Open Geospatial Consortium

Publication date: 2014-08-22

Approval Date: 2013-01-20

Posted Date: 2012-11-01

Reference number of this OGC® document: 12-133

Reference URL for this document: http://www.opengeospatial.org/doc/PER/ows9/ws-facade

Category: Public Engineering Report

Editor: John Hudson

OGC® Web Services Facade for OGC IP Engineering Report

Warning

This document is not an OGC Standard. This document is an OGC Public Engineering Report created as a deliverable in an OGC Interoperability Initiative and is 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, any OGC Engineering Report should not be referenced as required or mandatory technology in procurements.

Document type: OGC® Public Engineering Report
Document subtype: NA
Document stage: Approved for public release
Document language: English

Abstract

This document describes the Web Services Façade which was developed by LISAsoft as part of the OWS-9 testbed. The document also includes discussions about lessons learned during the development, and suggestions for future development.

This Engineering Report documents the Web Services Façade work done within OWS-9 as an extensible, open source tool, which supports translations between different protocols for a specific web service. For the OWS-9 testbed, it has been set up to translate between POST and SOAP services for a Web Feature Service. However, it can be configured to support translations between multiple protocols, such as REST, SOAP, KVP, JSON, as well as supporting multiple web services.

The Web Services Façade is an extensible, open source tool, which supports translations between different protocols for a specific web service. For the OWS-9 testbed, it has been set up to translate between POST and SOAP services for a Web Feature Service. However, it can be configured to support translations between multiple protocols, such as REST, SOAP, KVP, JSON, as well as supporting multiple web services.

Keywords

ogcdoc, ogc documents, web services, faced, wfs, post, soap

License Agreement

Contents

OGC® Web Services Facade for OGC IP Engineering Report

1 Introduction

1.1 Document contributor contact points

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

Name	Organization
Cameron Shorter	LISAsoft
John Hudson	LISAsoft

1.2 Revision history

Date	Release	Editor	Primary clauses modified	Description
16/10/2012	0.1	John Hudson	All	Initial version
26/10/2012	0.2	Cameron Shorter	All	Review
1/11/2012	1.0	Michelle Handscomb	All	Final

1.3 References

- Available on the OGC portal: OWS 5 SOAP/WSDL Common Engineering Report. References

