TEST BED GEO-SERVICES GEONOVUM

COMMISSIONED BY	:	Geonovum ICTU (programme RENOIR)
COMMISSIONED TO	:	Atos Origin
AUTHOR(S)	:	Atos Origin Project Team
EDITORS	:	Elwin Koster (Atos Origin) Tom Visser (Atos Origin)
REVIEWED BY	:	Thijs Brentjens (Geonovum) Bart van den Eijnden (Geonovum) Olaf van Gorp (ICTU) Michel Grothe (Geonovum) Clemens Portele (Interactive instruments) Marcel Reuvers (Geonovum) Frank Terpstra (ICTU)
VERSION	:	1.0
STATUS	:	Final
DOCUMENT DATE	:	16 February 2010
NUMBER OF PAGES	:	72

Contents

1 Introduction

This report contains the results of a project named 'Testbed geo-services op de OSB (Overheids-servicebus) en register' (i.e. test bed for geoservices on the OSB (Dutch Governmental service bus) and register). Note that the OSB is a set of standards, not a physical service bus, as the name could suggest[1]. The OSB profile concerned is 'WUS' (WSDL, UDDI, SOAP), version 1.1 (OSB 2008-a). The ultimate aim for this project is the embedding of geo-services in the Dutch 'e-government'.

During the project a test bed was developed to investigate, if two important geo-services, Web Map Service (WMS) and Web Feature Service (WFS), can be invoked according to the OSB standards.

At first, the scope of the project is described in more detail (chapter 2). Chapter 3 contains some background information on the OSB, the Dutch key registers and the geostandards that apply to the project.

Chapter 4 describes the preparations that were carried out in order to be able to specify and run the tests. The test bed architecture and the way the test bed was implemented is described. The chapter also contains a discussion on SOAP/WSDL in the context of geo-services.

The test results, including a large number of listings, are presented in chapter 5. The research questions are answered in chapter 6. In Chapter 7 the conclusions and recommendations are given. Chapter 8 contains a list of references. The report concludes with a number of appendices.

Initiators of the project were Geonovum and ICTU (the ICT department of the Dutch Government). The project was carried out by Atos Origin in the period October 2009 – January 2010 with the support of both Geonovum and ICTU.

[1] Also note that the name 'OSB' is in the process of being replaced by 'Digikoppeling'.

2 Scope

2.1 Background

ICTU, Logius[2] and Geonovum are co-operating to promote the embedding of 'geo' in the Dutch 'e-government'. Over 70 activities were defined in a national service delivery program. Ten of them are directly related to 'geo' and are assigned to one of the three organizations mentioned above.

Geonovum is, amongst others, responsible for the following two activities:

[A] Geo-services on the OSB;

[B] Interaction between the national geo-register (NGR) and the OSB register.

2.2 Problem definition

For both activities, mentioned above, Geonovum has created a work package in this project. Work package A is the subject of this study; for work package B a separate document will be published.

Several key registers like BRT, BRK, BAG, BRO and BGT (see paragraph 3.3) contain geo-information. All registrations have to be made available in a way compliant to the OSB standards. These standards have not specifically been targeted at the international geo-standards or the European implementation of these standards. So, it is useful to investigate to which extent it is possible to fulfill the regulatory requirements and, where necessary, to propose additional activities to address open issues, e.g. submitting change requests to standardization organizations.

A technological summary of the problem definition can be formulated as:

- OSB WUS requires the use of WSDL/SOAP while only a few WSDL/SOAP bindings for WMS and WFS operations are specified in the standards[3]. How to fill this gap?

2.3 Objective

The aim of work package A is to gain experience in the use of WMS en WFS over WSDL/SOAP according to OSB WUS guidelines, OGC standards and INSPIRE guidelines. The result would be a better understanding of the way geo-services can become OSB compliant. The outcome could also lead to change proposals for the OSB.

The main result of the project is this report, giving the answers to a number of research questions (see below). A secondary objective was the development of a test bed that can be used in future presentations and tests. A description of the test bed and its results are also part of this report.

2.4 Use cases and research questions

Three use cases with their research questions are described in the original assignment for this project (Geonovum 2009). Use case 1 and 2 are part of this study, use case 3 was removed in an early stage of the project, because it was realized that it would not give additional insight.

Use Case 1 addresses the use of WMS over OSB according to OGC, ISO/TC 211 and INSPIRE. The research questions are[4]:

1. Is it possible to invoke WMS operations according to the OGC standards WMS 1.1.1 and OWS Common 1.2 and OSB 1.1?

[2] IT-governance department of the Ministry of Internal Affairs, formerly known as GBO.Overheid.

[3] For WMS no SOAP/WSDL bindings exist. A SOAP binding is specified in the WFS 1.1.0 standard. WSDL is not part of this standard; however, a WSDL is presented in the OGC schemas: http://schemas.opengis.net/wfs/1.1.0/wsdl/.

[4] Some questions from the original request were removed during the project.

2. Is it possible to invoke WMS operations according to the INSPIRE standards and OSB 1.1?
3. What are the differences re WSDL/SOAP according to OGC, INSPIRE and OSB in the context of WMS?
4. Why would you use WMS based on WSDL/SOAP for retrieving Geo-data?

Use case 2 adresses the use of WFS over OSB according to ISO/TC 211 and OGC. The research questions are:

1. Is it possible to invoke WFS operations according to ISO/TC 211 standards and OSB 1.1?
2. Is it possible to invoke WFS operations according to the OGC standards WFS 1.1.0 and OWS Common 1.2 and OSB 1.1?
3. What are the differences re WSDL/SOAP according to OGC, ISO/TC 211 and OSB in the context of WFS?

3 Context

3.1 Overheidsservicebus (Digikoppeling)

Overheidsservicebus
The Overheidsservicebus (Digikoppeling)[5] is a Dutch e-government standard that government organizations can use to implement electronic information exchange which is consistent with the Dutch Government Reference Architecture (NORA).

One of the cornerstones of NORA is 'exchanging information according to the principles of service oriented architecture'. In a service oriented environment, information is offered as a service, a service which describes in detail both the information which is offered and the conditions under which it is offered (and may be consumed).

In a service oriented environment, organizations act as information providers (or: service providers) or information consumers (or: service consumers), or even both. The actual exchange of information is carried out using electronic services (typically, web services technology), which in turn make use of an underlying messaging framework.

Overheidsservicebus provides standards that implementing parties can use to create an 'interface' that allows for standardized, authenticated and secure message exchange based on these electronic services.

The Overheidsservicebus standards cover electronic messaging between all types of government organizations, ranging from ministries to municipalities. In order to allow these organizations to exchange messages, it is important that the different internal software landscapes present within these organizations can interoperate with each other. This can be achieved by having organizations implement a so-called adapter, which converts their internal messaging formats to formats that comply with Overheidsservicebus standards (and vice versa).

An important underlying principle here is 'implement once, reuse many times'. An Overheids-servicebus compliant adapter implementation allows a government organization to communicate with any Overheidsservicebus-compliant service. Essentially, the Overheidsservicebus adapter provides the generic endpoint (or 'information exchange boundary') from/to an organization.

Thus, Overheidsservicebus may also be viewed as representing the common landscape between the information exchange boundaries of all government organizations exchanging compliant messages.

[5] From January 2010 onwards, the product name "Overheidsservicebus (OSB)" will be replaced by "Digikoppeling"

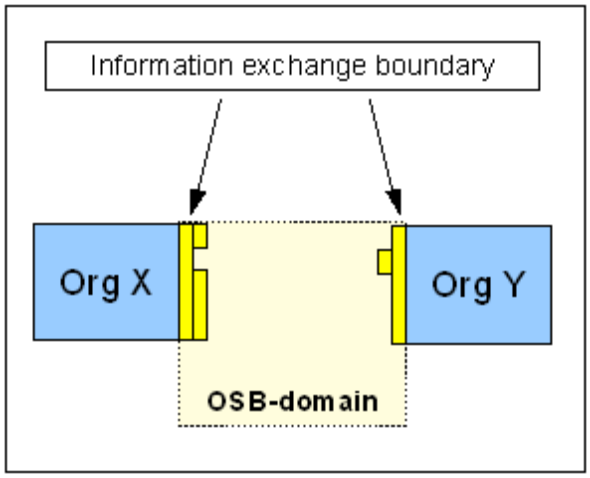

Figure 1 – OSB between information exchange boundaries

Products

Overheidsservicebus provides the following products:

- A set of standards that specify how the information exchange boundaries of an organization should be configured ("OSB Koppelvlak[6] standards"). The standards describe both synchronous request-response messaging (WUS) as well as asynchronous reliable messaging (ebMS);
- A number of software products that support this configuration: OSB Compliancy Facility, CPA Creation Facility, OSB Gateway;
- The OSB Service Register[7]. In order to allow parties to find information about the services they need, a service register is required. OSB Service Register is the central facility in which service providers can publish the details of their services in order to have them available to service requesters (consumers). Consumers can search the register for these details, and propose a service contract to the provider. Hence, OSB Service Register plays an important part in service lifecycle management.

Overheidsservicebus standards focus principally on the 'transport layer' of information exchange (also referred to as the 'logistics layer'). This layer is positioned between the layer which represents the physical network connectivity and the layers above, which represent the applications, the data and the (business) processes which are involved in electronic information exchange (see Figure 2).

The transport layer typically deals with aspects of communication like authentication, security, communication protocols, and so on.

Overheidsservicebus presumes the availability of a physical network facility (allowing traffic over both the internet and the Dutch 'Digi netwerk' network), and does not say anything about message structure, message semantics and message content insofar as these are not included in the 'Koppelvlak standards'.

[6] Koppelvlak is the Dutch word for interface.

[7] Currently called 'Digikoppeling Service Register'

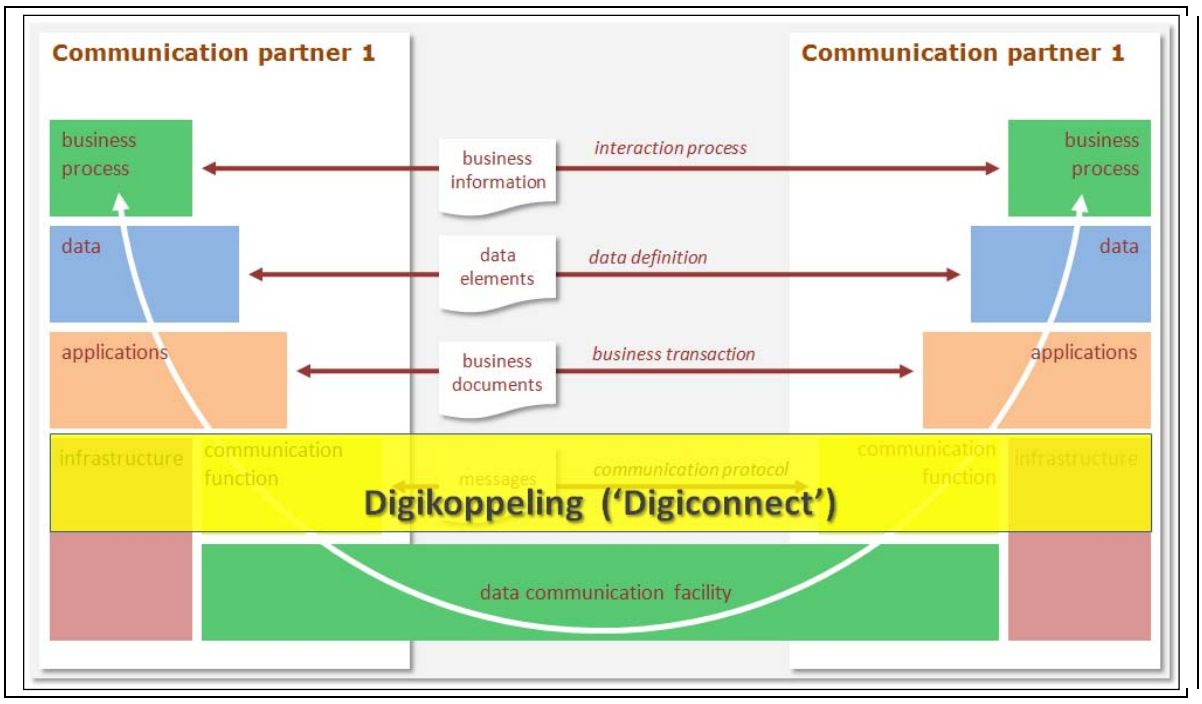

Figure 2 - Digikoppeling

3.2 Overheidsservicebus (Digikoppeling) koppelvlak standards

Two OSB koppelvlak standards have been designated and approved by the College of Standardization; these are the standards ebMS and WUS.

