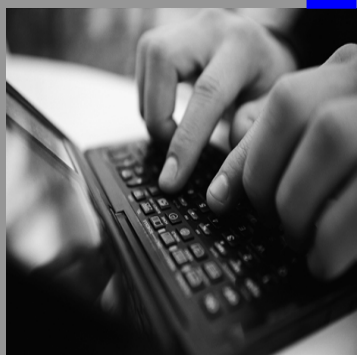


How-to Guide
SAP NetWeaver 7.0 (2004s)

How To… Set http-Header Parameters using the Axis Framework

Version 1.10 – June 2008

Applicable Releases:
SAP NetWeaver '04
SAP NetWeaver 7.0 (2004s)
End-to End-Process Integration
Enabling Application-to-Application Processes
Enabling Business-to-Business Processes

1 Scenario

As per XI 3.0 SP 20 (SAP NetWeaver 7.0 SP12, Usage Type Process Integration, respectively), HTTP header parameters can be set dynamically by means of the Axis Framework used by XI's SOAP adapter. Within the scope of this How-to guide the relevant settings will be exemplified by the parameters *Cookie*, *Set-Cookie* and *SOAPAction* which often occur in SOA communication scenarios.

HTTP cookies are a frequently used technology to enable stateful client-server interactions. This How-to guide provides a detailed description, how cookies can be exchanged between the HTTP header and XI's payload.

According to World Wide Web Consortium (W3C), "*The SOAPAction HTTP request header field can be used to indicate the intent of the SOAP HTTP request. The value is a URI identifying the intent. SOAP places no restrictions on the format or specificity of the URI or that it is resolvable. An HTTP client MUST use this header field when issuing a SOAP HTTP Request*."[1] This How-to guide provides a detailed description, how the SOAPAction parameter can be set dynamically by means of the Axis Framework.

2 Introduction

2.1 What are Cookies

The HTTP protocol is stateless by definition. Each HTTP call is technically independent from any precedent call. In order to enable stateful processing, session information has to be handled on client side. HTTP cookies are a common technique to associate the session on the client side to the one on the server side. The cookies contain a unique session identifier and they are passed forth and back as HTTP header parameters. Cookies are mainly used in the following use cases:

- Cookies may support server-sessions where a specific set of session data is kept on the server. This data can then be set-up, re-used, enhanced in multiple sequential HTTP calls.
- Cookies allow single sign-on to web servers. A session ID transported by the cookie parameters might be used instead of a repetitive sending of logon data.
- Cookies can be used to enable load balancing on the servers.

Cookies are transported as name-value pairs of the attributes "`Set-Cookie`" and "`Cookie`" in the HTTP header. In case that a session has to be kept during a sequence of HTTP calls, the response message of the first HTTP call of that session returns the cookie information in the Set-Cookie attribute:

```
HTTP/1.0 200 OK
Content-Type: text/xml; charset=utf-8
Set-Cookie: SessionID=myOwnPrivateSession12345
```

All following HTTP calls sent within the scope of that session will make use of the content of the `Set-Cookie` parameter. The client application will have to provide the cookie information of the initial response message in all following request messages. This

[1] See: http://www.w3.org/TR/2000/NOTE-SOAP-20000508/

information is transported in the `Cookie` attribute of the HTTP header of the request message:

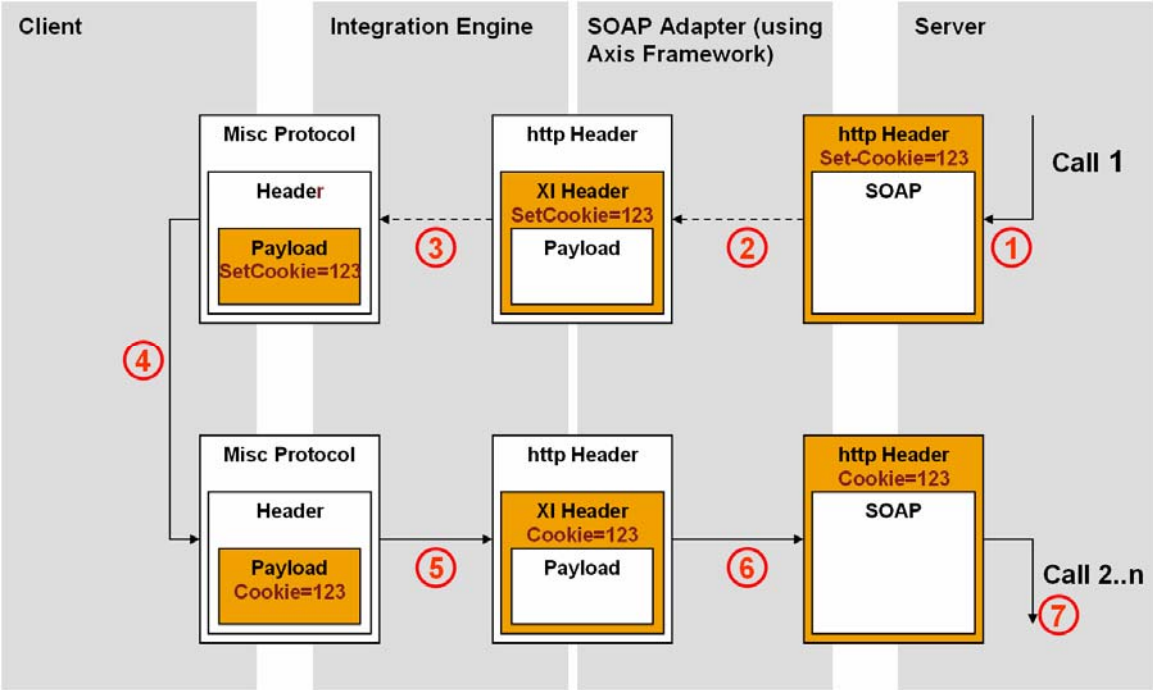
```
POST /service/services/MyService HTTP/1.0
Content-Type: text/xml; charset=utf-8
Host: <host>:<port>
Cookie: SessionID=myOwnPrivateSession12345
```

The server application will than allocate the HTTP call to the correct session.

2.2 How to Handle Cookies in XI

Since XI 3.0 SP 20 (SAP NetWeaver 7.0 SP12, Usage Type Process Integration, respectively), HTTP header attributes can be set and read in the SOAP adapter using the Axis Framework provided by the Adapter Engine. The content of the header attributes can be transferred from/to the DynamicConfiguration header of the XI SOAP protocol using the DynamicConfigurationHandler of the Axis Framework. The relevant settings will be explained in details in section 3. The XI-header parameters of the DynamicConfiguration (the so-called adapter-specific message attributes – ASMAs) can then be accessed (read/write) in a mapping program.

The following schema provides an overview how cookies can be handled during message processing in XI.



1. During the first call (logon, session create, …) the server application creates cookie to identify the session and sends it in the Set-Cookie parameter of the HTTP header of the response message of the synchronous call.
2. The SOAP adapter transfers the header parameter “Set-Cookie” to the DynamicConfiguration header of the XI SOAP message using the Axis framework.
3. The DynamicConfiguration header of the XI SOAP message can be accessed by means of a mapping program and its content can be transferred to the message payload.

4. The client application handles the cookie within its session and adds it to the payload of all following calls belonging to that session.
5. A mapping program transfers the content of the cookie to the DynamicConfiguration header of the XI SOAP message.
6. By means of the Axis framework the content of the cookie is written to the HTTP header of synchronous request message in the SOAP adapter.
7. The server application allocates the HTTP call to the correct session.

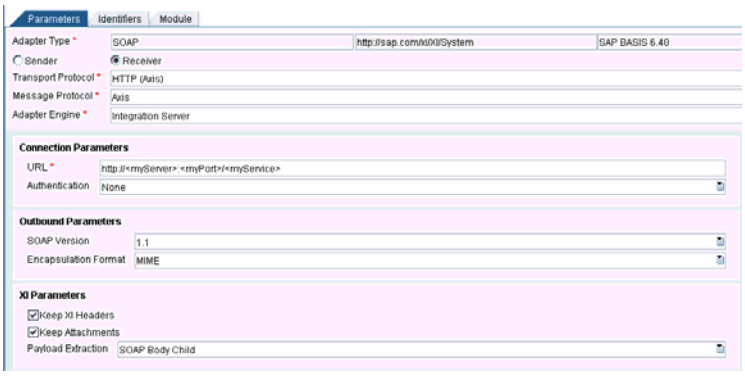
2.3 How to set the SOAPAction HTTP Header Field dynamically

The handling of the `SOAPAction` HTTP Header field in the Axis Framework of the SOAP adapter follows the schema of the Cookie attribute described in section 3.3: In a first step an ASMA of the DynamicConfiguration header of the XI SOAP protocol has to be filled with the relevant value (e.g. by means of a mapping). Its content is then populated to the HTTP header using the DynamicConfigurationHandler of the Axis Framework. Since the configuration for the SOAPAction header field varies slightly from the one for the Cookie attribute, it will be highlighted in section 3.4.