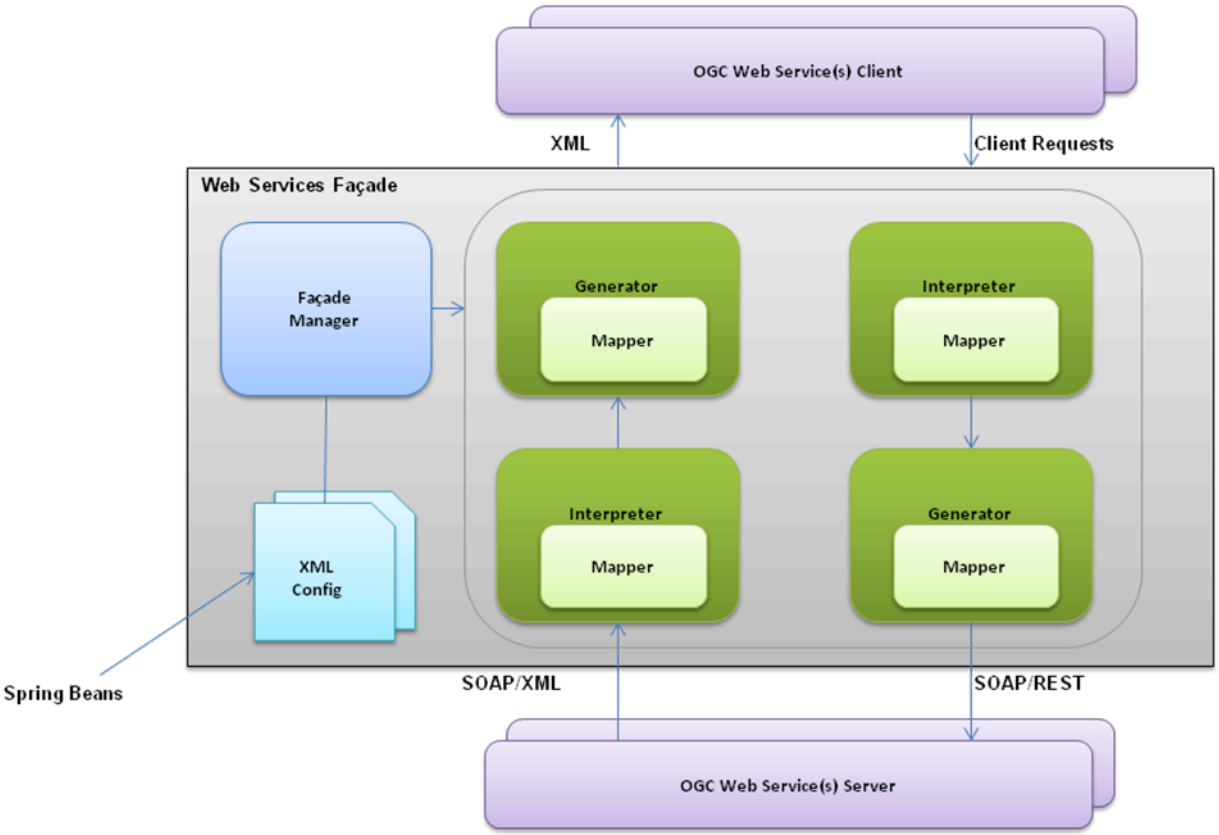
2 Web Services Façade Overview

The Web Services Façade is an extensible, open source tool, which supports translations between different protocols for a specific web service. For the OWS-9 testbed, it has been set up to translate between POST and SOAP services for a Web Feature Service. However, it can be configured to support translations between multiple protocols, such as REST, SOAP, KVP, JSON, as well as supporting multiple web services.

It is based on Spring Beans configuration management and Maven dependency management which makes configuration of new proxies simple and flexible.

2.1 Web Services Façade Design

The following diagram shows the design of the Web Services Façade (WSF).



Clients send the request to the WSF as if it were the OGC service, the proxy is initialized by recognition of the context URL. From here mappers provide the knowledge of specific translations between models. They are used by generators to map from a model to a request or response. Interpreters use mappers to map from a request or response to a model. Generators and interpreters understand the transport layer – HTTP requests and responses. Mappers understand the content transported – operations and resource identifiers.

Models are generic containers for information obtained from a request or response. All mappers support translation to / from a common model format using a simple dictionary array. This allows additional translation combinations by plugging the translations together via the common model format, for instance we could translate KVP to SOAP or XML to KVP using the common model objects interchangeably.

Model objects can be extended if the generic model won't support a specific implementation, but this precludes the model from being useful to mappers that don't support the extended version.

Below is a sequence diagram which details the flow process of the interpreting proxy