In this test bed only the koppelvlak standard WUS has been applied. EbMS is used for reports and mutations, messages for which a guaranteed delivery is important. WUS is used for queries and guaranteed delivery is not important. If no answer to an information request is forthcoming, a repetition of the query will suffice. Repeating the query is also sufficient for the use of Geo standards within the e-Government, as these are queries about Geo-information resources.

Koppelvlakstandaard WUS
WUS is an acronym for WSDL, UDDI and SOAP. What is meant here is a family of international standards of OASIS and W3C. They are often referred to as WS-*. These standards deal with application-to-application web services that can carry information over heterogenic middleware platforms and applications (and, in this way, gain access to the web services). All OSB web services that are based on WUS should conform to the koppelvlakstandaard WUS (OSB 2008-a).

WSDL – A web service is (partly) defined in a formal way and can be handled in an automatic way by the service description described in WSDL. This WSDL gives a description of the requirements necessary for communication. It is an abstract definition of the web service. The web service actually communicates with SOAP messages that are generated based on the WSDL.

Security – This koppelvlak standard mandates security on transportation level according to double TLS or SSLv3. This is being realized with PKI government certificates for which for the OSB a number of optional field have to be completed mandatory.

Compliance facilities

The OSB makes compliance facilities available to help government organizations to set up their services in the OSB domain. In this way they can verify whether the koppelvlak standard was implemented correctly.

The OSB compliance facility is a stand alone system which is accessible from the Internet. It consists of a component for WUS (for queries) and a component for ebMS (for reports). Each component offers a specific function for the service provider and service requester. This way the OSB compliance facility WUS makes it possible to test the implementation of the OSB koppelvlakstandaard WUS.

The OSB compliance facility WUS-WSDL describes the koppelvlak for communication with the OSB compliance facility WUS. A web service is implemented that is based on this (compliance service of the organization) or a web service client (compliance client of the organization). The idea behind this is: if the compliance client or service of the organization can communicate successfully in the OSB domain with the compliance facility, then the logistical layer can be applied too for their own web service or for a web service client.

OSB Gateway

To support organizations in the implementation of the standards of the OSB, the OSB Gateway has been developed. This OSB Gateway is a facility that is placed in organizations, or in a department of the digital mailroom, to zip and unzip messages. Both the forwarding and receiving of messages from other organizations then occurs via the OSB koppelvlakken. In consultation with the Ministry of the Interior and Kingdom Relations, it was decided to develop OSB Gateway as open source software. Implementations and local modifications are done by individual organizations and the market. Version 2.0 of the OSB Gateway is now available.

The OSB Gateway:

- can process 7200 messages on average per hour;
- can deal with a higher periodically top load (16 messages per second or more) over a shorter period of time (minutes);
- can deal with WUS messages till 20MB (large messages seriously increase throughput times);
- can be implemented double in a redundant setup for higher availability, however increasing capacity (load balancing) is not possible with double implementation;
- can have its platform installed independently in a Linux or Microsoft Windows environment.

3.3 Key registers

The OSB takes care of the traffic of messages for the key registers. A number of key registers have geometry which can be ascribed to the “geo-domain”. These key registers are viewed here with a globe.

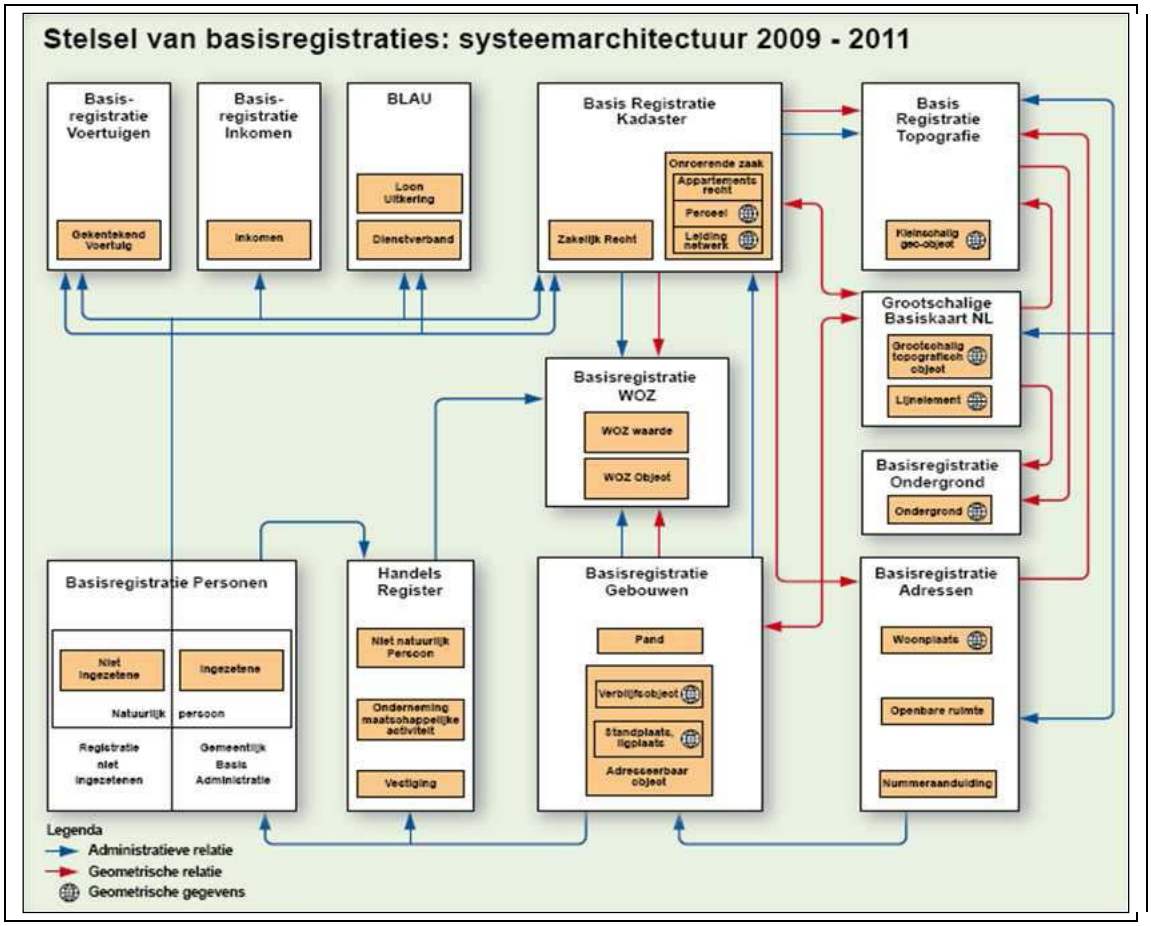

Figure 3 – Key Registers

Key register		Content standard
1st tranche		
BGR	Basis Building Register	StUF, BAG[8]
BRA	Key register Addresses	StUF, BAG
BRK	Key register Land Registry	NEN3610, IMKAD
BRT	Key register Topography	NEN3610, TOP10NL
2nd tranche		
BGT	Key register Large ScaleTopography	NEN3610, IMGeo
Candidate 3rd tranche		
BRO	Key register Underground	NEN3610, IMBOD
Candidate 4th tranche		
WOZ	Key register WOZ	StUF, WOZ

For the standards a distinction is made between source keepers and consumers. It was recently agreed that the “StUF-family” and “NEN3610-family” (NEN3610, information models, GML, WMS and WFS) should be used for the key registers in their contact with the consumers. In this way having to register a standard per key register is avoided.

[8] The recording of the co-ordinates conforms to NEN3610.

Research into the mandatory standards for the consumers of key registers was done by the programme “Common accessibility van key registers (GOB)”. One of the GOB standards is the “Standaard Berichtstructuur”. This standard is predominantly important for the functioning of the key registers as a system, as the key registers no longer function as separate systems, but are integrated:

- A lot of consumers retrieve data from various key registers;
- Key registers (and definitely key registers that do not belong to the first tranche) also provide data from other key registers;
- Key registers are unlocked altogether, so consumers do not need to communicate with each separate key registration nor do they need to combine data from various key registers.

If consumers query various key registers and receive replies or mutations from several key registers, it is important that the systems used by the consumers for communications with all the key registers are able to speak the same "language". The key registration Income, for example, can use the same message structure for personal data as the GBA. This is not the case at the moment; consumers need separate interfaces for, for example, communication with the GBA, Land Registry, Trade Register etc.

With regard to the developments above, it is important that the logistical layer of the standards for geo-information can be handled by the OSB. Logistics occur within the standards that were developed at international level; Web Map Service (WMS) and the Web Feature Service (WFS).

Those standards are filled in semantically for Europe (INSPIRE) and the Netherlands (NEN3610). In the Netherlands this development is guided by Geonovum. These standards are in the process of being adopted by the College of Standardization onto a list of open standards.

3.4 Geo standards

In this report, a number of geo standards are referred to. In this chapter they are briefly introduced.

3.4.1 Web Services Common

> The OpenGIS® Web Services Common (WS-Common) Interface Standard specifies parameters and data structures that are common to all OGC Web Service (OWS) Standards. The standard normalizes the ways in which operation requests and responses handle such elements as bounding boxes, exception processing, URL requests, URN expressions, and key value encoding. Among its uses, this document serves as a normative reference for other OGC Web Service standards, including the OpenGIS Web Map Service (WMS)], Web Feature Service (WFS), and Web Coverage Service (WCS) standards. Rather than continuing to repeat this material in each such standard, each standard will normatively reference parts of this document.[9]

The version of the standard that this document refers to is 1.2 (OGC 2009). This does not yet have an official status, but it contains OGC’s most elaborated vision on the use of SOAP/WSDL.

[9] http://www.opengeospatial.org/standards/common

3.4.2 Web Map Service

The OpenGIS® Web Map Service Interface Standard (WMS) provides a simple HTTP interface for requesting geo-registered map images from one or more distributed geospatial databases[10]. A WMS request defines the geographic layer(s) and area of interest to be processed. The response to the request is one or more geo-registered map images (returned as JPEG, PNG, etc) that can be displayed in a browser application[11]. The interface also supports the ability to specify whether the returned images should be transparent so that layers from multiple servers can be combined or not.[12]

Web Map Service versions 1.1.1 (OGC 2002) and 1.3.0 (OGC 2003) are subject of use case 1 in this report.

The WMS standard provides three different operations:

- GetCapabilities requests the possibilities offered by the WMS Service. The response is an XML-message (the Capabilities document). It contains for example the coordinate systems offered as well as the available layers.
- GetMap is the way to request a specific map. Parameters that can be used in this request are e.g. coordinate system and layers.
- GetFeatureInfo is an optional operation. This operation is used to request additional information about geo-objects (features) at a specified location in a raster image (retrieved by a previous GetMap operation).

