attachments

If you select *Keep Headers* in the SOAP adapter, the SOAP adapter transforms a SOAP message with attachments to an XI message with attachments and the other way around. Attachments can only be handled by Java and ABAP proxies, the SOAP adapter, the Mail adapter and the sender File adapter. If you need an attachment in another scenario such as IDoc to SOAP, then you have to create the attachment in your own adapter module that you apply on the *Module* tab page of the channel definition. For more information, see the How-To Guide *How to Create Modules for the J2EE Adapter Engine* .

## 4.3 Web Service Security

As of SP15 the sender SOAP adapter can check whether a SOAP adapter call is made using HTTP (default), HTTPS, or HTTPS with client authentication.
The receiver SOAP adapter provides HTTP and HTTPS and, as of SP13, it provides HTTPS with client authentication. You must exchange the required certificates for the HTTPS connection with the help of the *Key Storage* service in the *J2EE Visual Administrator*.

## 4.4 Providing XI Message Header Fields Inside the SOAP Message

The SOAP adapter enables you to exchange fields of the XI message header such as sender system or message ID. There are different options for providing these header fields:

If you want to use functionality, you have to select *Keep Headers* in the adapter configuration of the Integration Directory. To provide the header fields, you have the following options:

1. If you select *Keep Headers*, the XI message header fields are stored in the SOAP header.
2. If you select *Use Encoded Headers*, the receiver SOAP adapter adds the special parameter *X-XMB_WS_ENCODED* to the HTTP header and stores the XI

message header information here. The sender SOAP adapter checks for this parameter and creates the XI message header with the information provided here.

3. If you select *Use Encoded Headers* and *Use Query String*, the receiver SOAP adapter adds the XI message header information to the query string of the URL. The sender SOAP adapter creates the XI message header according to the information provided in the query string.

If you use this parameter in the receiver channel, your target URL must end with a '?' or a '&'.

The following parameters are used:

| | |
|---|---|
| MessageClass | Message class:<br>• ApplicationMessage<br>• ApplicationResponse<br>• SystemAck<br>• ApplicationAck<br>• SystemError<br>• ApplicationError |
| ProcessingMode | • Synchronous<br>• Asynchronous |
| MessageId | Message ID as GUID with format: xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx |
| RefToMessageId | Reference to message ID, as GUID with format: xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx |
| ConversationId | As string with maximum length of 60 |
| TimeSent | Time stamp (displayed as ISO8601 UTC datetime YYYY-MM-DDThh:mm:ssZ) |
| Sender.Party | Name of the sender party (displayed as agency:scheme:name) |
| Sender.Service | Service of the sender |
| Receiver.Party | Name of the receiver party (displayed as agency:scheme:name) |
| Receiver.Service | Service of the receiver |
| Interface | Specified as nsuri ^lcname |
| QualityOfService | • BestEffort<br>• ExactlyOnce<br>• ExactlyOnceInOrder |
| QueueId | For ExactlyOnceInOrder |

Here is an example of a string provided by the SOAP adapter in the HTTP message header or in the URL, when you apply the corresponding parameters:

```
version=3.0&MessageClass=ApplicationMessage&ProcessingMode=synchr
onous&MessageId=13490851-9aae-11d8-9e93-
f28d0a12631c&TimeSent=2004-04-
30T13%3A55%3A44Z&Sender.Party=016%3Apattern_33%3AAEG_837654&Sende
r.Service=SRM1&Receiver.Party=12_55%3A017%3ABASF&Receiver.Service
=SALES&Interface=http%3A%2F%2Fsap.com%2Fexample%2Fsrm%5ESRM1
```

