Open Geospatial Consortium Inc.

Date: 2008-01-24

Reference number of this document: OGC® 07-158

Editors: **Rüdiger Gartmann, Bastian Schäffer**

OpenGIS® Wrapping OGC HTTP-GET and -POST Services with SOAP - Discussion Paper

Warning

This document is not an OGC Standard. This is an OGC Discussion Paper and is therefore not an official position of the OGC membership. It is distributed for review and comment. It is subject to change without notice and may not be referred to as an OGC Standard. Further, an OGC Discussion Paper should not be referenced as required or mandatory technology in procurements.

Recipients of this document are invited to submit, with their comments, notification of any relevant patent rights of which they are aware and to provide supporting documentation.

i. Preface

This Discussion Paper describes how interfaces based on HTTP-GET or HTTP-POST can be mapped to a SOAP interface in a generic way without loosing any information. This ensures that a re-mapping to the initial HTTP-GET or -POST binding is possible in order to support standard clients only capable of HTTP-GET and -POST communication. Furthermore, it specifies how a WSDL document can be derived from the service capabilities.

Thus, this document specifies how services based on HTTP-GET and -POST can be used in a SOAP-based infrastructure. This allows to leverage existing WS-* standards such as WS-Security.

ii. Document terms and definitions

This document uses the standard terms defined in Subclause 5.3 of [OGC 05-008], which is based on the ISO/IEC Directives, Part 2. Rules for the structure and drafting of International Standards. In particular, the word “shall” (not “must”) is the verb form used to indicate a requirement to be strictly followed to conform to this standard.

iii. Document contributor contact points

All questions regarding this document should be directed to the editor or the contributors:

Name	Organization
Rüdiger Gartmann	University of Münster, Institute for Geoinformatics
Bastian Schäffer	University of Münster, Institute for Geoinformatics

Wrapping OGC HTTP-GET and -POST Services with SOAP - Discussion Paper

1 Background

In the past, the OGC primarily specified and standardized services with HTTP-GET and -POST bindings. Currently, a change towards supporting SOAP bindings is performed, but this change only applies to new service specifications, and even those may still define HTTP-GET and -POST bindings. This leads to a situation that HTTP-GET/POST and SOAP bindings will probably exist in parallel for a long time.

However, the mainstream IT world primarily supports SOAP as a protocol for web service interaction, and therefore many standards and specifications for web services from outside the OGC only address SOAP bindings.

Since there is a consensus within OGC to incorporate existing web service standards as much as possible, there are many opportunities to apply mainstream standards to OGC. An example would be the area of security and rights management, incorporating standards such as WS-Security, WS-Policy, etc.

Incorporating those standards can provide solutions for SOAP-based OGC services, but this would disregard all HTTP-Get and -POST-based services which currently are in the majority.

One possible solution would be to transform those external standards to HTTP-GET and -POST, what would mean a lot of efforts to do so and would lead to incompatibility with non-OGC services.

Another way to support HTTP-GET and -POST-based services would be to define an easy and generic approach to provide these services with a SOAP binding, which would enable a transformation of any HTTP-GET and -POST services to a SOAP service.

Such a generic approach would even allow to build generic wrapper components which could be applied to any existing HTTP-GET and -POST service and the according clients, so that the transformation to SOAP could be completely transparent to the client application.

Thus, this document proposes an injective mapping for the following constellations:

- HTTP-GET (KVP encoding) ⇔ SOAP document style
- HTTP-POST (KVP encoding) ⇔ SOAP document style

- HTTP-POST (XML encoding) ⇔ SOAP document style

2 References

[1] SOAP Version 1.2 Part 1: Messaging Framework, W3C Recommendation 24 June 2003, http://www.w3.org/TR/2003/REC-soap12-part1-20030624/

[2] SOAP 1.2 Attachment Feature, W3C Working Group Note 8 June 2004, http://www.w3.org/TR/2004/NOTE-soap12-af-20040608/

[3] SOAP Message Transmission Optimization Mechanism, W3C Recommendation 25 January 2005, http://www.w3.org/TR/2005/REC-soap12-mtom-20050125/