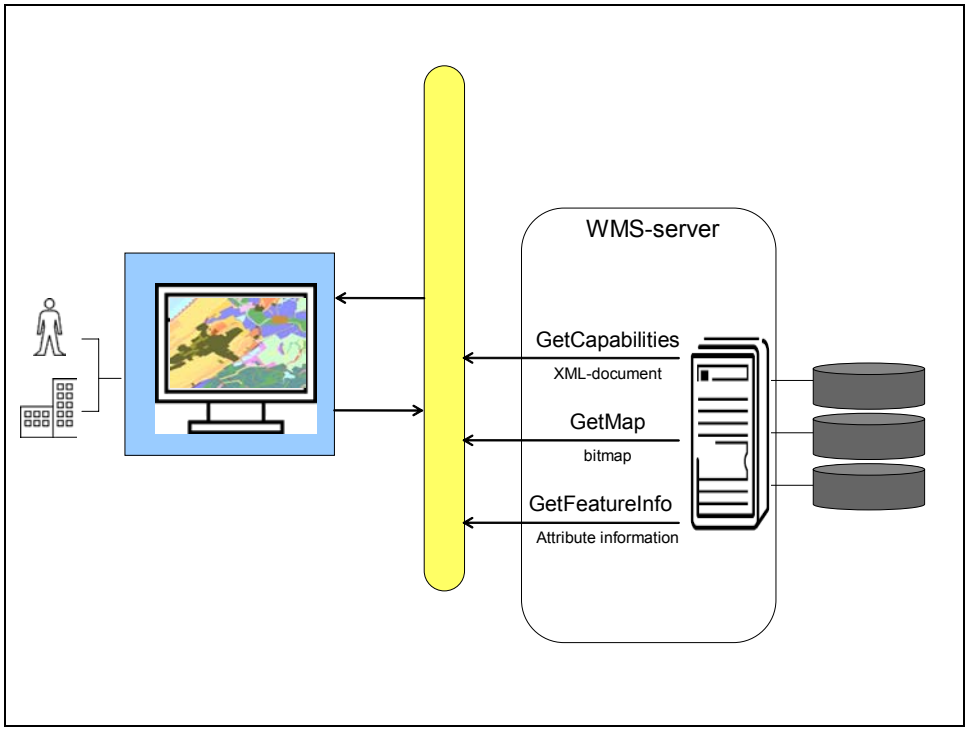

Figure 4 – WMS Services

[10] Of course, other data sources are also possible.
[11] Of course, desktop WMS clients are also possible.
[12] http://www.opengeospatial.org/standards/wms

Version 1.3.0 can be seen as the current version. However, version 1.1.1. is probably still used more commonly. Version WMS 1.3.0 is since 2005 an ISO 19128 standard. Since there are possibilities for local varieties, a Dutch version of the standard exists based on WMS 1.1.1 (Geonovum 2007). In this Dutch variant specific items are the file format (PNG), projection (RD) and the mandatory support of GetFeatureInfo.

In October 2009 the EC published the commission regulation on network services. In annex III the view service is described. This commission regulation is worked out in an “Technical Guidance to implement INSPIRE View Services”. The technical guidance refers to WMS 1.3.0. Therefore, the 1.3.0 version will be the designated standard by 2011. The Netherlands will make the transition to version 1.3.0 in 2010.

3.4.3 Web Feature Service

> The OGC Web Feature Service allows a client to retrieve and update geospatial data encoded in Geography Markup Language (GML) from multiple Web Feature Services. (OGC 2005-a)

Web Feature Service (WFS) is an interface for retrieval, publishing, editing and analysis of geographical vector data. WFS uses Geography Markup Language (GML) for data transfer. Objects that are a result of a WFS request are delivered and presented in GML.

WFS contains quite a number of operations. In the context of this document the interesting operations are

- GetCapabilities, which has the same function as the WMS operation with the same name;
- DescribeFeatureType, which offers detailed meta-information about a feature in the form of an XML schema;
- GetFeature, which returns information about selected features. The query language used is Filter Encoding. With this query language spatial queries can be defined. Examples of Spatial Queries are “From a dataset with buildings, present all officebuildings within a range of 200 meter from a road (a line geometry) . An example is shown in the following partial listing:

```
<ogc:Filter xmlns:ogc="http://www.opengis.net/ogc" xmlns:gml="http://www.opengis.net/gml">
  <ogc:And>
    <ogc:DWithin>
      <ogc:PropertyName>building_geometry</ogc:PropertyName>
      <gml:LineString srsName="http://www.opengis.net/gml/srs/epsg.xml#28992">
        <gml:posList>204500.0 612100.0 204000.0 611600.0 204000.0 611100.0</gml:posList>
      </gml:LineString>
      <ogc:Distance units='m'>200</ogc:Distance>
    </ogc:DWithin>
    <ogc:PropertyIsEqualTo>
      <ogc:PropertyName>building_type</ogc:PropertyName>
      <ogc:Literal>office</ogc:Literal>
    </ogc:PropertyIsEqualTo>
  </ogc:And>
</ogc:Filter>
```

Listing 1 - Example Filter encoding

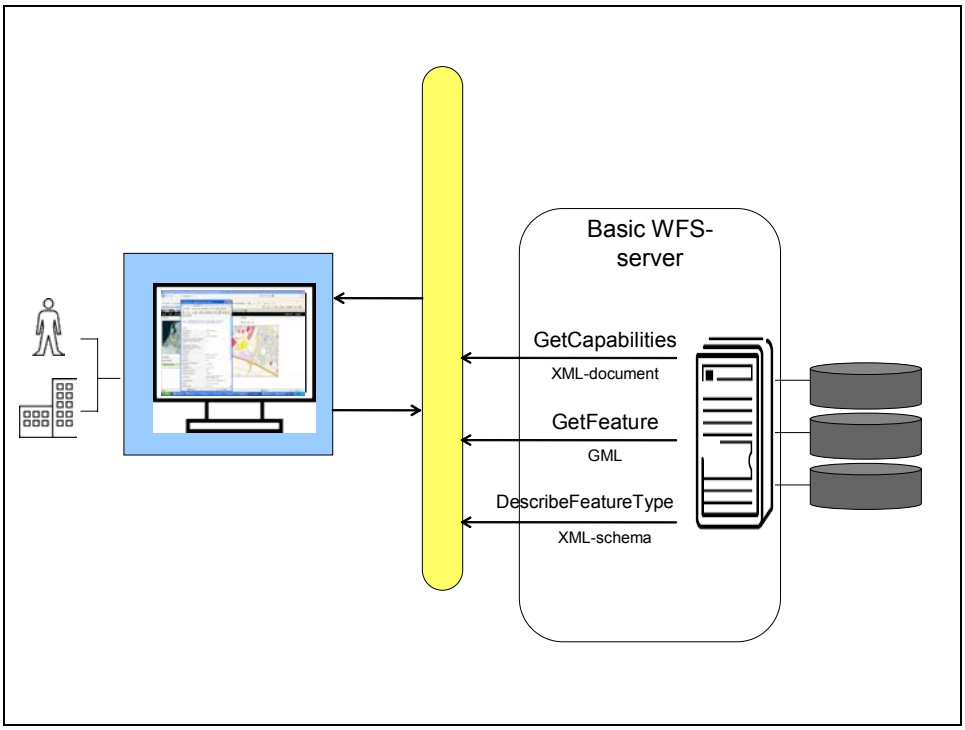

Figure 5 – WFS services

The current version of WFS is 1.1.0 (OGC 2005-a). Its related Filter Encoding specification is (OGC 2005-b). Both specifications were submitted to ISO/TC 211 (ISO/TC 211 2009-b) and (ISO/TC 211 2009-a) respectively, probably resulting in OGC WFS version 2.0.

WFS 1.1 requires implementations to support GML 3.1.1 , that is used by the Dutch standard base scheme for geo-information (NEN 2005) and the Dutch information models. In addtion, WFS 1.1 allows implementations to support other versions of GML and output formats.

Because of a number of specific circumstances and degrees of freedom there is a specific Dutch profile of Basic WFS, based on WFS 1.1 (Ravi 2006).

4 Preparations

4.1 Test bed architecture

During test bed preparation, we realized that there are multiple ways to convey service requests and responses as SOAP messages. The first approach can be designated as the 'infrastructural' way. This approach is illustrated by the following figure:

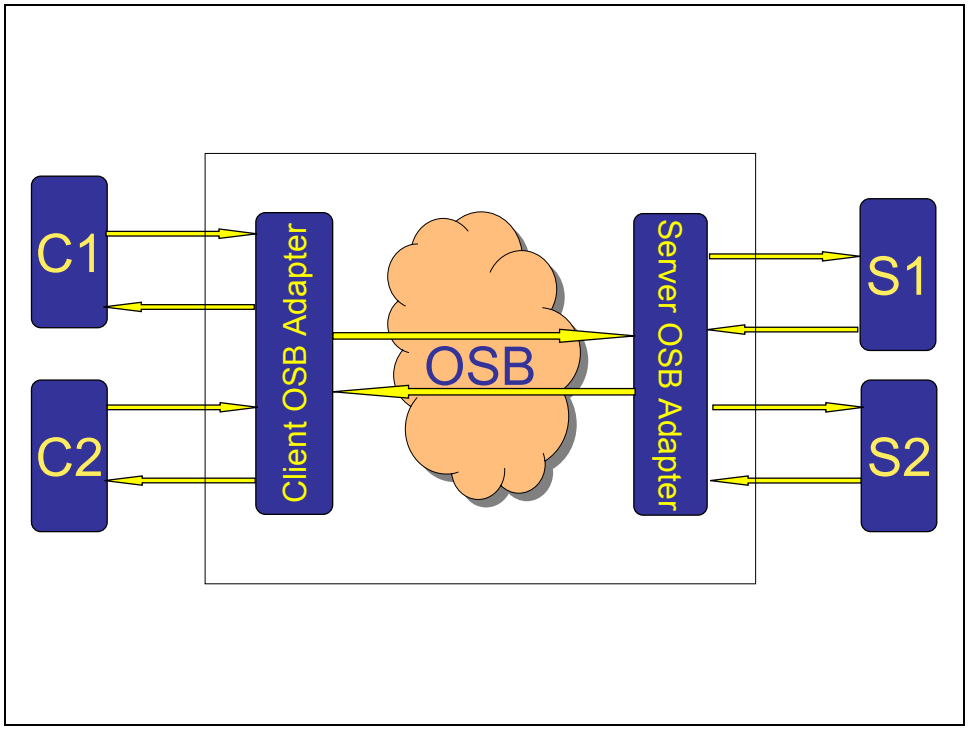

Figure 6 - Infrastructure oriented architecture

Clients (C1 and C2) and Servers (S1 and S2) do not communicate directly, but by means of a pair of intermediate software entities, the OSB Adapters. This architecture enables the transport of any information content, so also of WMS/WFS content. The OSB Adapters, which are part of the infrastructure itself, wrap this content into a SOAP envelope according to OSB standards and convey it to a second OSB Adapter.

This architecture was chosen for use case 1. During implementation, however, we realized that there are a few major concerns with respect to this architecture.

1) It is not the WMS/WFS service itself that is transmitted, but a technical service, that also contains information that is only needed for communication purposes. The WMS/WFS part is not subject to any validation but conveyed 'as-is'. On a functional level, there is no awareness whatsoever that communication is taking place according to OSB standards. Service discovery by means of the OSB service register is not really applicable.
2) To explain the second concern, it is necessary to show the architecture with multiple client and server organizations:

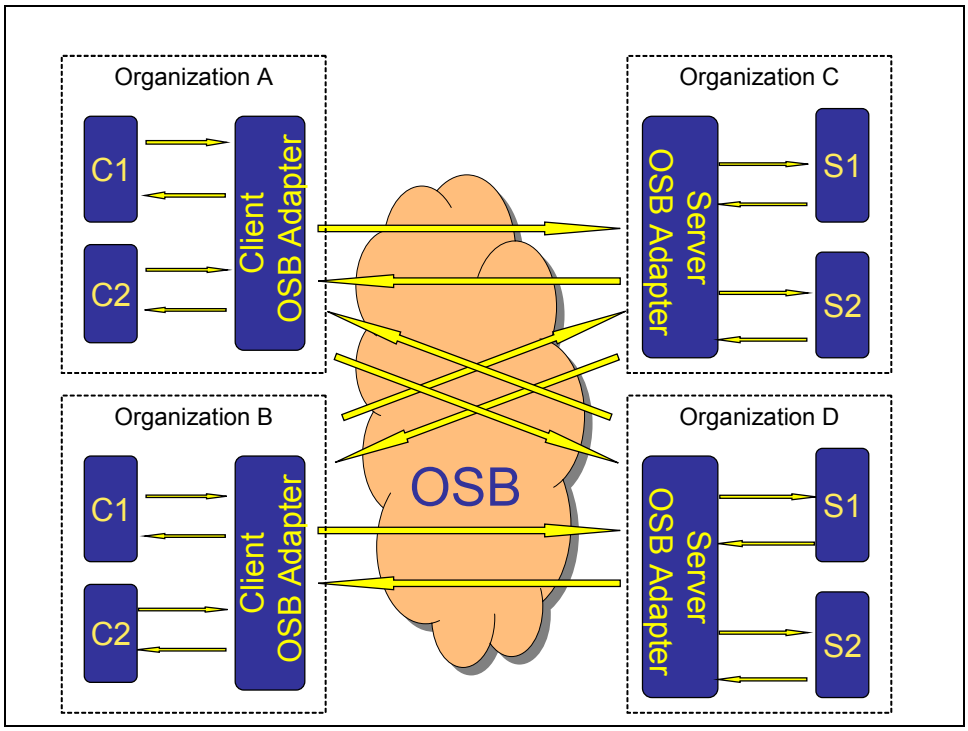

Figure 7 - Infrastructure oriented architecture with multiple organizations

A client OSB Adapter has to know the address of the server OSB Adapter, that gives access to the WMS/WFS service, that the WMS/WFS client addresses. For this to work, it would be necessary to create a directory service for these infrastructural services. Since the services are purely infrastructural, this directory service would be completely separate from the OSB Service Register. This would be a high price to pay.