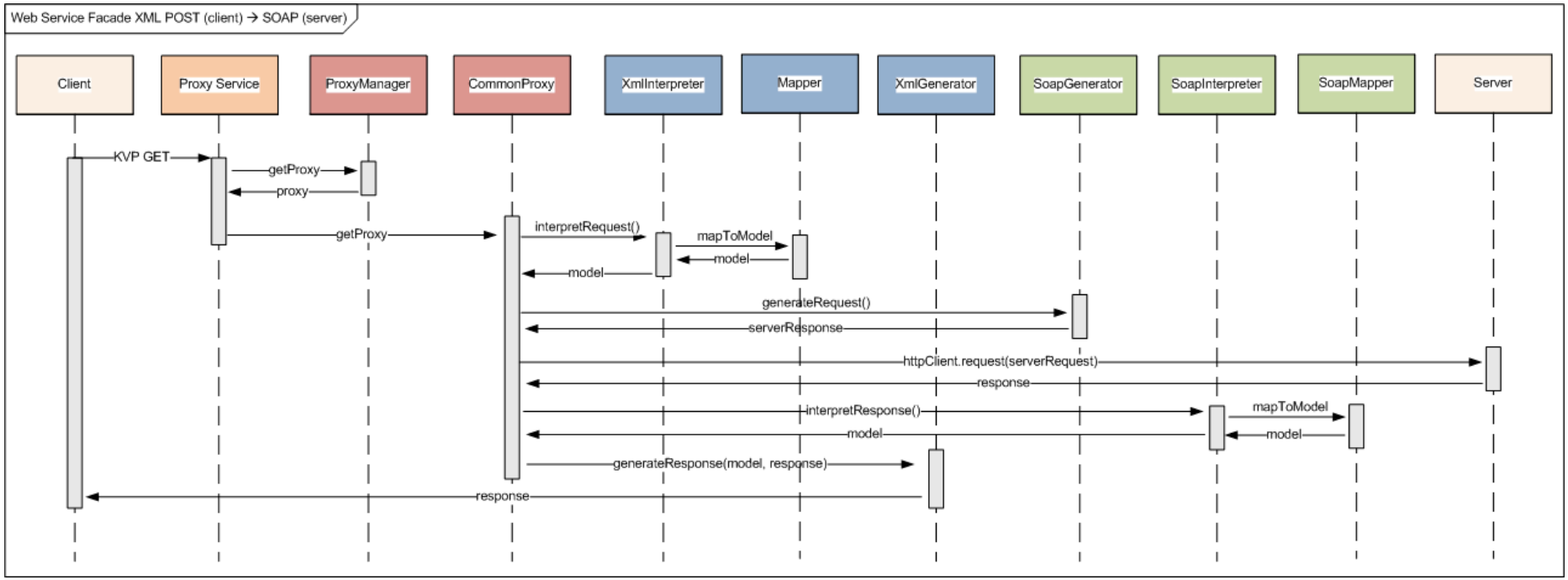


Figure 1: Flow diagram of proxy control

2.2 Web Services Façade Configuration

An example of what the Web Services Façade configuration would look like:

```
<!-- XML POST to SOAP using Generic interpreters -->
   <bean id="xmlToSoap" class="com.lisasoft.wsfacade.proxies.Proxy">
      <property name="proxyContextUrl" value="/postXML" />
      <property name="proxyManagedUrls">
         <map>
            <entry>
               <key>
                  <value>?WSDL</value>
               </key>
               <value>xml/wfs_2_0_wsdl_template.xml</value>
            </entry>
         </map>
      </property>
      <property name="name" value="xmlToSoap" />
      <property name="serviceRequestType" value="post" />
      <property name="serviceUrl"
         value="http://services.interactive-
instruments.de/xsprojects/ows9-tds/services/ltds/wfs" />
      <property name="clientRequestInterpreter"
ref="postXmlBodyInterpreter" />
      <property name="clientResponseGenerator"
ref="xmlClientBodyGenerator" />
      <property name="serverRequestGenerator" ref="xmlBodyGenerator" />
      <property name="serverResponseInterpreter"
ref="kvpXmlBodyInterpreter" />
   </bean>
```

Multiple proxies can be defined in one context to support multiple services and translations. For instance if the translations were supported you could specify one proxy to translate WMS KVP service "A" into SOAP while a different proxy translates the same service into JSON. You could then specify a third proxy to translate WFS SOAP service "B" into KVP. Each proxy is named and this name forms part of the proxied service URL to allow clients to identify which translating proxy of the WSF proxy instance to use. The proxy manager handles delegation of incoming requests to the appropriate proxy.

The proxyManagedUrls property is a key/value paring of URLs which are controlled at the proxy level and are not pushed to the actual service. This can be used in a number of ways – for instance to override the capabilities document or in the case of the example proxies to publish a WSDL document describing the SOAP web service bindings.

Each proxy configuration needs the connection details to the service and the type of service (rest, soap, etc). The type of client is also specified. The above example overrides

the default mappers for custom WFS specific ones that do things like injecting SOAP service details into a GetCapabilities response.

A basic security provider can also be added to the proxy and if present will be initialized when the proxy is about to send a request to the endpoint service. This mechanism is a small addition to the project and has not been thoroughly tested or contributed to.

Below is the flow diagram of the provided basic security layer:

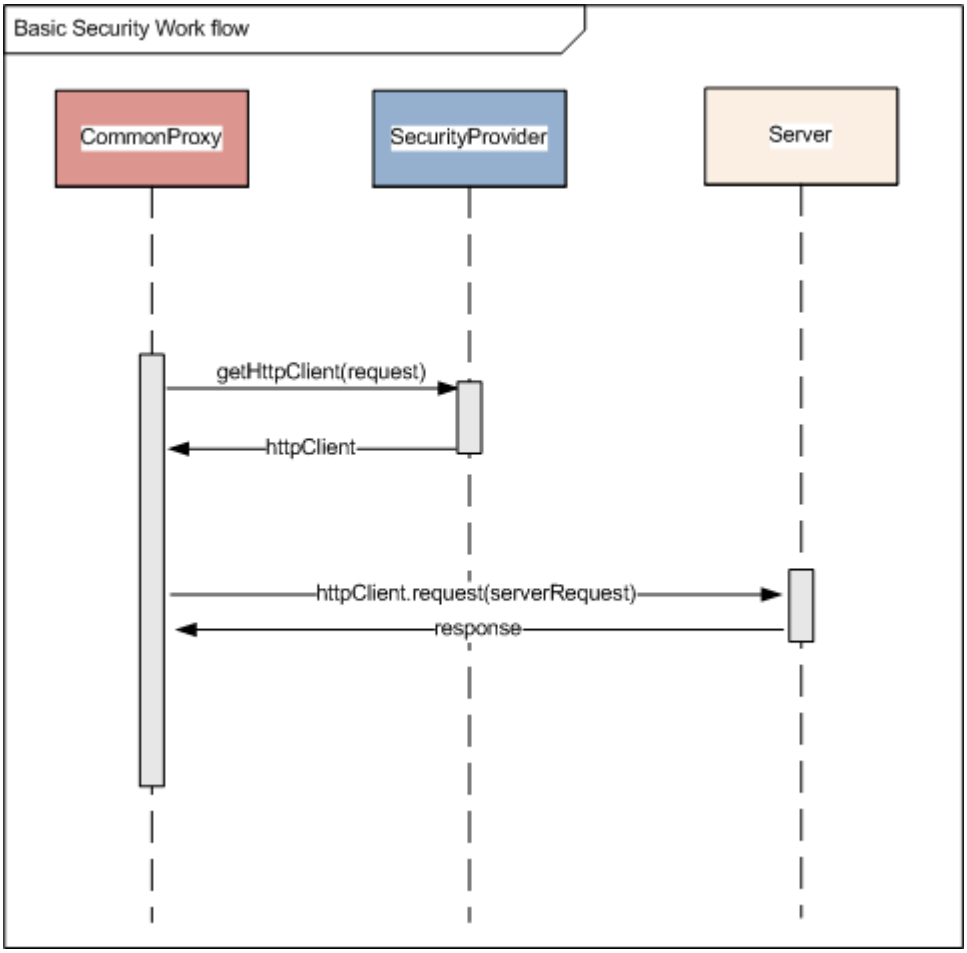


Figure 2: Flow control of security provider mechanism

This security mechanism has several missing elements surrounding https connections but at its simplest offers basic security and at its best could be extended to inject external security broker data to the outgoing request. This idea was first introduced in OWS-5 test bed by Bastian Schäffer. Below is an illustration of SOAP security injection at the proxy level published by Bastian Schäffer.