[4] XML-binary Optimized Packaging, W3C Recommendation 25 January 2005, http://www.w3.org/TR/2005/REC-xop10-20050125/

[5] Web Services Description Language (WSDL) 1.1, W3C Note 15 March 2001, http://www.w3.org/TR/2001/NOTE-wsdl-20010315

3 Proposal

3.1 Architecture

The proposed architecture consists of a server-side proxy and a client-side proxy. The client-side proxy receives service requests via HTTP-GET and -POST and transforms them into a SOAP protocol. The server-side proxy receives these SOAP requests and restores the initial HTTP-GET or -POST request. This transformation allows applying functionalities to HTTP-GET and -POST services which are defined for SOAP only, such as security or rights management.

The architecture is sketched in Figure 1:

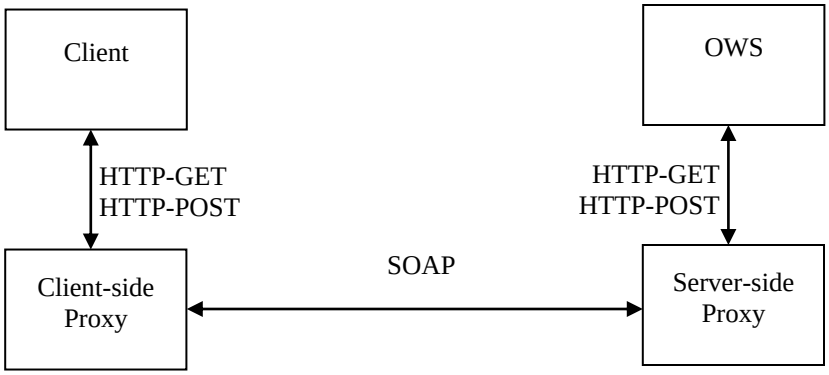

Figure 1 — Proxy Architecture

3.2 Requests

All requests have to use SOAP 1.2 [1].

3.2.1 HTTP-GET (KVP encoding) to SOAP

The only parameter encoding used in combination with HTTP-GET is KVP encoding. Thus, all KVPs have to be encoded in a simple XML structure in order to be submitted within a SOAP request.

The XML Schema for request encoding looks as follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified"
attributeFormDefault="unqualified" targetNamespace="http://www.ifgi.de/kvp2xml">
    <xs:element name="RequestProperty">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="property" minOccurs="0" maxOccurs="unbounded">
                    <xs:complexType mixed="true">
                        <xs:complexContent mixed="true">
                            <xs:extension base="xs:anyType">
                                <xs:attribute name="name" type="xs:string"
use="required"/>
                            </xs:extension>
                        </xs:complexContent>
                    </xs:complexType>
                </xs:element>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
</xs:schema>
```

This leads to a representation pattern of each separate KVP in a property element:

```
<property name="Key">Value</property>
```

Thus, a HTTP-GET request such as

```
http://anyservice.de?SERVICE=wms&VERSION=1.1.1&REQUEST=GetMap&LAYERS=layer1,layer2&FORMAT
=image%2Fpng&TRANSPARENT=true&HEIGHT=200&WIDTH=200&BBOX=
2634000,5708000,2638000,5712000&SRS=EPSG%3A31492&STYLES=default
```

would be represented as follows:

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
 <soap:Body>
  <RequestProperty xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://www.ifgi.de/kvp2xml" xsi:schemaLocation="http://www.ifgi.de/kvp2xml
http://v-ebiz.uni-muenster.de:8083/OWS-5/KVP2XML.xsd">
   <property name="SERVICE">wms</property>
   <property name="VERSION">1.1.1</property>
   <property name="REQUEST">GetMap</property>
   <property name="LAYERS">layer1,layer2</property>
   <property name="FORMAT">image/png</property>
   <property name="TRANSPARENT">True</property>
   <property name="HEIGHT">200</property>
   <property name="WIDTH">200</property>
```



```
   <property name="BBOX">2634000,5708000,2638000,5712000</property>
   <property name="SRS">EPSG:31492</property>
   <property name="STYLES">default</property>
  </RequestProperty>
 </soap:Body>
</soap:Envelope>
```

3.2.2 HTTP-POST (KVP encoding) to SOAP

The mapping of HTTP-POST requests using KVP encoding to a SOAP request is the same as for GET requests as described in section 3.2.1.