3) It appeared from the tests that it is not possible to convey the messages completely unmodified; some insignificant HTTP Header information is lost.

It can be concluded that the infrastructural architecture cannot be regarded as a solution suitable for a production environment. Still, using this architecture in the test bed is instructive.

A second approach, called the functional architecture, is sketched in the following figure:

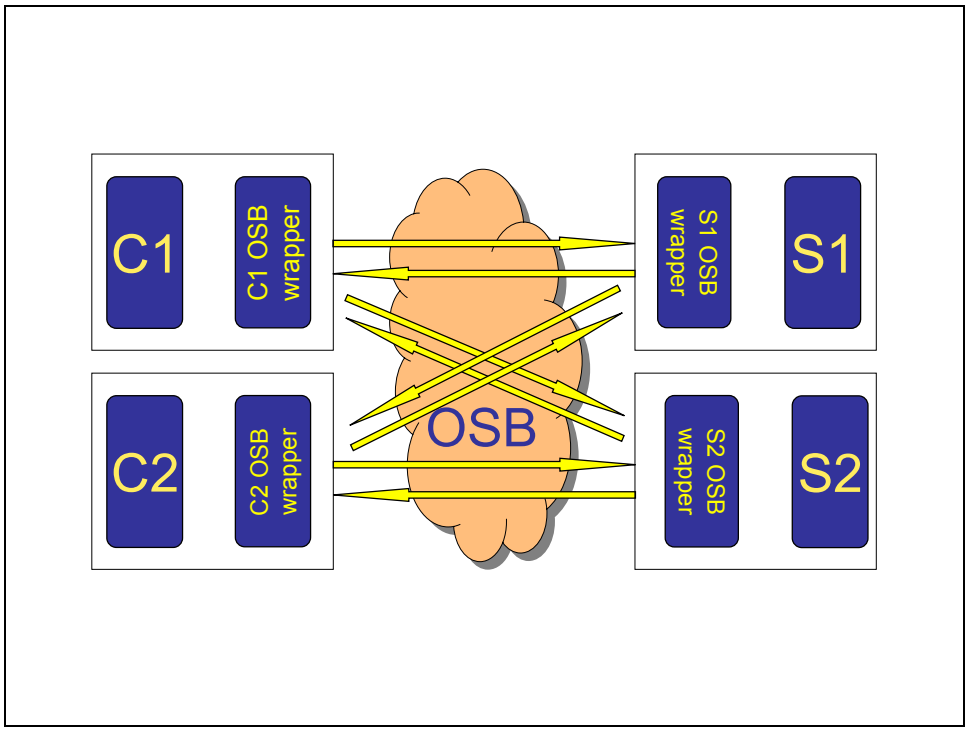

Figure 8 - Functional architecture

Here, clients and servers communicate directly by means of SOAP messages. OSB translation is performed by the individual clients and servers themselves. However, current WMS Client and Server software products (C1, S1, etc.) are not SOAP based, so OSB wrappers are needed to make them look like 'OSB clients' and 'OSB servers'. Possibly, in future these wrappers would not be needed anymore.

In this solution, the OSB wrappers are part of the clients and servers themselves, so they act on the functional level. The server addresses are made public in the OSB Service Register, so service discovery is taken care of. In this approach, the contents of the SOAP Body are exactly according to the WMS/WFS signatures. A limiting factor is the fact that not for all WMS operations a standard XML encoding is available. E.g. for a GetMap request, a proprietary encoding will have to be defined before being able to send it by means of a SOAP transport.

4.2 Technical details

4.2.1 General

Tooling

The tools which are used during development are the following:

- Eclipse (including plug-ins M2Eclipse, Tomcat plug-in, Subversive)
- Maven 2
- Subversion
- TortoiseSVN
- SOAP UI
- WebScarab

Components
The application consists of the following components:
- J2E 1.6
- Apache HTTP Components
- DOM Parser
- JAX-WS

Environment
The applications are deployed in an environment containing the following components:
- Apache Tomcat Servlet Container
- Windows or Ubuntu
- QuantumGIS (as a geo-client)
- Mapserver 5.4.2 (WMS for use case 1)
- Geoserver 2.0.0 (WFS for use case 2)
- Shapefiles (use case 1)
- PostgreSQL 8.4.1 / PostGis 1.4.0 (use case 2)

Security
- PKI Certificates
- Mutual authentication

4.2.2 Use case 1

Motivation general architecture
One of the requirements of use case 1 is that, for both client and server, existing software had to be used to implement the solution. Therefore the architecture that was designed had to be able to route the communication between a client application and a server application. This without altering the request received by the WMS server and the response received by the client.

A way to route data that is commonly available to many clients is a configurable proxy server. Because of the general availability of these techniques the decision was made to create a HTTP proxy. The main advantage to this solution is that it enables transparent routing of requests and responses, without the client- and server applications having any knowledge about the OSB-standards.

Technical implementation
To enable easy deployment of the proxy, it was implemented as a servlet. This servlet receives the HTTP request from the geo-client. From this request it extracts the required information, such as host name, port number and request parameters. This information is then parsed into an XML DOM document, forming a valid OSB compliant SOAP message. Using the connectivity API of Apache HTTP Components framework this message is sent to a configurable web service endpoint.

When the web service receives the SOAP message it uses XPath expressions to extract the information it needs and uses this to create a new HTTP request. The new request is identical to the original request from the geo-client. This request is then sent to Mapserver, again using Apache HTTP Components. The response from Mapserver is copied into the body of an OSB compliant SOAP response message and sent back to the proxy servlet.

If the response from Mapserver contains an image, for example with an GetMap response, this image cannot be directly added to the SOAP message. The OSB 1.1 standards prohibit the use of SOAP with Attachments. To still be able to send binary data, the image is first encoded using a

Base64 encoding algorithm. The resulting ASCII string is added to the response and sent back to the proxy.

The proxy servlet receives the SOAP response message and extracts the Mapserver response from the SOAP response using XPath expressions and sends it back to the geo-client. If the response contains encoded binary data, the data is decoded back into the original binary data first. If an error occurs during communication between the OSB components (the proxy and the web service) this is regarded as a communication error. Thus a timeout shall occur in the geo-client. If an error occurs because of a false request, e.g. an incorrect bounding box, the request is passed on to Mapserver which returns an error. This error is then passed back to the geo-client, keeping the OSB communication fully transparent.

The communication described above is synchronous. The application flow as described above is added to Appendix E as sequence diagrams.

4.2.3 Use case 2

Motivation general architecture
Because use case 2 consists of a custom geo-client and WFS server, a more functional approach is possible. The WSDL describing this web service has to adhere to a combination of three standards: OGC, INSPIRE and OSB.

The WSDL contains all elements of the WFS XSD's as described by the OGC. For this proof of concept, only the GetCapabilities and the GetFeature requests have been implemented. The availability of formal XSD's created the possibility of server-side validation. The use of a more functional WSDL enables better (automated) discovery by consumers of the web service.

Technical implementation
For this use case we have created a simple graphical user interface, to enable the user to send requests to the server and view the response message. This geo-client connects directly to the JAX-WS web service. The sent request is OSB compliant. The interface consists of a JSP page with a servlet that controls the communication with the web service.

The WFS web service publishes two operations. One for GetCapablities and one for GetFeature. After receiving the SOAP message from the client, the body is extracted using XPath expressions. This content is then processed as in any standard WFS server. Finally the response data is added to the body of a SOAP message and returned to the client.

The client uses XPath expressions to extract the body from the SOAP message and displays it to the user. SOAP Faults are also handled and displayed by the client. The communication described above is synchronous. The application flow as described above is added to Appendix E as sequence diagrams.

4.2.4 Security

The OSB standards require that all communication is securely encrypted using TLS / SSLv3. Any party connecting using OSB standards also must implement mutual authentication. To achieve this, the use of both client and server side PKI certificates is required.

The certificates used in the test bed were issued by a Certificate Service Provider of the Dutch government. The certificates used in the test bed were only suited for testing. They can not be used in production environments.

The implementation of TLS and SSLv3 is mainly a configuration issue. In Apache Tomcat this is configurable in the 'server.xml' file. There you configure the keystores and truststores that the server can use. We used a PKCS12 keystore and a JKS truststore for both server and client.

4.2.5 Availability

The testbed implementation of use case 2 has been registered in the OSB Service Register[13], see (OSB 2010-a), as "wfsService" within the business service "AtosOrigin_GeonovumTestBed_v01" (version may vary). Its usage details (partly based on the WSDL definition) and the compliance document can be found there.

To test this implementation end-to-end, a simple web application is available. It offers a number of text boxes, pre-filled with the GetCapabities and GetFeature example requests from this report. These requests are sent to the wfsService, mentioned above, either using the OSB standards or directly. Its location is http://212.159.196.130/tgmWebserviceClient/ . The communication, that takes place between the client web application and the wfsService, is logged in files that can be retrieved from http://212.159.196.130:81/logs/ : tgmWebserviceClient.log and tgmWebservice.log.

To test the implementation of use case 1, requests from a WMS client have to be routed through a proxy, that is located at 212.159.196.130, port 80. The logfiles concerned are tgmProxy.log and tgmServer.log, also available at http://212.159.196.130:81/logs/ .

On a weekly basis, the logfiles are archived in an appropriately named folder and cleared.

4.3 Analysis: SOAP/WSDL in OGC and INSPIRE

OGC and web service bindings

Historically, the GI community has its own more or less independent position within the generic IT world. This is also true for the development of standards. OGC developed its first web service standards in 1997, but at that time most XML standards were still under development and web standards did not exist at all. Microsoft just started to work on SOAP XML-RPC. This explains that most OGC standards specify the use of HTTP/GET and HTTP/POST with KVP (key value pair) encoding, or HTTP/POST with POX (plain old XML).

As a consequence, the OGC web service architecture as it is implemented, is somewhat outdated with regard to today's technologies. Therefore, OGC started the process of identifying how the functional requirements captured in its web service standards should be represented using SOAP/WSDL and REST. In principle KVP, POX, SOAP/WSDL and REST can support the same functional requirements although there are some issues, for instance terms like "operation".

4.3.1 OGC and SOAP/WSDL

In 2006, OGC decided that all new OGC web service standards should specify an optional SOAP/WSDL binding. The main reason for this choice was that SOAP enables the use of general WS-Security and WS-Policy standards. Other arguments were:

- SOAP/WSDL are IT-standards used by communities that also use geo-services;

[13] https://serviceregister.overheid.nl (needs authorization)

- Development tools are available that support SOAP/WSDL by hiding the web service aspects; this is regarded as the least important argument; probably, these tools impose too many constraints to be useful for the GI community;
- SOAP/WSDL services are easy to invoke in e.g. BPEL-scenarios.

