SOAP Adapter

Web Services by definition are services, which use Web technologies (HTTP, Email) to offer services. For integration purposes, SOAP Services over HTTP are the most useful flavor of Web Services. A good introduction into SOAP can be found on www.w3schools.com.

The Bridge also has a "Simple Object Access Protocol" (SOAP) adapter ready. SOAP is a lightweight protocol for the exchange of information in a decentralized, distributed environment. SOAP is an XML language, which is commonly transported in HTTP packages.
The SOAP adapter enables you to use any other Web Service as backend for the Bridge. Instead of getting data via SQL from a database, the SOAP adapter sends a request to the other Web Service and gets back a SOAP response message. This message is then mapped to data items, similar to the process described in the chapter on database adapters.
Today, there are already many services, which speak SOAP. You may also have several Bridge services that speak together via SOAP (for instance as part of a Software Oriented Architecture (SOA) environment).

> The Bridge supports SOAP version 1.1 for RPC encoded services and SOAP version 1.2 for document-literal encoded services.

The steps involved in calling a Web Service through the Bridge are as follows:

- a client requests the Bridge Web Service
- the xUML Runtime then makes a HTTP/SOAP call to the linked Web Service
- the result is mapped to the defined data structure
- the output is then sent back to the client

On this Page:

Related Pages:

- Importing WSDL or XSD
- SOAP Adapter Reference
- XML - UML Class Mapping

Importing the Web Service Interface

Each Web Service has its own distinct interface, defined by the names of the operations and their parameters. Before an external Web Service can be used, its interface definition must be imported to the UML model or from a WSDL document. This functionality is implemented in Builder (for more information see Importing WSDL or XSD). This action will create a new UML diagram, containing all necessary building blocks for you to start modeling a service based on the WSDL.
The following example uses the WSDL file provided by our public Bridge. The service returns a exchange rate. You may download the WSDL directly from the Bridge at http://services.e2ebridge.com:30000/ExchangeRateProvider/ExchangeRateProvider?WSDL.
It might be, the Builder import fails when importing other WSDL, because not all Web Services follows the WSDL standard strictly. Then you must draw the definitions manually by following the import rules defined in Import - Export Mechanisms.
Open the newly generated "ExchangeRateProvider.wsdl.import.xml" file after importing the WSDL. The Builder placed the interface definition of the service in the package "Services". The stereotype <<SOAPRPCOperation>> indicates a web service in rpc-sytle. The nodes in that folder are the graphical representation of the definitions in the WSDL file.

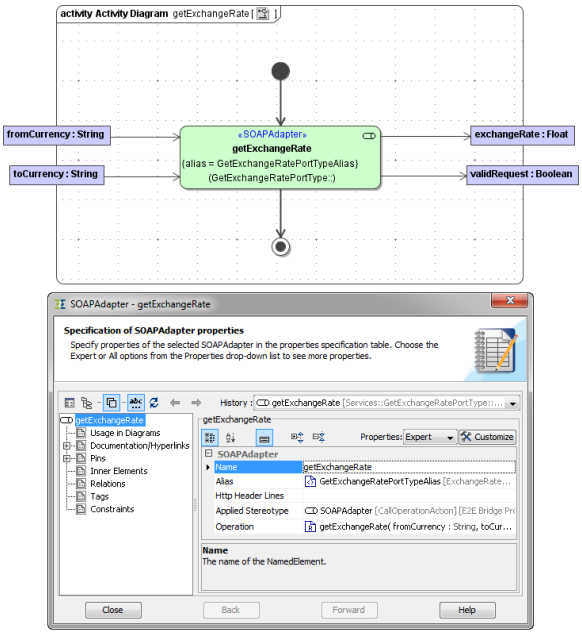
Figure: Imported Web Service RPC style

The Web Service itself is displayed as an interface. This interface features one operation "getExchangeRate" and has defined 4 parameters.
All input and output data structures are stored in the "Types" package. Since our example only uses base type parameters, this package is empty.

Accessing the Imported Web Service

The Bridge-based Services, which can use the imported Services, are defined in the "Services" package. The following figure shows how to model and access the imported web service. The following figure displays the integration of the "getExchangeRate" operation into a Bridge-based service.