3.2.3 HTTP-POST (XML encoding) to SOAP

Since the XML payload of a HTTP request does not have to be transformed in order to be conveyable via SOAP, the XML payload is inserted into the SOAP request Body part without further modifications.

3.3 Responses

3.3.1 XML Response

Technically, XML responses can be returned by SOAP services without any modification. Nevertheless, Capabilities documents have to be transformed in order to describe the modified service bindings. See section 3.4.1 for further details.

3.3.2 Binary Response

Binary responses have to be transformed into valid SOAP responses. Although using SOAP with attachments [2] is a popular way to do so, this specification is not in a status of a W3C recommendation. Thus, MTOM (SOAP Message Transmission Optimization Mechanism) [3] in combination with XOP (XML-binary Optimized Packaging) [4] has to be used for returning binary payload. Therefore the response has to be wrapped in an element of type base64Binary. The schema for a binary reponse looks like:

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified"
attributeFormDefault="unqualified">
    <xs:element name="binaryPayload" type="xs:base64Binary"/>
</xs:schema>
```

3.3.3 Non XML or binary Respone

Some OGC web service operations allow responses to have other types than binaray or XML. For instance, the WMS GetFeatureInfor operation offeres text/plain or text/html as MIME types for their responses. To handle those special cases, the same mechanism as presented in section 3.3.2 has to be used.

3.4 Service Description

3.4.1 Capabilities

A Capabilities document contains for each operation a ‘DCPType’ node which defines the binding for this operation. The sub node ‘HTTP’ remains, since also the SOAP binding will be available via HTTP. The sub node under the HTTP node has to be

changed from ‘Get’ or ‘Post’ to ‘SOAP’. The sub node under ‘Get’ or ‘Post’ contains the URL to the described operation. This URL has to be changed to the current SOAP wrapper URL.

To identify the original binding of each operation, the <SOAP> section has to include a ‘constraints’ element as follows:

Table 1 – Constraint Definition

Initial Binding	Constraint Element
Get/KVP	<constraint name=”OriginalBinding”> <Value>Get/KVP</Value> </constraint>
Post/KVP	<constraint name=”OriginalBinding”> <Value>Post/KVP</Value> </constraint>
Post/XML	<constraint name=”OriginalBinding”> <Value>Post/XML</Value> </constraint>

A DCPType element of a capabilities document such as

```
<DCPType>
  <HTTP>
    <Get>
      <OnlineResource xmlns:xlink="http://www.w3.org/1999/xlink"
xlink:href="[ServiceURL?]" xlink:type="simple"/>
    </Get>
  </HTTP>
</DCPType>
```

would then be converted to