The following opposite viewpoints are also widespread:
- the web is RESTful, SOAP/WSDL are a waste of time;
- SOAP and WSDL add complexity, but do not add anything helpful.

OGC follows a general approach in the SOAP/WSDL bindings in OGC web service standards. All follow more or less the same approach with some variations:
- Use of the 'document literal' encoding style to send the same messages used also in POX bindings; not RPC;
- General messages are described in WSDL; these will in general include extension points ("late binding"), often using substitution groups;
- For exceptions, OWS exception messages are embedded in a SOAP fault element.[14]

This approach enables the use of, for example, WS-security standards, but does not align with the use of W3C Web Services in other communities, in particular due to "late binding" and schema complexity.

OGC is developing a generic standard, called OWS (OGC Web Services) Common (OGC 2009), which covers aspects that are shared between other OGC standards. OWS Common 1.2 is under construction; this version will adopt SOAP 1.2, due to the support for MTOM (Message Transmission Optimization Mechanism).

4.3.2 SOAP/WSDL in WMS

WMS 1.1 (OGC 2002) and WMS 1.3 (OGC 2004) both predate OGC's SOAP/WSDL decision, so they don't specify a SOAP binding. There is no WSDL description for these WMS versions. The next WMS version will be based on OWS Common 1.2. For WMS the possibility to use MTOM is important because the response of the WMS-operation getMap consists of (sometimes) large amounts of binary data.

Inspire supports a WSDL/SOAP binding for SOAP 1.1 with MTOM is as a non-standard extension. It is likely that INSPIRE will select WMS 1.3 without any SOAP references for the time being.

4.3.3 SOAP/WSDL in WFS

WFS 1.1 also predates the SOAP/WSDL decision, but still includes a SOAP 1.2 binding. A WSDL description is available informatively. WFS 2.0 is available in draft and has a SOAP 1.1 binding to align with OWS Common 1.1 and WS-I basic profile 1.1. A normative web service description is available. WFS 2.0 deviates from OWS Common 1.2 which uses SOAP 1.2. The most important argument for WMS to select SOAP 1.2, i.e. the support of MTOM, does not apply for WFS.

INSPIRE's position with respect to WFS is unclear. The draft technical guidance references the INSPIRE SOAP framework, but normatively also the SOAP/WSDL sections in the WFS 2.0 draft.

OGC web services traditionally include a getCapabilities operation that provides most service metadata. This overlaps the WSDL description of operations, but also goes beyond what WSDL has

[14] This subject is further discussed in par. 6.9

been created for. For instance: in getCapabilities you will get information about co-ordinate reference systems supported for a feature type in WFS; this information is not part of a WSDL. Additional metadata may be provided in other operations, e.g. describeFeature.

4.3.4 Closing notes

- OGC has and will always support other bindings beside SOAP/WSDL, as long as the rest of the world does it, too.
- Due to the early use of XML and web services in the GI community, SOAP/WSDL are not a natural fit to how web services are used in this community.
- Currently there seems to be a much larger push towards more RESTful “bindings” than for WSDL/SOAP.

5 Test bed results

In order to be able to answer the research questions, two use cases have been specified to run in the test bed. In this chapter these use cases are described in detail with regard to the several stages of message transfer between client and server. The first use case addresses two versions of WMS Servers (WMS 1.1.1 and WMS 1.3.0). In the second use case a WFS Server is involved. The version used is 1.1.0. The original use case also addressed a WFS 2.0.0 server, but this version is not yet available on the market, so this part could not be implemented.

5.1 Scenarios use case 1

Use case 1 contains a number of WMS 1.1.1 and WMS 1.3.0 service calls.

5.1.1 WMS 1.1.1

In the first part of use case 1, a user, by means of an off the shelf WMS client, is addressing a WMS 1.1.1 server that serves risk maps. At first, the capabilities of this server are retrieved. After this, a map of one of the available layers (Risico_installatie) is requested. This scenario is outlined in the following Sequence Diagram.

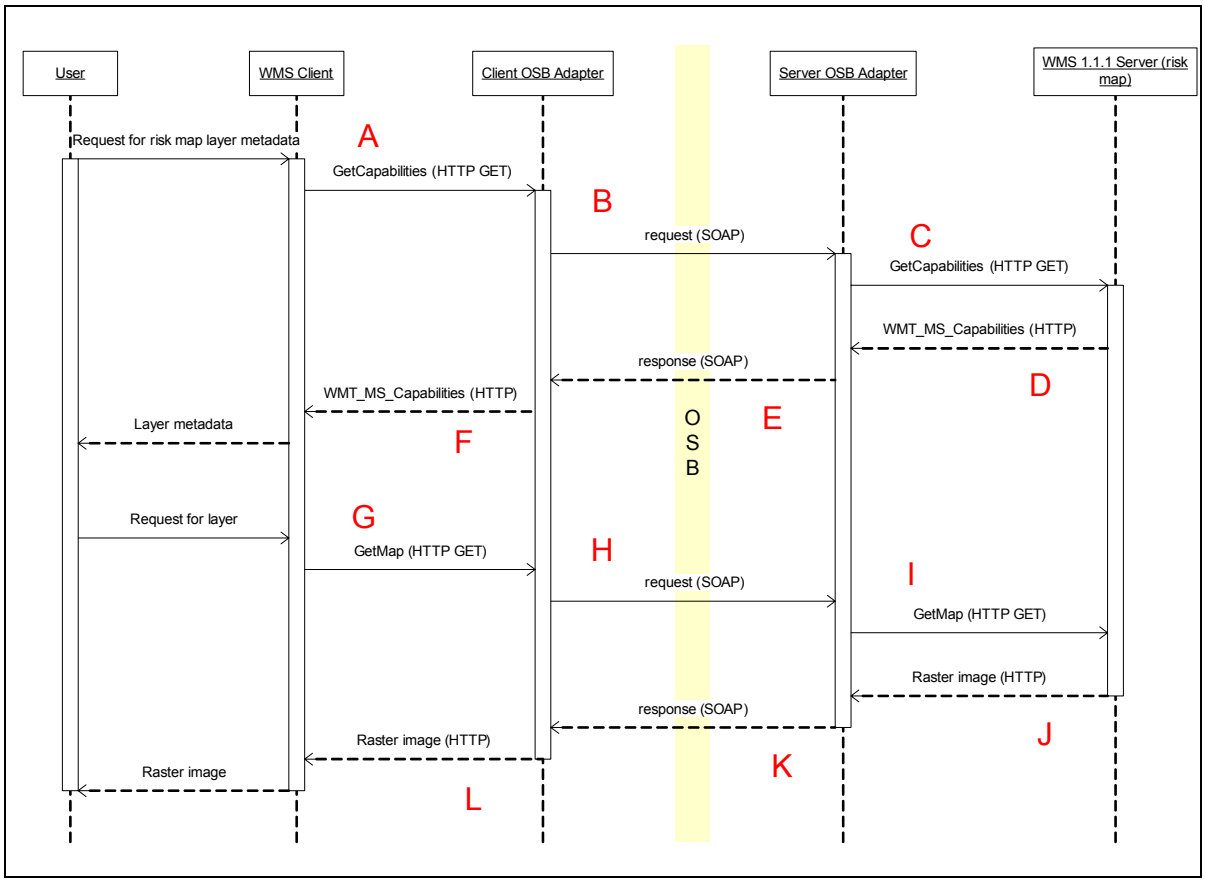

Figure 9 - Sequence diagram - Use case 1, part 1

Below, the information content during the different stages of the message transfer[15] is shown. The applicable WSDL sources are given in Appendix B.

A. GetCapabilities (HTTP/GET)

According to the WMS 1.1.1 standard only a HTTP/GET binding is defined for WMS requests. Listing 2 shows the GetCapabilities request that results from the user's action in the WMS client. The parameters in the URL are in KVP (keyword/value pair) encoding.

```
GET http://212.159.196.130:81/cgi-bin/mapserv.exe? map=D:%5CData%5CRisicokaart.map&amp;version=1.1.1&amp;
                        SERVICE=WMS&amp;REQUEST=GetCapabilities HTTP/1.1
host: 212.159.196.130:81
user-agent: Quantum GIS - 1.4.0-Enceladus
proxy-connection: keep-alive
```

Listing 2

B. request (SOAP)

The HTTP request is received by the Client OSB Adapter, which transforms it into an OSB compliant SOAP message:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:wsa="http://www.w3.org/2005/08/addressing">
    <SOAP-ENV:Header>
        <wsa:To>https://212.159.196.130:8443/tgmServer/RelayService</wsa:To>
        <wsa:ReplyTo>
            <wsa:Address>http://www.w3.org/2005/08/addressing/anonymous
            </wsa:Address>
        </wsa:ReplyTo>
        <wsa:MessageID>http://www.atosorigin.com/1043179418520435
        </wsa:MessageID>
        <wsa:Action>https://www.atosorigin.com/osbtestbed/RelayServiceRequest
        </wsa:Action>
    </SOAP-ENV:Header>
    <SOAP-ENV:Body>
        <atos:request xmlns:atos="http://www.atosorigin.com/wms"
            xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
            xsi:schemaLocation="http://www.atosorigin.com/wms
                                http://212.159.196.130:81/schemas/geonovum-wms.xsd">
            <headers>
                <header>
                    <headerName>host</headerName>
                    <headerValue>212.159.196.130:81</headerValue>
                </header>
                <header>
                    <headerName>user-agent</headerName>
                    <headerValue>Quantum GIS - 1.4.0-Enceladus</headerValue>
                </header>
```

[15] General remarks to the source listings:
1. some white space has been added to improve readability;
2. in reality, the information flow between both adapters is encrypted, according to OSB requirements.

```
                <header>
                    <headerName>proxy-connection</headerName>
                    <headerValue>keep-alive</headerValue>
                </header>
            </headers>
            <method>GET</method>
            <protocol>HTTP/1.1</protocol>
            <queryString>map=D:%5CData%5CRisicokaart.map&amp;version=1.1.1&amp;
                                SERVICE=WMS&amp;REQUEST=GetCapabilities</queryString>
            <requestURL>http://212.159.196.130:81/cgi-bin/mapserv.exe</requestURL>
        </atos:request>
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Listing 3

The Client OSB Adapter parses the URL string in order to separate the server address and the request parameters. Both become part of the Body of the SOAP message, as <requestURL> and <queryString> respectively. Also, all HTTP header information is encoded in the SOAP Body. The OSB standards mandate the usage of the document literal style, which means amongst others that the SOAP Body should contain only one direct child element. Therefore, all defined elements are encoded as children of one higher level element (<request>).

The Client OSB Adapter also defines a SOAP Header block and a SOAP Envelope, all according to the OSB standards.

Detailed notes:

1. The namespace indicated by the SOAP-ENV prefix indicates that the version is SOAP 1.1, as required;
2. The namespace indicated by the wsa prefix (WS Addressing) is also required by the OSB standards;
3. The wsa:To value refers to the Server OSB Adapter;
4. The wsa:Address is prescribed by the OSB standards.

C. GetCapabilities (HTTP/GET)

At server side the original HTTP request (Listing 2) is reconstructed almost completely. The only difference is an extra HTTP header (Connection: Keep-Alive), that is added by Tomcat.

D. WMT_MS_Capabilities (HTTP)

The Server OSB Adapter receives the SOAP message and is able to reconstruct the original HTTP/GET request from the contents of the SOAP Body. This reconstructed request is sent to the WMS Server, which returns a Capabilities document. The frame of this document is shown in Listing 4.

```
HTTP/1.1 200 OK
Date: Fri, 05 Feb 2010 09:06:40 GMT
Server: Apache/2.2.11 (Win32)
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
```

```
Transfer-Encoding: chunked
Content-Type: application/vnd.ogc.wms_xml
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE WMT_MS_Capabilities SYSTEM "http://schemas.opengis.net/wms/1.1.1/WMS_MS_Capabilities.dtd" [
    <!ELEMENT VendorSpecificCapabilities EMPTY>
]>
<!-- end of DOCTYPE declaration -->
<WMT_MS_Capabilities version="1.1.1">
 …
</WMT_MS_Capabilities>
```

Listing 4

E. response (SOAP)

The Server OSB Adapter wraps the received Capabilities document into a SOAP response message, see Listing 5.