3 The Step By Step Solution

3.1 General settings for the Axis Framework of the SOAP adapter

1. Set up the SOAP receiver Communication Channel to handle HTTP header parameter dynamically



It is important to set following parameters:

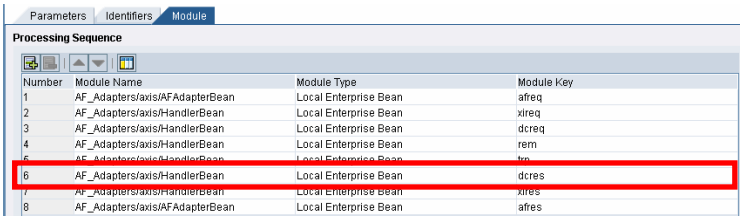
Transport Protocol	`HTTP(Axis)`
Message Protocol	`Axis`
Keep XI Headers	`"true"`

3.2 Transferring the content of Set-Cookie to the message payload

1. We assume that the content aimed for the cookie is provided by the Set-Cookie parameter in the HTTP message header.

```
HTTP/1.0 200 OK
Content-Type: text/xml; charset=utf-8
Set-Cookie: SessionID=myOwnPrivateSession12345
```

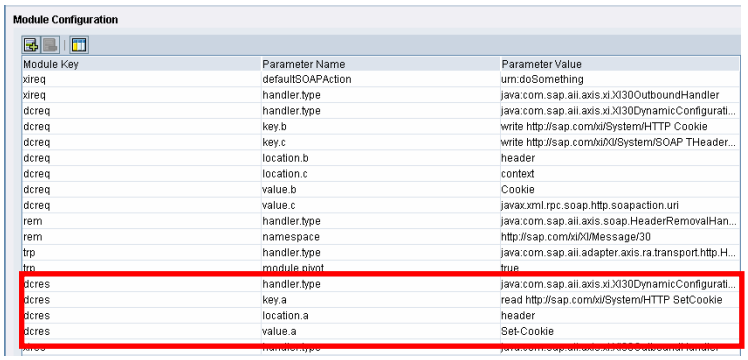
2. To transfer the HTTP header parameter `Set-Cookie` to the XI's Dynamic Configuration header called `SetCookie` add a module of type AF_Adapters/axis/HandlerBean to your processing sequence of the SOAP receiver Communication Channel created before. It has to be inserted just before the predefined module "`xires`". You may choose an arbitrary name as module key (here " `dcres`")



3. Specify the following parameters for that module

Module Key	Parameter Name	Parameter Value
dcres	handler.type	java:com.sap.aii.axis.xi.XI30DynamicConfigurationHandler
dcres	key.a	read http://sap.com/xi/System/HTTP SetCookie
dcres	location.a	header
dcres	value.a	Set-Cookie

Your Module Configuration should look like this:



Module Configuration

Module Key	Parameter Name	Parameter Value
xireq	defaultSOAPAction	urn:doSomething
xireq	handler.type	java:com.sap.aii.axis.xi.XI30OutboundHandler
dcreq	handler.type	java:com.sap.aii.axis.xi.XI30DynamicConfigurati...
dcreq	key.b	write http://sap.com/xi/System/HTTP Cookie
dcreq	key.c	write http://sap.com/xi/XI/System/SOAP THeader...
dcreq	location.b	header
dcreq	location.c	context
dcreq	value.b	Cookie
dcreq	value.c	javax.xml.rpc.soap.http.soapaction.uri
rem	handler.type	java:com.sap.aii.axis.soap.HeaderRemovalHan...
rem	namespace	http://sap.com/xi/XI/Message/30
trp	handler.type	java:com.sap.aii.adapter.axis.ra.transport.http.H...
trp	module.pivot	true
dcres	handler.type	java:com.sap.aii.axis.xi.XI30DynamicConfigurati...
dcres	key.a	read http://sap.com/xi/System/HTTP SetCookie
dcres	location.a	header
dcres	value.a	Set-Cookie

4. Set up a mapping program for the response message (here: a Message Mapping).
Develop a user-defined function `getHeaderParameter` to transfer the content of XI's Dynamic Configuration parameter `SetCookie` to the message payload.

```
public String getHeaderParameter(String paramName,String paramNamespace,Container container){
    //write your code here
    DynamicConfiguration conf = (DynamicConfiguration) container.getTransformationParameters().get(StreamTransformationConstants.DYNAMIC_CONFIGURATION);
    DynamicConfigurationKey key = DynamicConfigurationKey.create(paramNamespace, paramName);
    String value = conf.get(key);

    return value;
}
```

5. Set up a mapping to transfer the parameter to the relevant field of your message payload. The input parameters have to follow the names of the keys of XI's Dynamic Configuration header defined in step 3 of this section.

