SOAP Headers

Example File (Builder projectAdd-ons/SOAP):

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

The SOAP header is an optional element of a soap request or response. In the header section, meta information on the actual soap message can be set. These meta information can hold additional information about routing of the message, information on encryption or affiliation to a transaction.
The SOAP header can only exist once within a message and has to be the first element within the SOAP envelope. The header though can consist of multiple SOAP header elements. These header blocks are identified via a unique URI and, through that, can be assigned to specific nodes within the message structure. Through this mechanism, the SOAP message stays flexible and can be extended by specifying additional header blocks. The communicating partners do not need any specifications on these header blocks.

The content of these headers can be of any information. A typical usage in practice would be the transfer of security information or setting a transaction ID as the following example shows:

```
<?xml version="1.0" ?>
<env:Envelope xmlns:env="http://www.w3.org/2002/12/soap-envelope">
  <env:Header>
    <login:security xmlns:login="http://example.com" env:role="
http://example.com/Login" env:mustUnderstand="true">
      <login:username>Simon Sample</login:username>
      <login:password>aPassword</login:password>
    </login:security>
    <bridge:TransactionID xmlns:bridge="http://e2e.ch/bridge"
>dkdme44546kkjdkcvdfldkjvopinmoisf</bridge:TransactionID>
  </env:Header>
  <env:Body > ... ... </env:Body>
</env:Envelope>
```

The SOAP message does not necessarily need to be sent from one sender to one receiver. The message can also reach its goal over intermediate stations. The elements can directly be addressed to these intermediate services assuming they understand and process the header information.

Runtime 2019.9 With xUML service adapter calls, the xUML Runtime adds the following outgoing HTTP headers containing correlation information to the request:

- **X-Transaction-Id** or **xTransactionId** (in JMS context)
  This header identifies the transaction the call belongs to. You can set the transaction id manually with setTransactionID. If not set, the Runtime will generate one.
  This header will be passed through the callstack to identify all service calls that belong to a transaction.
- **X-Request-Id**
  This header identifies the unique request. The Runtime generates a unique number for each adapter call.
- **X-Sender-Host** and **X-Sender-Service**
  These headers contain the sender host resp. the sender service. They are set by the Runtime automatically.

Transaction id and request id will be logged to the transaction log on the adapter call. Having this information, you can use this for error analysis or usage metrics.

On this Page:

Related Pages:

- Adding Attachments to a SOAP Call
- SOAP Call and HTTP Headers
- Providing the SOAP Adapter with URL Parameter
- Contents of the Transaction Log

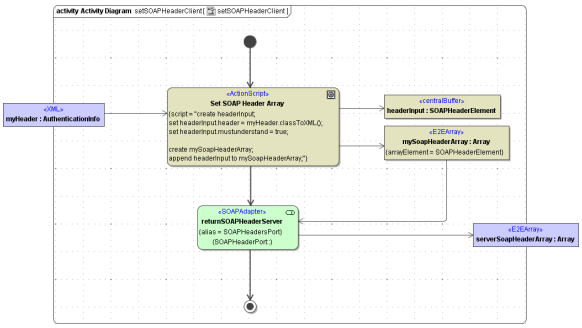
SOAP Headers in the Bridge

In Bridge context, it is important to differentiate between the following scenarios:

- SOAP headers going into a SOAP call via an adapter within a service (activity **setSOAPHeader Client** in above mentioned example)
- SOAP headers of the called SOAP service itself (activity **returnSOAPHeaderServer** in above mentioned example)

Setting the Headers before calling a SOAP adapter

The following screenshot shows a SOAP adapter call with the SOAP headers being set beforehand.



Provide the SOAP headers in an array containing elements of type **SOAPHeaderElement**. The xUML Runtime will then set the SOAP headers accordingly.
See Overview of Header Attributes and Elements below for an overview on type **SOAPHeaderElement**.

Overview of Header Attributes and Elements

```
<?xml version="1.0" ?>
<env:Envelope xmlns:env="http://www.w3.org/2002/12/soap-envelope">
  <env:Header>
    <login:security xmlns:login="http://example.com" env:role="
http://example.com/Login" env:mustUnderstand="true">
      <login:username>Simon Sample</login:username>
      <login:password>aPassword</login:password>
    </login:security>
    <bridge:TransactionID xmlns:bridge="http://e2e.ch/bridge"
>dkdme44546kkjdkcvdfldkjvopinmoisf</bridge:TransactionID>
  </env:Header>
  <env:Body > ... ... </env:Body>
</env:Envelope>
```

Class	Attribute	Type	SOAP Attribute	Description	Example XML Fragment
SOAPHeaderElement	**encoding**	String	**encodingStyle**	Specifies the encoding style used to construct the message.	`env:encodingStyle="http://www.w3.org/2003/05/soap-encoding"`

	header	Blob	-	The header holds the actual header data of base type **Blob**. The blob must contain XML structured content. So, use either the classToXML() Operation (for objects of complex types) or the classToXMLFragment() Operation (for objects of base types) to provide the necessary structure.	
	mustunderstand	Boolean	**mustUnderstand**	This attribute says whether or not the recipient (indicated by the role attribute) is required to process a header entry.	`env:mustUnderstand="true"`
	name	String	-	Holds the name of a header block.	
	namespace	String	-	Defines the namespace to be assigned to the header element serialized with use=" encoded".	
	relay	Boolean	**relay**	If relay is set to true, it indicates that the SOAP header block must not be processed by any node that is targeted by the header block, but must only be passed on to the next targeted node.	`env:relay="true"`
	role	String	**role**	The role is optional and defines the recipient of the header message.	`env:role="http://example.com/Login"`