Figure: SOAP Adapter Example



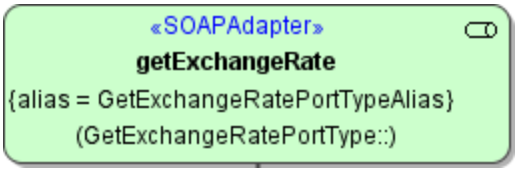
The interface to the web service has to be accessible via SOAP, so the call operation action node must be typed as <<SOAPAdapter>>. The Operation property contains the calling SOAP operation; its value is one of the imported port type operations that should be called (see the location of the available imported port types at figure Imported Web Service).

Furthermore the tagged value **alias** must be defined. The modeler defines an alias for the backend service to be used. The mapping of the alias to the actual location of the Web Service is done in the component diagram.

The activity diagram shown at figure SOAP Adapter Example will also be generated when using the Builder import function. It can be adjusted to e.g. combine it with other requests or to modify the output before sending it to the client. Please be aware not to change the generated input and output parameter's object names and types as they are representing the interface description of the imported Web Service. In this example, this is the following dependency between the getExchangeRate.wsdl and its representation in UML:
getExchangeRate.wsdl

```
...
<wsdl:operation name="getExchangeRate">
<wsdl:documentation>[Add documentation here]</wsdl:documentation>
<wsdl:input message="tns:GetExchangeRatePortType_getExchangeRate_Request"/>
<wsdl:output message="tns:GetExchangeRatePortType_getExchangeRate_Response"
/>
</wsdl:operation>
...
```

The operation defined in the WSDL file is specified in the operation property in the SOAP adapter node.



For a detailed overview on all SOAP parameters refer to SOAP Adapter Reference.

The SOAP operation parameters (**fromCurrency**, **toCurrency**, **exchangeRate**, and **validRequest** in this example) are serialized as described on XML - UML Class Mapping.

Component Diagram of a Service Using the SOAP Adapter

The physical definition, meaning which Bridge will fulfill this operation, is done in the component diagram on the SOAP alias. To make a link from the SOAP-call in the activity diagram to the corresponding component diagram the tagged value **alias** has to be used.

The <<SOAPAlias>> can hold all tagged values listed on SOAP Adapter Reference and also all tagged values from the <<URLAlias>> (see URL Adapter).

The alias-linked backend service artifact holds the location to the web service server, which supplies the "getExchangeRate" service.
This piece of information is defined at:

```
...
<wsdl:port binding="tns:GetExchangeRatePortType_SOAPBinding" name="
GetExchangeRatePortType">
<soap:address location="http://services.e2ebridge.com:30000
/ExchangeRateProvider/Services/GetExchangeRateService/Ports
/GetExchangeRatePortType"/>
</wsdl:port>
 ...
```

in the getExchangeRate.wsdl file.

Example File (Builder project Add-ons/SOAP):

<your example path>\Add-ons\SOAP\uml\soapWebServiceRPCstyle.xml

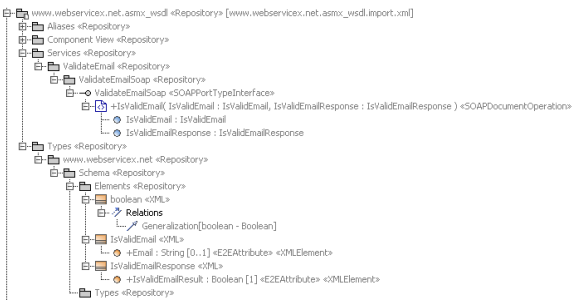
The above example shows how to access a web service in rpc style. The second, also popular style, is document style This is covered by this example:

Example File (Builder project Add-ons/SOAP):

<your example path>\Add-ons\SOAP\uml\soapWebServiceDOCstyle.xml

Note the stereotype<<SOAPDocumentOperation>>in this interface definition.

Figure: Imported Web Service DOC style



A third example shows how to call a service which requires authentication:

Example File (Builder project Add-ons/SOAP):

	<your example path>\Add-ons\SOAP\uml\soapUseSecureHelloWorldAdvancedExample.xml

The Bridge uses in his services SOAP RPC encoding for operations of a SOAP port type interface. The SOAP document-style encoding (also known as document-literal encoding) is also supported and can be chosen by selecting the stereotype <<DocumentEncoding>> on these port type operations. You can find more information about SOAP encodings in Encoding of SOAP Operations.