6. This field can now be evaluated in your client application and can be used in the following calls of your session.

```
...
<Item>SessionID=myOwnPrivateSession12345</Item>
...
```

3.3 Filling the `Cookie` parameter of the HTTP header

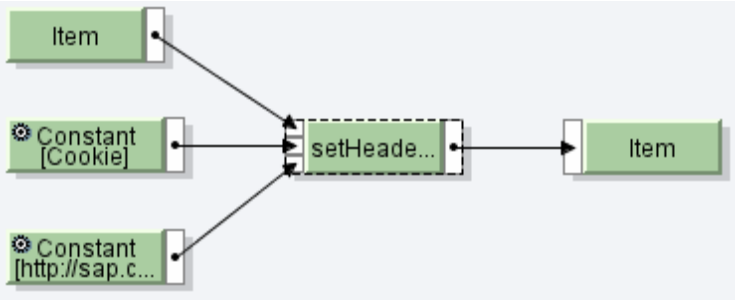
1. We assume that the content aimed for the Cookie parameter is provided by the message payload (here: within a field called Item).

```
…
<Item>SessionID=myOwnPrivateSession12345</Item
>
…
```

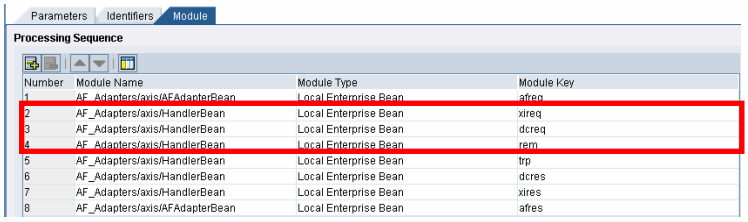
2. Set up a mapping program for the request mapping (here: a Message Mapping). Develop a user-defined function `setHeaderParameter` to transfer the content of the message payload to XI's Dynamic Configuration parameter `Cookie`.

```
public String setHeaderParameter(String
paramValue,String paramName,String
paramNamespace,Container container){

      //write your code here

      DynamicConfiguration conf
      =(DynamicConfiguration)
      container.getTransformationParameters().
      get(StreamTransformationConstants.DYNAMI
      C_CONFIGURATION);
      DynamicConfigurationKey key
      =DynamicConfigurationKey.create(paramNam
      espace, paramName);

      conf.put(key, paramValue);

      return  paramValue;
}
```

3. Set up a mapping to transfer the parameter to XI's Dynamic Configuration parameter Cookie. The input parameters have to follow the names of the keys of XI's Dynamic Configuration header defined in step 5 of this section (here: `http://sap.com/xi/System/HTTP Cookie`).



4. To transfer XI's Dynamic Configuration header called Cookie to the HTTP header parameter Cookie you have to add two modules of type AF_Adapters/axis/HandlerBean to your processing sequence of the SOAP receiver Communication Channel created before. They have to be inserted just after the predefined module "`xireq`". You may choose an arbitrary name as module key (here " `dcreq`" and "`rem`")