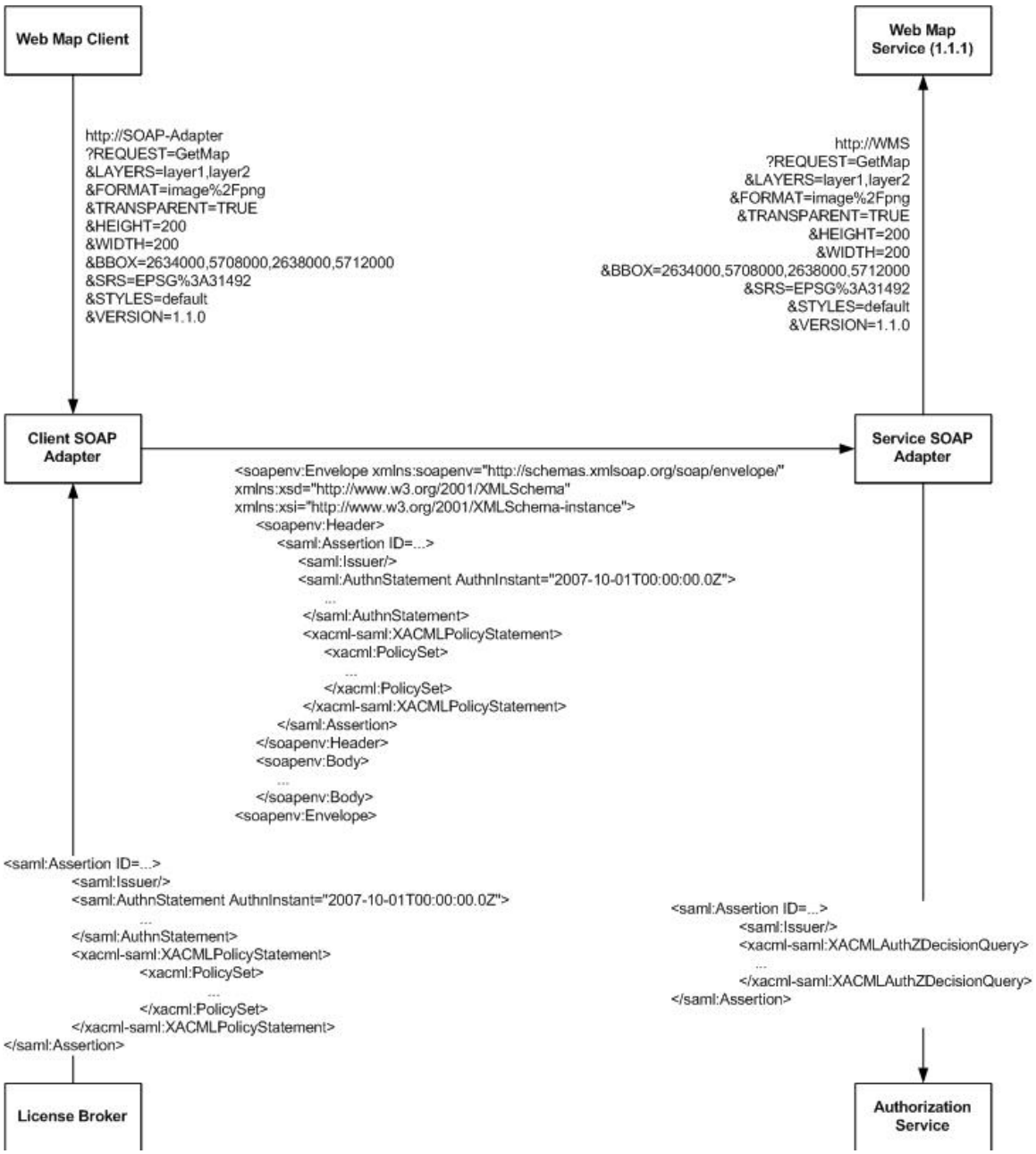


Figure 3: Adding a License Token to the SOAP Request (referenced from OWS5)

2.3 Web Services Façade Project Scope

LISAsoft has developed a POST->SOAP Web Services Façade for a WFS which will cover a limited set of use cases, including:

- Translation of the request from XML POST to SOAP
- Translation of the response from SOAP back to XML
- Adjust the capabilities document requested to include SOAP binding information and advertise a WSDL
- Simple support for a request for the WSDL at the proxy
- Provide a pre-configured bundle of the code, under an Open Source license

The project builds on work done in OWS8 by LISAsoft's James Groffen. The quality of the software has been improved by re-organising the code to use Maven dependency management and adding unit testing.

- The structure of the project is very open source like with modules which are built using maven.
- The core module is the core interfaces and major concrete objects – such as generic mappers and generators.
- The wmts module is legacy code from the previous work and serves as an example of how one can create specific service wrappers
- The test-harness module is a fully contained web application which uses the other modules for testing
- The web module is the main web application which wraps the entire application into a deployable WAR

The project is built on Spring Beans. This method of configuration and wiring allows the project to be extremely flexible and allow any possible combination of proxy. The implemented proxy is a POST XML to SOAP and back again proxy binding which is testing the OWS9 data endpoint:

3 Web Services Façade Outcomes

3.1 The following was achieved as part of OWS-9

- POST XML to SOAP proxy
- Code is in a github repository
  - https://github.com/lisasoft/wsfacade
  - Code is licensed under GPL v3
  - Project wiki pages are up
    - https://github.com/lisasoft/wsfacade/wiki/Design
- Application has been moved to Spring context loading for 'pluggable' architecture

3.2 The following bonus extensions were added by LISAsoft

- Added jUnit test architecture to project
- A Security interface layer added
- ISecurityProvider is an interface which allows for pluggable security (through String)
- Example concrete security provider written BasicSecurityProvider

3.3 Future work

More configuration work needs to be done in the KVP to SOAP mapping to fully see a functioning proxy for KVP. This is due to the way OGC has implemented the standard approach to SOAP by embedding standard OGC XML in a SOAP Envelope.

Unfortunately KVP to SOAP mappings cannot be generated generically as the SOAP mapper needs to map the parsed KVP key/value pairs into XML. This is non-trivial and code needs to be developed in order to create this proxy mapping.

Security is limited in this design as it interacts solely with an unauthorized user, the provided security is weak at best but could be expanded upon in later test beds.