```
<DCPType>
  <HTTP>
    <SOAP>
      <OnlineResource xmlns:xlink="http://www.w3.org/1999/xlink"
xlink:href="[WrapperURL]?" xlink:type="simple"/>
      <constraint name="OriginalBinding>
        <Value>Get/KVP</Value>
      </constraint>
    </SOAP>
  </HTTP>
</DCPType>
```

The constraints element is needed to be able to distinguish between SOAP bindings following a SOAP specification and SOAP bindings resulting from wrapping a GET or POST service. This allows a retransformation to the original protocol.

3.4.2 WSDL

The presented proxy approach relies solely on SOAP and therefore describing the service with a WSDL 1.1 [5] document is good practice.

Applying the ideas introduced above, it does not become necessary to distinguish between a KVP or XML OWS requests, since all KVP requests are mapped to XML (see 3.2.1). Furthermore, late binding operations such as WFS.getFeature do not need to be treated specially, since the proxy approach just forwards the requests and simple xsd:anyType placeholders can be used.

The following sample WSDL should provide an idea on how to incorporate the presented solution into a WSDL:

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap12/"
xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/"
xmlns:tns="http://new.webservice.namespace"
targetNamespace="http://www.opengis.net/wms/wsdl" xmlns:wms="http://www.opengis.net/wms"
xmlns:ogcwsdl="http://www.opengis.net/ogc/wsdl">
    <wsdl:types>
        <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified" attributeFormDefault="unqualified"
targetNamespace="http://www.ifgi.de/kvp2xml">
            <xs:element name="RequestProperty">
                <xs:complexType>
                    <xs:sequence>
                        <xs:element name="property" minOccurs="0" maxOccurs="unbounded">
                            <xs:complexType mixed="true">
                                <xs:complexContent mixed="true">
                                    <xs:extension base="xs:anyType">
                                        <xs:attribute name="name" type="xs:string"
use="required"/>
                                    </xs:extension>
                                </xs:complexContent>
                            </xs:complexType>
                        </xs:element>
                    </xs:sequence>
                </xs:complexType>
            </xs:element>
        </xs:schema>
        <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified" attributeFormDefault="unqualified">
            <xs:element name="binaryPayload" type="xs:base64Binary"/>
        </xs:schema>
        <xs:schema targetNamespace="http://www.opengis.net/wms/wsdl">
            <xs:import namespace="http://www.opengis.net/wms" schemaLocation="http:/v-
ebiz.uni-muenster.de:8083/OWS-5/wms.xsd"/>
        </xs:schema>
    </wsdl:types>
    <wsdl:message name="GetCapaMessage GET">
        <wsdl:part name="request" type="tns:RequestProperty"/>
    </wsdl:message>
    <wsdl:message name="GetCapaResult">
        <wsdl:part name="response" element="wms:WMT_MS_Capabilities"/>
    </wsdl:message>
    <wsdl:message name="GetMap GET">
        <wsdl:part name="request" type="tns:RequestProperty"/>
    </wsdl:message>
    <wsdl:message name="GetMapResult">
        <wsdl:part name="response" type="tns:binaryPayload"/>
    </wsdl:message>
    <wsdl:portType name="WMS Port Type">
        <wsdl:operation name="GetCapabilities">
            <wsdl:input message="tns:GetCapaMessage GET"/>
            <wsdl:output message="tns:GetCapaResult"/>
            <wsdl:fault name="exception" message="ogcwsdl:ServiceExceptionMessage"/>
        </wsdl:operation>
        <wsdl:operation name="GetMap">
            <wsdl:input message="tns:GetMap GET"/>
            <wsdl:output message="tns:GetMapResult"/>
            <wsdl:fault name="exception" message="ogcwsdl:ServiceExceptionMessage"/>
        </wsdl:operation>
    </wsdl:portType>
    <wsdl:binding name="WMS SOAP_Binding" type="tns:WMS_HTTP_Port_SOAP">
        <wsdl:documentation>
         WMS interface bound to SOAP over HTTP/1.1.
```



```
      </wsdl:documentation>
        <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>
        <wsdl:operation name="GetCapabilities">
            <soap:operation
soapAction="http://www.opengis.net/wms/requests#GetCapabilities"/>
            <wsdl:input>
                <soap:body use="literal" parts="request"/>
            </wsdl:input>
            <wsdl:output>
                <soap:body use="literal"/>
            </wsdl:output>
            <wsdl:fault name="exception">
                <soap:fault use="literal" name="exception"/>
            </wsdl:fault>
        </wsdl:operation>
        <wsdl:operation name="GetMap">
            <soap:operation soapAction="http://www.opengis.net/wms/requests#GetMap"/>
            <wsdl:input>
                <soap:body use="literal"/>
            </wsdl:input>
            <wsdl:output>
                <soap:body use="literal"/>
            </wsdl:output>
            <wsdl:fault name="exception">
                <soap:fault use="literal" name="exception"/>
            </wsdl:fault>
        </wsdl:operation>
    </wsdl:binding>
    <wsdl:service name="WMS">
        <wsdl:port name="WMS SOAP" binding="tns:WMS SOAP_Binding">
            <soap:address location="http:/v-ebiz.uni-
muenster.de:8083/kvp2xml/sampleWMS"/>
        </wsdl:port>
    </wsdl:service>
</wsdl:definitions>
```