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:wsa="http://www.w3.org/2005/08/addressing">
    <SOAP-ENV:Header>
        <wsa:To>http://www.w3.org/2005/08/addressing/anonymous</wsa:To>
        <wsa:RelatesTo>http://www.atosorigin.com/1043179418520435</wsa:RelatesTo>
        <wsa:MessageID>http://www.atosorigin.com/1048738276314832</wsa:MessageID>
        <wsa:From>http://www.w3.org/2005/08/addressing/anonymous</wsa:From>
        <wsa:Action>https://www.atosorigin.com/osbtestbed/RelayServiceResponse</wsa:Action>
    </SOAP-ENV:Header>
    <SOAP-ENV:Body>
        <atos:response xmlns:atos="http://www.atosorigin.com/wms"
            xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
            xsi:schemaLocation="http://www.atosorigin.com/wms
                                http://212.159.196.130:81/schemas/geonovum-wms.xsd">
            <content-length>8378</content-length>
            <content-type>text</content-type>
            <headers>
                <header>
                    <headerName>Date</headerName>
                    <headerValue>Fri, 05 Feb 2010 09:06:40 GMT</headerValue>
                </header>
                <header>
                    <headerName>Server</headerName>
                    <headerValue>Apache/2.2.11 (Win32)</headerValue>
                </header>
                <header>
                    <headerName>Keep-Alive</headerName>
                    <headerValue>timeout=5, max=100</headerValue>
                </header>
                <header>
                    <headerName>Connection</headerName>
                    <headerValue>Keep-Alive</headerValue>
                </header>
                <header>
                    <headerName>Transfer-Encoding</headerName>
                    <headerValue>chunked</headerValue>
```

```
            </header>
            <header>
                <headerName>Content-Type</headerName>
                <headerValue>application/vnd.ogc.wms_xml</headerValue>
            </header>
        </headers>
        <payload><![CDATA[
                <?xml version='1.0' encoding="UTF-8" standalone="no" ?><!DOCTYPE WMT_MS_Capabilities
                SYSTEM "http://schemas.opengis.net/wms/1.1.1/WMS_MS_Capabilities.dtd" [<!ELEMENT
                VendorSpecificCapabilities EMPTY>  ]>  <!-- end of DOCTYPE declaration -->
                <WMT_MS_Capabilities version="1.1.1">...</WMT_MS_Capabilities>
                ]]>
        </payload>
        <payload-type>text</payload-type>
    </atos:response>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Listing 5

The Capabilities document is recorded in the <payload> element in the SOAP Body. The content of this element cannot be described by a HTTP Content-header, so it needs its own customized description. Therefore some additional metadata is included in the <response> element: <content-length>, <content-type> and <payload-type>.

Detailed notes:

1. The <wsa:RelatesTo> contains the MessageID of the Request message that it replies to.
2. The values of <wsa:To>, <wsa:From> and <wsa:Action> are defined according to the OSB Standards.
3. In a 100% solution, the HTTP protocol version and status code should also have been retained.

F. WMT_MS_Capabilities (HTTP)

At client side, the reconstructed response is identical, except for:

1. An added HTTP header (Server: Apache-Coyote/1.1). This results in two HTTP headers with the Server keyword.
2. An overruled HTTP header (Date; a date/time update).

G. GetMap (HTTP/GET)

Based on the received Capabilities document the WMS Client shows a list of available layers. The user chooses one of them to be displayed as a raster image, in this case “Risico_installatie”. This leads to the HTTP Request in Listing 6.

```
GET http://212.159.196.130:81/cgi-bin/mapserv.exe?map=D:%5CData%5CRisicokaart.map&
    SERVICE=WMS&VERSION=1.1.1&REQUEST=GetMap&
    BBOX=197536.269578,499571.670831,210054.030397,507548.343696&SRS=EPSG:28992&
    WIDTH=848&HEIGHT=541&
    LAYERS=Risico_installatie%2CRisico_inrichting%2CKwetsbaarobject%2COngevallen_gevaarlijke_stoffen&
    STYLES=%2C%2C%2C&FORMAT=image/gif&DPI=96&TRANSPARENT=TRUE HTTP/1.1
```

```
host: 212.159.196.130:81
user-agent: Quantum GIS - 1.4.0-Enceladus
proxy-connection: keep-alive
```

Listing 6

The query string contains amongst others the parameter DPI. This is not a standard WMS parameter, but can be regarded as a vendor specific parameter, which is allowed according to the WMS 1.1.1 standard[16].

H. request (SOAP)

The architecture, that is designed for this use case, does not look at the content of the message that has to be transported. So the following listing is, except for the content of the <queryString> element, equal to Listing 3.

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:wsa="http://www.w3.org/2005/08/addressing">
    <SOAP-ENV:Header>
        <wsa:To>https://212.159.196.130:8443/tgmServer/RelayService</wsa:To>
        <wsa:ReplyTo>
            <wsa:Address>http://www.w3.org/2005/08/addressing/anonymous</wsa:Address>
        </wsa:ReplyTo>
        <wsa:MessageID>http://www.atosorigin.com/1043256038926785</wsa:MessageID>
        <wsa:Action>https://www.atosorigin.com/osbtestbed/RelayServiceRequest</wsa:Action>
    </SOAP-ENV:Header>
    <SOAP-ENV:Body>
        <atos:request xmlns:atos="http://www.atosorigin.com/wms"
            xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
            xsi:schemaLocation="http://www.atosorigin.com/wms
                                http://212.159.196.130:81/schemas/geonovum-wms.xsd">
            <headers>
                <header>
                    <headerName>host</headerName>
                    <headerValue>212.159.196.130:81</headerValue>
                </header>
                <header>
                    <headerName>user-agent</headerName>
                    <headerValue>Quantum GIS - 1.4.0-Enceladus</headerValue>
                </header>
                <header>
                    <headerName>proxy-connection</headerName>
                    <headerValue>keep-alive</headerValue>
                </header>
            </headers>
            <method>GET</method>
            <protocol>HTTP/1.1</protocol>
            <queryString>map=D:%5CData%5CRisicokaart.map&SERVICE=WMS&VERSION=1.1.1&
                REQUEST=GetMap& BBOX=197536.269578,499571.670831,210054.030397,507548.343696&
                SRS=EPSG:28992&WIDTH=848&HEIGHT=541&LAYERS=Risico_installatie%2C
                    Risico_inrichting%2CKwetsbaarobject%2COngevallen_gevaarlijke_stoffen&
                STYLES=%2C%2C%2C&FORMAT=image/gif&DPI=96&TRANSPARENT=TRUE
            </queryString>
```

[16] In WMS 1.3.0 vendor specific parameters are not mentioned anymore.

```
            <requestURL>http://212.159.196.130:81/cgi-bin/mapserv.exe</requestURL>
        </atos:request>
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Listing 7

I. GetMap (HTTP/GET)

The same remarks apply as in "*C. GetCapablities (HTTP/GET)*".

J. Raster image (HTTP)

The Server OSB Adapter relays the message to the WMS Server, which returns a raster image. This image is wrapped in the following HTTP response:

```
HTTP/1.1 200 OK
Date: Fri, 05 Feb 2010 09:08:04 GMT
Server: Apache/2.2.11 (Win32)
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Transfer-Encoding: chunked
Content-Type: image/gif
```

Listing 8

K. response (SOAP)

The SOAP message that is composed by the Server OSB Adapter is equal to what is shown in Listing 5, except for the <payload>. The <![CDATA[…]] block effectively contains the base64 encoded raster image.

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:wsa="http://www.w3.org/2005/08/addressing">
    <SOAP-ENV:Header>
        <wsa:To>http://www.w3.org/2005/08/addressing/anonymous</wsa:To>
        <wsa:RelatesTo>http://www.atosorigin.com/1043256038926785</wsa:RelatesTo>
        <wsa:MessageID>http://www.atosorigin.com/1043256527369797</wsa:MessageID>
        <wsa:From>http://www.w3.org/2005/08/addressing/anonymous</wsa:From>
        <wsa:Action>https://www.atosorigin.com/osbtestbed/RelayServiceResponse</wsa:Action>
    </SOAP-ENV:Header>
    <SOAP-ENV:Body>
        <atos:response xmlns:atos="http://www.atosorigin.com/wms"
            xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
            xsi:schemaLocation="http://www.atosorigin.com/wms
                                http://212.159.196.130:81/schemas/geonovum-wms.xsd">
            <content-length>3916</content-length>
            <content-type>image/gif</content-type>
            <headers>
                <header>
                    <headerName>Date</headerName>
```

```
                <headerValue>Fri, 05 Feb 2010 09:08:04 GMT</headerValue>
            </header>
            <header>
                <headerName>Server</headerName>
                <headerValue>Apache/2.2.11 (Win32)</headerValue>
            </header>
            <header>
                <headerName>Keep-Alive</headerName>
                <headerValue>timeout=5, max=100</headerValue>
            </header>
            <header>
                <headerName>Connection</headerName>
                <headerValue>Keep-Alive</headerValue>
            </header>
            <header>
                <headerName>Transfer-Encoding</headerName>
                <headerValue>chunked</headerValue>
            </header>
            <header>
                <headerName>Content-Type</headerName>
                <headerValue>image/gif</headerValue>
            </header>
        </headers>
        <payload><![CDATA[...]]></payload>
        <payload-type>payload-type-binary</payload-type>
    </atos:response>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Listing 9

L. Raster image (HTTP)

The same remarks apply as in "*F. WMT_MS_Capabilities (HTTP)*".

5.1.2 WMS 1.3.0

In the second part of use case 1, a WMS 1.3.0 server is addressed. This server serves topographical maps. At first, the capabilities of this server are retrieved. After that, a map of one of the available layers (Wegdeel Polygon) is requested. This scenario is outlined in the following Sequence Diagram.