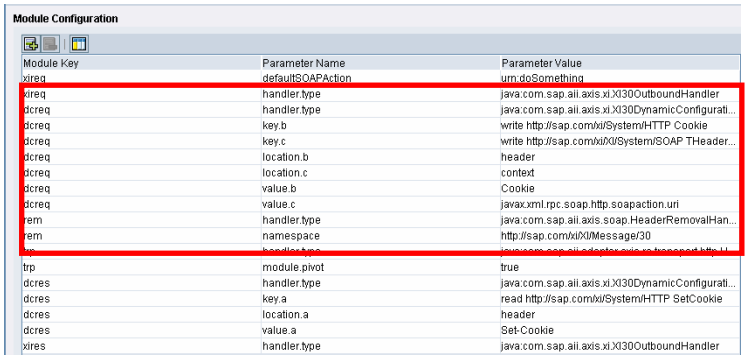


Number	Module Name	Module Type	Module Key
1	AF_Adapters/axis/AFAdapterBean	Local Enterprise Bean	afreq
2	AF_Adapters/axis/HandlerBean	Local Enterprise Bean	xireq
3	AF_Adapters/axis/HandlerBean	Local Enterprise Bean	dcreq
4	AF_Adapters/axis/HandlerBean	Local Enterprise Bean	rem
5	AF_Adapters/axis/HandlerBean	Local Enterprise Bean	trp
6	AF_Adapters/axis/HandlerBean	Local Enterprise Bean	dcres
7	AF_Adapters/axis/HandlerBean	Local Enterprise Bean	xires
8	AF_Adapters/axis/AFAdapterBean	Local Enterprise Bean	afres

5. Specify the following parameters for these modules

Module Key	Parameter Name	Parameter Value
dcreq	handler.type	java:com.sap.aii.axis.xi.XI30DynamicConfigurationHandler
dcreq	key.b	write http://sap.com/xi/System/HTTP Cookie
dcreq	location.b	header
dcreq	value.b	Cookie
rem	handler.type	java:com.sap.aii.axis.soap.HeaderRemovalHandler
rem	namespace	http://sap.com/xi/XI/Message/30

Your Module Configuration should look like this.



Module Configuration

Module Key	Parameter Name	Parameter Value
xireq	defaultSOAPAction	urn:doSomething
xireq	handler.type	java:com.sap.aii.axis.xi.XI30OutboundHandler
dcreq	handler.type	java:com.sap.aii.axis.xi.XI30DynamicConfigurati...
dcreq	key.b	write http://sap.com/xi/System/HTTP Cookie
dcreq	key.c	write http://sap.com/xi/XI/System/SOAP THeader...
dcreq	location.b	header
dcreq	location.c	context
dcreq	value.b	Cookie
dcreq	value.c	javax.xml.rpc.soap.http.soapaction.uri
rem	handler.type	java:com.sap.aii.axis.soap.HeaderRemovalHan...
rem	namespace	http://sap.com/xi/XI/Message/30
trp	module.pivot	true
dcres	handler.type	java:com.sap.aii.axis.xi.XI30DynamicConfigurati...
dcres	key.a	read http://sap.com/xi/System/HTTP SetCookie
dcres	location.a	header
dcres	value.a	Set-Cookie
xires	handler.type	java:com.sap.aii.axis.xi.XI30OutboundHandler

Note:

1. The parameters key.c, location.c and value.c are discussed in section 3.4

2. The module "`rem`" has to be used in order to remove the XI specific SOAP header.

6. The header of your HTTP request looks like the following

```
POST /service/services/MyService HTTP/1.0
Content-Type: text/xml; charset=utf-8
Host: <host>:<port>
SOAPAction: "urn:doSomething"
Cookie:
SessionID=myOwnPrivateSession12345
```

3.4 Filling the `SOAPAction` parameter of the HTTP header

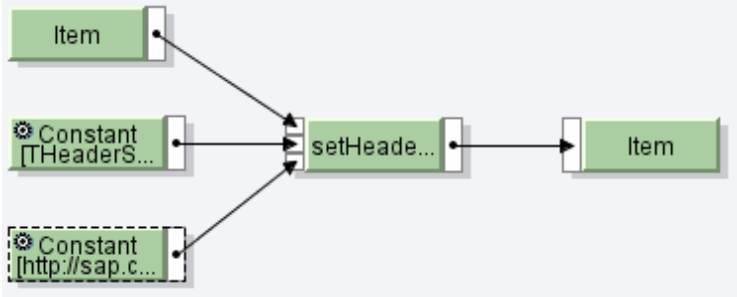
1. We assume that the content aimed for the SOAPAction parameter is provided by the message payload (here: within a field called Item).

```
…
<Item>urn:doSomethingDifferent</Item>
…
```