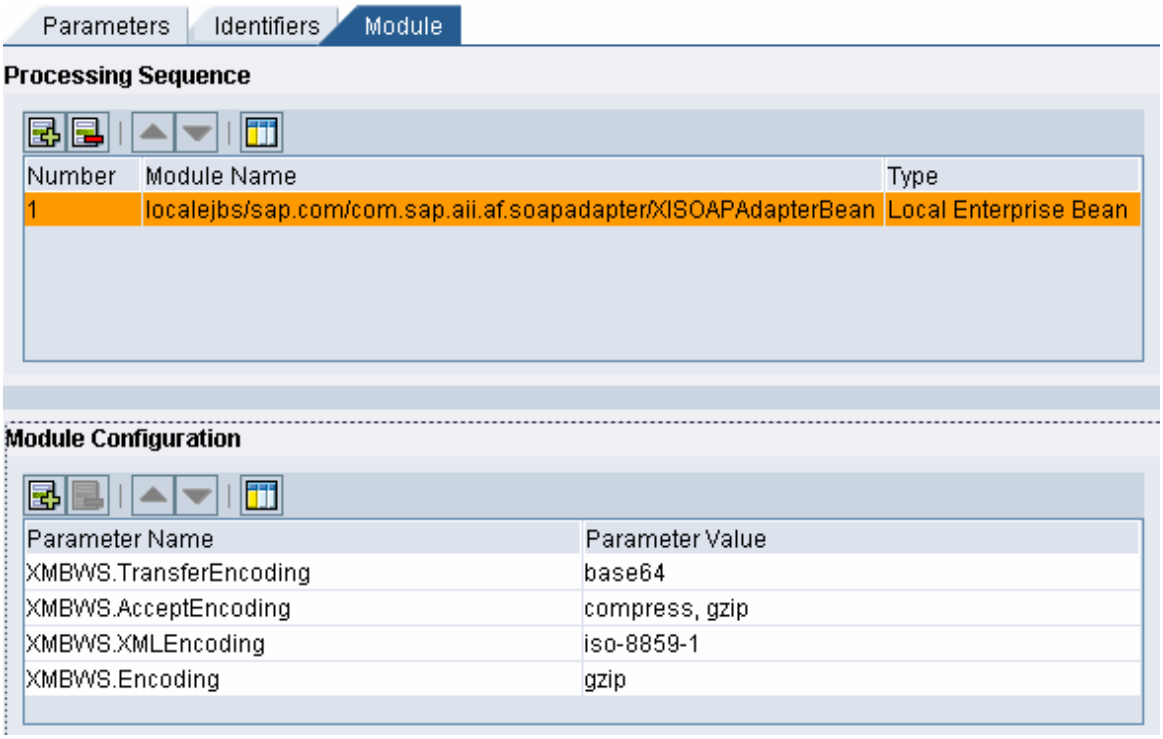
In the sender SOAP adapter, you can use only some of the parameters, but the first parameter must always be the version. For example, you send from the Web service client to the following URL:

```
http://<host>:<j2ee-port>/XISOAPAdapter/MessageServlet?
channel=<party>:<service>:<channel>&version=3.0&Interface=http%3A
%2F%2Fsap.com%2Ftest%5ETest
```

This will overwrite the default interface and namespace of the sender channel.

## 4.5 Parameter Settings on the Module Tab Page

To influence HTTP headers you can use the following parameters in the module configuration according to the standard module of the SOAP adapter `localejbs/sap.com/com.sap.aii.af.soapadapter/XISOAPAdapterBean`:



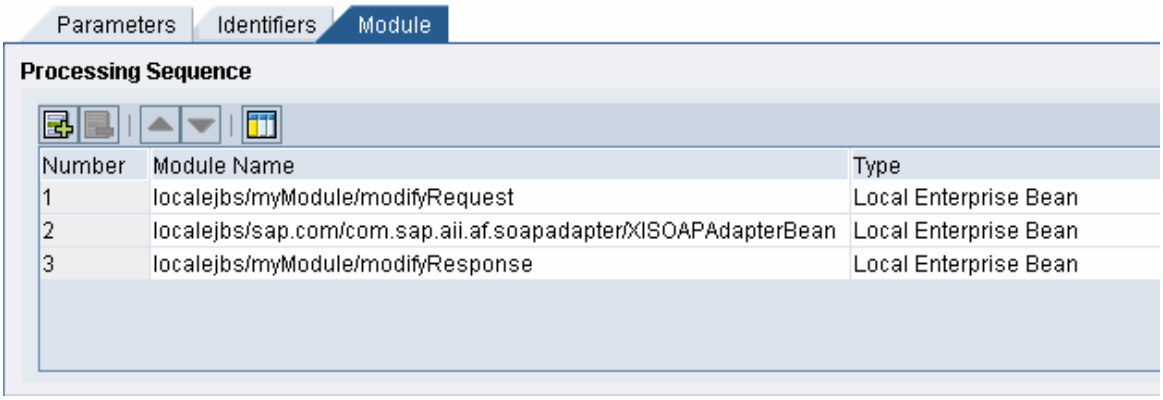

| Module Configuration Parameter | HTTP Header Entry | Example |
|---|---|---|
| XMBWS.XMLEncoding | Content-Type charset | iso-8859-1 |
| XMBWS.TransferEncoding | Content-Transfer-Encoding | base64 |
| XMBWS.Encoding | Content-Encoding | gzip |
| XMBWS.AcceptEncoding | Accept-Encoding | compress, gzip |

The receiver SOAP adapter uses these parameters for the request message; the sender SOAP adapter uses these parameters for the response message.

## 4.6 Adding Customer Modules in the SOAP Adapter

The receiver SOAP adapter allows processing of the request and the response message of a synchronous call. You can add your own adapter modules in the *Processing Sequence* on the *Module* tab page of your SOAP adapter communication channel. Put

the module for processing the request before the standard module; put the module for processing the response after the standard module.




For more information about developing adapter modules, see the How-To Guide *How to Create Modules for the J2EE Adapter Engine*.

If you want to use adapter modules for the sender SOAP adapter as well, you must be aware that the transformation of the SOAP message to an XI message takes place after the user adapter module is called, which means that you have to deal with the SOAP message and not the XI message. Therefore, you cannot use the standard modules such as PayloadSwapBean.

As of SP 17 localejbs/ will no longer be added to the module name when you enter a new adapter.

# 5 Additional Information

## 5.1 Public Web Services

The Web service scenarios are taken from http://www.xmethods.net/
On this page you will find a number of public Web services, which you can use for testing.

## 5.2 Useful Notes

SAP Note 856597 FAQ: XI 3.0 SOAP Adapter

www.sdn.sap.com/irj/sdn/howtoguides