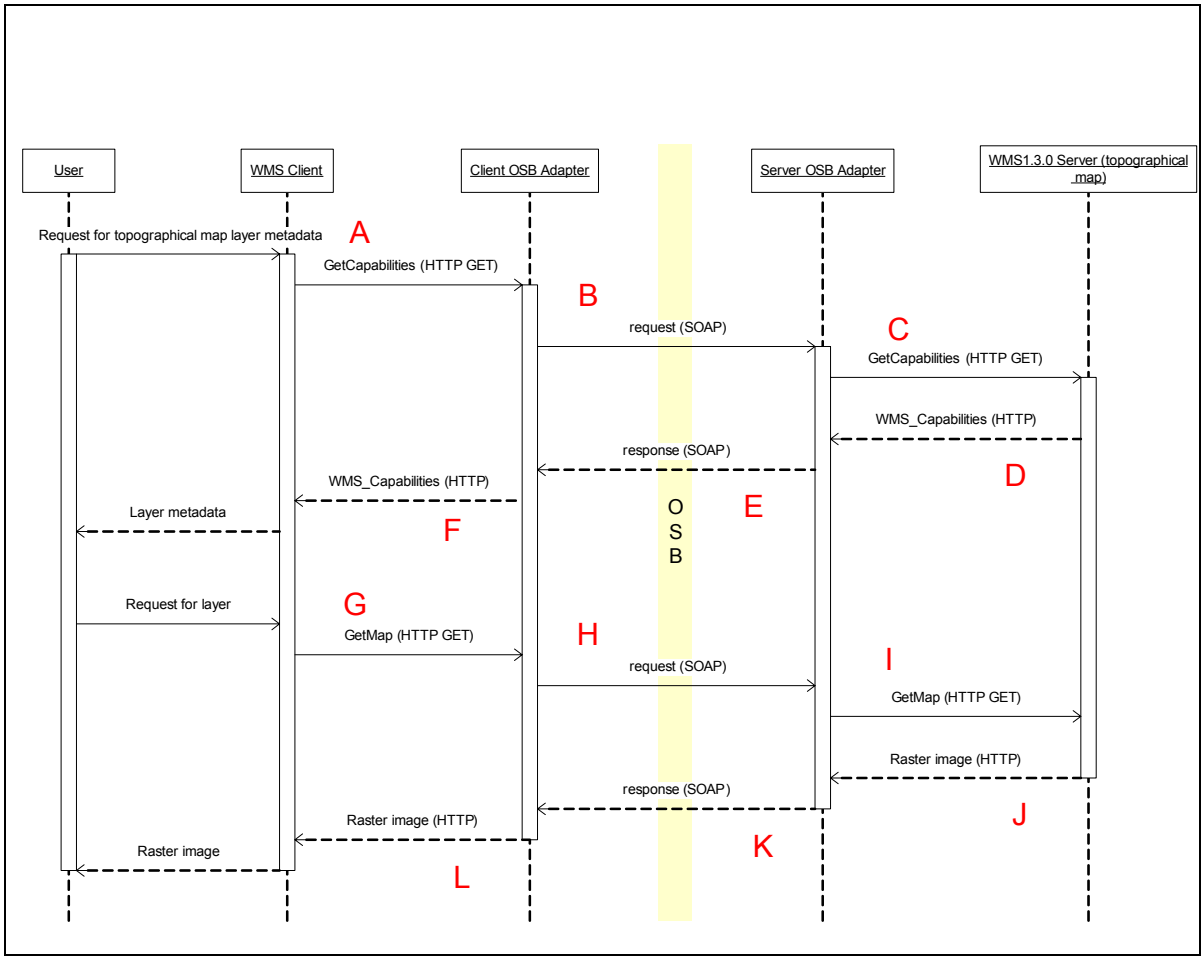

Figure 10 - Sequence diagram - Use case 1, part 2

Because of the INSPIRE requirements we also included the mandatory LANGUAGE parameter in the GetCapabilities and GetMap requests in the scenario[17]. This parameter is not defined by the WMS 1.3.0 standard and ignored by the WMS Server. Since the Capabilities document should advertise the LANGUAGES supported, this functionality had to be simulated. This is, however, not shown in the diagram. In the Capabilities document in Listing 12, the simulated response is included.

Because of the large similarity between the WMS 1.1.1 and WMS 1.3.0 listings, only the GetCapabilities listings for WMS 1.3.0 are given, without repeating comments already given in the WMS 1.1.1 scenario. Again HTTP/GET is used as the transport protocol because this is the only way current WMS clients work.

[17] Note that (OGC 2009) specifies a different way to handle multi-linguism.

A. GetCapabilities (HTTP/GET)

```
GET http://212.159.196.130:81/cgi-bin/mapserv.exe?map=D:%5CData%5CTOP10NL.map&version=1.3.0&
    SERVICE=WMS&REQUEST=GetCapabilities&LANGUAGE=dut HTTP/1.1
host: 212.159.196.130:81
user-agent: Quantum GIS - 1.4.0-Enceladus
proxy-connection: keep-alive
```

Listing 10

B. request (SOAP)

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:wsa="http://www.w3.org/2005/08/addressing">
    <SOAP-ENV:Header>
        <wsa:To>https://212.159.196.130:8443/tgmServer/RelayService</wsa:To>
        <wsa:ReplyTo>
            <wsa:Address>http://www.w3.org/2005/08/addressing/anonymous</wsa:Address>
        </wsa:ReplyTo>
        <wsa:MessageID>http://www.atosorigin.com/1043473418449633</wsa:MessageID>
        <wsa:Action>https://www.atosorigin.com/osbtestbed/RelayServiceRequest</wsa:Action>
    </SOAP-ENV:Header>
    <SOAP-ENV:Body>
        <atos:request xmlns:atos="http://www.atosorigin.com/wms"
            xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
            xsi:schemaLocation="http://www.atosorigin.com/wms
                                http://212.159.196.130:81/schemas/geonovum-wms.xsd">
            <headers>
                <header>
                    <headerName>host</headerName>
                    <headerValue>212.159.196.130:81</headerValue>
                </header>
                <header>
                    <headerName>user-agent</headerName>
                    <headerValue>Quantum GIS - 1.4.0-Enceladus</headerValue>
                </header>
                <header>
                    <headerName>proxy-connection</headerName>
                    <headerValue>keep-alive</headerValue>
                </header>
            </headers>
            <method>GET</method>
            <protocol>HTTP/1.1</protocol>
            <queryString>map=D:%5CData%5CTOP10NL.map&amp;version=1.3.0&amp;
                    SERVICE=WMS&amp;REQUEST=GetCapabilities&amp;LANGUAGE=dut
            </queryString>
            <requestURL>http://212.159.196.130:81/cgi-bin/mapserv.exe</requestURL>
        </atos:request>
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Listing 11

D. WMS_Capabilities (HTTP)

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<WMS_Capabilities version="1.3.0" xmlns="http://www.opengis.net/wms"
    xmlns:sld="http://www.opengis.net/sld" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:ms="http://mapserver.gis.umn.edu/mapserver"
    xsi:schemaLocation="
        http://www.opengis.net/wms http://schemas.opengis.net/wms/1.3.0/capabilities_1_3_0.xsd
        http://www.opengis.net/sld http://schemas.opengis.net/sld/1.1.0/sld_capabilities.xsd
        http://mapserver.gis.umn.edu/mapserver
            http://212.159.196.130:81/cgi-bin/mapserv.exe?map=D:\Data\TOP10NL.map&amp;
                service=WMS&amp;version=1.3.0&amp;request=GetSchemaExtension">
  …
  <Capability>
      …
      <INSPIRE:ViewCapabilities xmlns:INSPIRE="http://www.inspire.org"
          xsi:schemaLocation="http://www.inspire.org http://212.159.196.130:81/schemas/inspireview.xsd">
          <INSPIRE:Languages>
              <INSPIRE:Language default="true">eng</INSPIRE:Language>
              <INSPIRE:Language>dut</INSPIRE:Language>
          </INSPIRE:Languages>
          <INSPIRE:CurrentLanguage>dut</INSPIRE:CurrentLanguage>
      </INSPIRE:ViewCapabilities>
    …
  </Capability>
  …
</WMS_Capabilities>
```

Listing 12

E. response (SOAP)

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:wsa="http://www.w3.org/2005/08/addressing">
    <SOAP-ENV:Header>
        <wsa:To>http://www.w3.org/2005/08/addressing/anonymous</wsa:To>
        <wsa:RelatesTo>http://www.atosorigin.com/1043473418449633</wsa:RelatesTo>
        <wsa:MessageID>http://www.atosorigin.com/1043937927381734</wsa:MessageID>
        <wsa:From>http://www.w3.org/2005/08/addressing/anonymous</wsa:From>
        <wsa:Action>https://www.atosorigin.com/osbtestbed/RelayServiceResponse
        </wsa:Action>
    </SOAP-ENV:Header>
    <SOAP-ENV:Body>
        <atos:response xmlns:atos="http://www.atosorigin.com/wms"
            xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
            xsi:schemaLocation="http://www.atosorigin.com/wms
                                http://212.159.196.130:81/schemas/geonovum-wms.xsd">
            <content-length>10792</content-length>
            <content-type>text</content-type>
            <headers>
                <header>
                    <headerName>Date</headerName>
```

```
                    <headerValue>Fri, 05 Feb 2010 09:12:01 GMT</headerValue>
                </header>
                <header>
                    <headerName>Server</headerName>
                    <headerValue>Apache/2.2.11 (Win32)</headerValue>
                </header>
                <header>
                    <headerName>Keep-Alive</headerName>
                    <headerValue>timeout=5, max=100</headerValue>
                </header>
                <header>
                    <headerName>Connection</headerName>
                    <headerValue>Keep-Alive</headerValue>
                </header>
                <header>
                    <headerName>Transfer-Encoding</headerName>
                    <headerValue>chunked</headerValue>
                </header>
                <header>
                    <headerName>Content-Type</headerName>
                    <headerValue>text/xml</headerValue>
                </header>
            </headers>
            <payload><![CDATA[
                <?xml version='1.0' encoding="UTF-8" standalone="no" ?><WMS_Capabilities version="1.3.0"
                xmlns="http://www.opengis.net/wms"   xmlns:sld="http://www.opengis.net/sld"
                xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                xmlns:ms="http://mapserver.gis.umn.edu/mapserver" xsi:schemaLocation="http://www.opengis.net/wms
                http://schemas.opengis.net/wms/1.3.0/capabilities_1_3_0.xsd  http://www.opengis.net/sld
                http://schemas.opengis.net/sld/1.1.0/sld_capabilities.xsd  http://mapserver.gis.umn.edu/mapserver
                http://212.159.196.130:81/cgi-bin/mapserv.exe?map=D:\Data\TOP10NL.map&amp;
                service=WMS&amp;version=1.3.0&amp;request=GetSchemaExtension">…</WMS_Capabilities>
                    ]]>
            </payload>
            <payload-type>text</payload-type>
        </atos:response>
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Listing 13

5.2 Scenarios use case 2

Use case 2 contains a number of WFS 1.1.0 service calls. No WFS 2.0.0 server is available on the market yet, so no test could be carried out for this version. So, further analysis on this version is based on the specifications only.

5.2.1 WFS 1.1.0

For the second use case, a simple browser front end has been developed, which offers some form fields and related action buttons (the “LPG risk system” in Figure 11). When an action button is activated, the contents of the related form field are posted (HTTP POST) to a WFS 1.1.0 server according to the OSB standards. It is also possible not to use the OSB-wrapper and call the WFS Server directly. The form fields are prefilled with GetCapabilities and GetFeature requests, which according to the scenario have to be executed consecutively. The GetFeature request retrieves all LPG installations by using a suitable filter (value “RESERVOIR” for attribute “SOORT_I3”).

The scenario of this use case is shown in the following Sequence Diagram. It is based on the functional architecture that is sketched in 4.1.

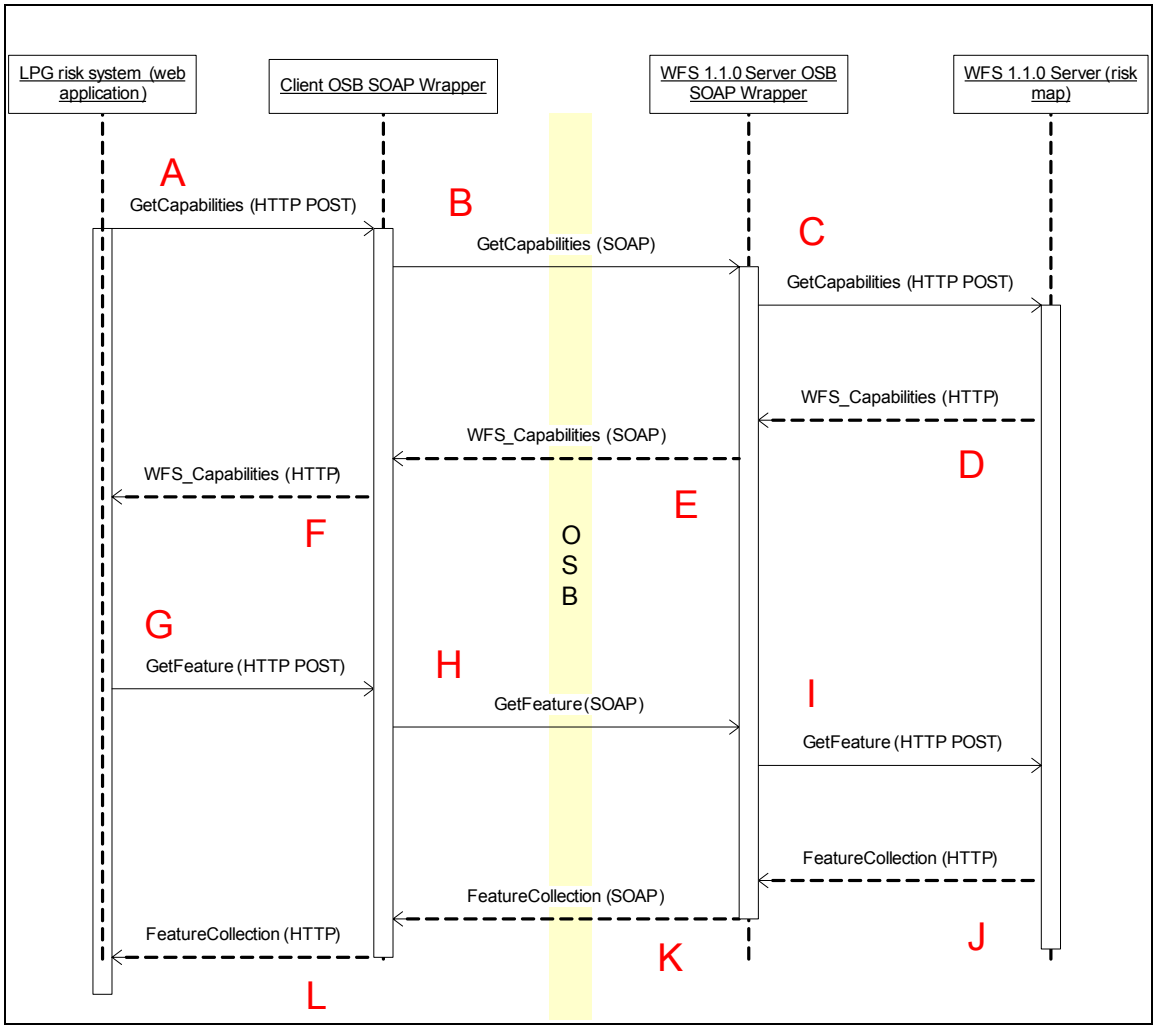

Figure 11 - Sequence diagram - Use case 2

Below, the information content is shown during the different stages of the message transfer. The applicable WSDL sources are given in Appendix C. In this use case, HTTP header information is not manipulated during transfer, and therefore not shown.

A/C. GetCapabilities (HTTP POST)

The GetCapabilities element is specified in a publicly available schema definition (wfs.xsd), so can be validated.