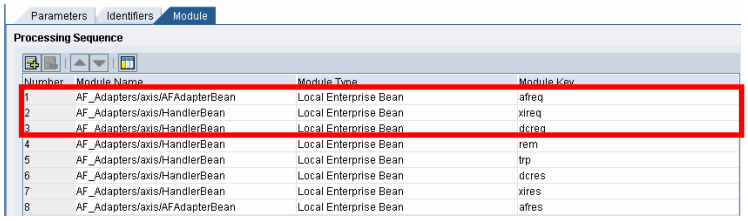
2. Set up a mapping program for the request mapping (here: a Message Mapping). Develop a user-defined function `setHeaderParameter` to transfer the content of the message payload to XI's Dynamic Configuration parameter `THeaderSOAPACTION` (You may reuse the user-defined function of step 2 of section 3.3).

```
public String setHeaderParameter(String
paramValue,String paramName,String
paramNamespace,Container container){

      //write your code here

      DynamicConfiguration conf
      =(DynamicConfiguration)
      container.getTransformationParameters().
      get(StreamTransformationConstants.DYNAMI
      C_CONFIGURATION);
      DynamicConfigurationKey key
      =DynamicConfigurationKey.create(paramNam
      espace, paramName);

      conf.put(key, paramValue);

      return  paramValue;
}
```

3. Set up a mapping to transfer the parameter to XI's Dynamic Configuration parameter `THeaderSOAPACTION`. The input parameters have to follow the names of the keys of XI's Dynamic Configuration header defined in step 5 of this section. You can make use of the Adapter-Specific Message Attribute `http://sap.com/xi/XI/System/SOAP THeaderSOAPACTION` of the SOAP adapter.



4. To transfer XI's Dynamic Configuration header called `THeaderSOAPACTION` to the HTTP header parameter SOAPAction, add two modules of type AF_Adapters/axis/HandlerBean to your processing sequence of the SOAP receiver Communication Channel created before. They have to be inserted just after the predefined module "`xireq`". You may choose an arbitrary name as module key (here " `dcreq`" and "`rem`")

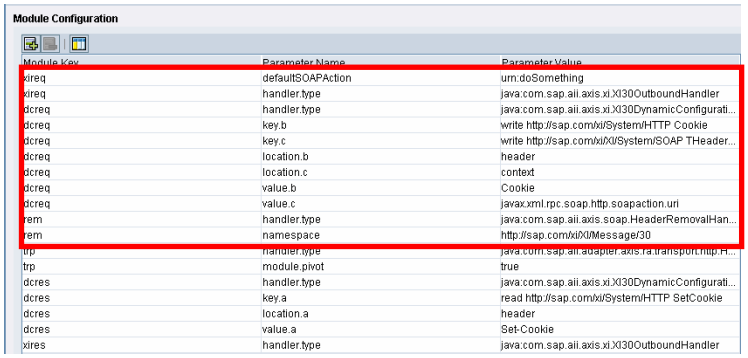


Number	Module Name	Module Type	Module Key
1	AF_Adapters/axis/AFAdapterBean	Local Enterprise Bean	afreq
2	AF_Adapters/axis/HandlerBean	Local Enterprise Bean	xireq
3	AF_Adapters/axis/HandlerBean	Local Enterprise Bean	dcreq
4	AF_Adapters/axis/HandlerBean	Local Enterprise Bean	rem
5	AF_Adapters/axis/HandlerBean	Local Enterprise Bean	trp
6	AF_Adapters/axis/HandlerBean	Local Enterprise Bean	dcres
7	AF_Adapters/axis/HandlerBean	Local Enterprise Bean	xires
8	AF_Adapters/axis/AFAdapterBean	Local Enterprise Bean	afres

5. Specify the following parameters for these modules

Module Key	Parameter Name	Parameter Value
xireq	defaultSOAPAction	urn:doSomething
xireq	handler.type	java:com.sap.aii.axis.xi.XI30OutboundHandler
dcreq	handler.type	java:com.sap.aii.axis.xi.XI30DynamicConfigurationHandler
dcreq	key.c	write http://sap.com/xi/XI/System/SOAP THeaderSOAPACTION
dcreq	location.c	context
dcreq	value.c	javax.xml.rpc.soap.http.soapaction.uri
rem	handler.type	java:com.sap.aii.axis.soap.HeaderRemovalHandler
rem	namespace	http://sap.com/xi/XI/Message/30

Your Module Configuration should look like this.



Note:

1. It is mandatory to set the parameter `defaultSOAPAction` in module xireq

2. The parameters key.b, location.b and value.b are discussed in section 3.3.

3. The module "`rem`" has to be used in order to remove the XI specific SOAP header.

6. The header of your HTTP request looks like the following

```
POST /service/services/MyService HTTP/1.0
Content-Type: text/xml; charset=utf-8
Host: <host>:<port>
SOAPAction: "urn:doSomethingDifferent"
Cookie:
SessionID=myOwnPrivateSession12345
```

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