```
<?xml version="1.0" encoding="UTF-8"?>
<wfs:GetCapabilities service="WFS" xmlns:wfs="http://www.opengis.net/wfs"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://www.opengis.net/wfs http://schemas.opengis.net/wfs/1.1.0/wfs.xsd" />
```

Listing 14

B. GetCapabilities (SOAP)

The GetCapabilities element is included in the SOAP Body as the sole element. So, the SOAP Body only contains a functional service. This is one of the main differences compared to the architecture used in use case 1. The other large difference is the contents of the <wsa:To> element. This element contains the address of the WFS Service (wrapped in its OSB SOAP Wrapper). The <wsa:Action> element has a value that is specific for GetCapabilities, while in use case 1, this element contained a generic value.

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:wsa="http://www.w3.org/2005/08/addressing">
    <SOAP-ENV:Header>
        <wsa:To>http://localhost:8084/tgmWebservice/WFSService</wsa:To>
        <wsa:ReplyTo>
            <wsa:Address>http://www.w3.org/2005/08/addressing/anonymous
            </wsa:Address>
        </wsa:ReplyTo>
        <wsa:MessageID>http://www.atosorigin.com/4411538627674</wsa:MessageID>
        <wsa:Action>https://www.atosorigin.com/osbtestbed/wfsgetCapabilitiesRequest
        </wsa:Action>
    </SOAP-ENV:Header>
    <SOAP-ENV:Body>
        <wfs:GetCapabilities service="WFS" xmlns:wfs="http://www.opengis.net/wfs"
            xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
            xsi:schemaLocation="http://www.opengis.net/wfs http://schemas.opengis.net/wfs/1.1.0/wfs.xsd" />
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Listing 15

D/F. WFS_Capabilities (HTTP)

The resulting WFS_Capabilities document looks as follows:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<wfs:WFS_Capabilities xmlns="http://www.opengis.net/wfs"
    xmlns:geonovum="http://www.atosorigin.com/geonovum"
    xmlns:gml="http://www.opengis.net/gml" xmlns:ogc="http://www.opengis.net/ogc"
    xmlns:ows="http://www.opengis.net/ows" xmlns:topp="http://www.openplans.org/topp"
    xmlns:wfs="http://www.opengis.net/wfs" xmlns:xlink="http://www.w3.org/1999/xlink"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    updateSequence="57" version="1.1.0"
    xsi:schemaLocation="http://www.opengis.net/wfs
                        http://nlhxgp92j:8080/geoserver/schemas/wfs/1.1.0/wfs.xsd">
    …
</wfs:WFS_Capabilities>
```

Listing 16

E. WFS_Capabilities (SOAP)

The WFS_Capabilities document becomes the SOAP Body in the response SOAP message:

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:wsa="http://www.w3.org/2005/08/addressing">
    <SOAP-ENV:Header>
        <wsa:To>http://www.w3.org/2005/08/addressing/anonymous</wsa:To>
        <wsa:RelatesTo>http://www.atosorigin.com/4411538627674</wsa:RelatesTo>
        <wsa:MessageID>http://www.atosorigin.com/4413795839481</wsa:MessageID>
        <wsa:From>http://www.w3.org/2005/08/addressing/anonymous</wsa:From>
        <wsa:Action>https://www.atosorigin.com/osbtestbed/wfsgetCapabilitiesResponse
        </wsa:Action>
    </SOAP-ENV:Header>
    <SOAP-ENV:Body>
        <wfs:WFS_Capabilities xmlns="http://www.opengis.net/wfs"
            xmlns:geonovum="http://www.atosorigin.com/geonovum"
            xmlns:gml="http://www.opengis.net/gml" xmlns:ogc="http://www.opengis.net/ogc"
            xmlns:ows="http://www.opengis.net/ows" xmlns:topp="http://www.openplans.org/topp"
            xmlns:wfs="http://www.opengis.net/wfs" xmlns:xlink="http://www.w3.org/1999/xlink"
            xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
            updateSequence="57" version="1.1.0"
            xsi:schemaLocation="http://www.opengis.net/wfs
                                http://nlhxgp92j:8080/geoserver/schemas/wfs/1.1.0/wfs.xsd">
        …
        </wfs:WFS_Capabilities>
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Listing 17

G/I. GetFeature (HTTP POST)

To retrieve all LPG installations, the following GetFeature request is posted:

```
<?xml version="1.0" encoding="UTF-8"?>
<wfs:GetFeature service="WFS" version="1.1.0"
        xmlns:wfs="http://www.opengis.net/wfs"
        xmlns:ogc="http://www.opengis.net/ogc"
        xmlns:geonovum="http://www.atosorigin.com/geonovum"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://www.opengis.net/wfs http://schemas.opengis.net/wfs/1.1.0/wfs.xsd">
    <wfs:Query typeName="geonovum:RisicoInstallatie">
        <ogc:Filter>
            <ogc:PropertyIsEqualTo>
                <ogc:PropertyName>geonovum:SOORT_I3</ogc:PropertyName>
                <ogc:Literal>RESERVOIR</ogc:Literal>
            </ogc:PropertyIsEqualTo>
        </ogc:Filter>
    </wfs:Query>
</wfs:GetFeature>
```

Listing 18

H. GetFeature (SOAP)

The request is further processed in the same way as the GetCapabilities request.

```
<SOAP-ENV:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:wsa="http://www.w3.org/2005/08/addressing">
    <SOAP-ENV:Header>
        <wsa:To>http://localhost:8084/tgmWebservice/WFSService</wsa:To>
        <wsa:ReplyTo>
            <wsa:Address>http://www.w3.org/2005/08/addressing/anonymous
            </wsa:Address>
        </wsa:ReplyTo>
        <wsa:MessageID>http://www.atosorigin.com/4477562635474</wsa:MessageID>
        <wsa:Action>https://www.atosorigin.com/osbtestbed/wfsgetFeatureRequest
        </wsa:Action>
    </SOAP-ENV:Header>
    <SOAP-ENV:Body>
        <wfs:GetFeature service="WFS" version="1.1.0"
            xmlns:wfs="http://www.opengis.net/wfs" xmlns:ogc="http://www.opengis.net/ogc"
            xmlns:geonovum="www.atosorigin.com/geonovum"
            xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
            xsi:schemaLocation="http://www.opengis.net/wfs http://schemas.opengis.net/wfs/1.1.0/wfs.xsd">
            <wfs:Query typeName="geonovum:RisicoInstallatie">
                <ogc:Filter>
                    <ogc:PropertyIsEqualTo>
                        <ogc:PropertyName>geonovum:SOORT_I3</ogc:PropertyName>
                        <ogc:Literal>RESERVOIR</ogc:Literal>
                    </ogc:PropertyIsEqualTo>
                </ogc:Filter>
            </wfs:Query>
        </wfs:GetFeature>
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Listing 19

J/L. FeatureCollection (HTTP)

The normal response to a GetFeature request is a FeatureCollection document:

```
<wfs:FeatureCollection xmlns:geonovum="http://www.atosorigin.com/geonovum"
        xmlns:gml="http://www.opengis.net/gml" xmlns:ogc="http://www.opengis.net/ogc"
        xmlns:ows="http://www.opengis.net/ows" xmlns:wfs="http://www.opengis.net/wfs"
        xmlns:xlink="http://www.w3.org/1999/xlink" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        numberOfFeatures="20" timeStamp="2010-02-02T10:25:16.636+01:00"
        xsi:schemaLocation="http://www.atosorigin.com/geonovum
            http://212.159.196.130:8080/geoserver/wfs?service=WFS&amp;version=1.1.0&amp;
                request=DescribeFeatureType&amp;typeName=geonovum%3ARisicoInstallatie
            http://www.opengis.net/wfs http://212.159.196.130:8080/geoserver/schemas/wfs/1.1.0/wfs.xsd">
    …
</wfs:FeatureCollection>
```

Listing 20

K. FeatureCollection (SOAP)

The FeatureCollection document is processed in the same way as the WFS_Capabilities document:

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:wsa="http://www.w3.org/2005/08/addressing">
    <SOAP-ENV:Header>
        <wsa:To>http://www.w3.org/2005/08/addressing/anonymous</wsa:To>
        <wsa:RelatesTo>http://www.atosorigin.com/4477562635474</wsa:RelatesTo>
        <wsa:MessageID>http://www.atosorigin.com/4477729387201</wsa:MessageID>
        <wsa:From>http://www.w3.org/2005/08/addressing/anonymous</wsa:From>
        <wsa:Action>https://www.atosorigin.com/osbtestbed/wfsgetFeatureResponse
        </wsa:Action>
    </SOAP-ENV:Header>
    <SOAP-ENV:Body>
        <wfs:FeatureCollection xmlns:geonovum="http://www.atosorigin.com/geonovum"
            xmlns:gml="http://www.opengis.net/gml" xmlns:ogc="http://www.opengis.net/ogc"
            xmlns:ows="http://www.opengis.net/ows"
            xmlns:wfs="http://www.opengis.net/wfs" xmlns:xlink="http://www.w3.org/1999/xlink"
            xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
            numberOfFeatures="20" timeStamp="2010-02-02T10:25:16.636+01:00"
            xsi:schemaLocation="http://www.atosorigin.com/geonovum
                    http://212.159.196.130:8080/geoserver/wfs?service=WFS&amp;version=1.1.0&amp;
                        request=DescribeFeatureType&amp;typeName=geonovum%3ARisicoInstallatie
                    http://www.opengis.net/wfs http://212.159.196.130:8080/geoserver/schemas/wfs/1.1.0/wfs.xsd">
                …
        </wfs:FeatureCollection>
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Listing 21

5.2.2 WFS 2.0.0

No WFS 2.0.0 server is available on the market yet, so no test could be carried out. So, further analysis on this version is based on the specifications only.

5.3 OSB performance overhead

Figure 12 shows the average response times of 4 x 500 service requests according to the architecture of use case 1. The 4 requests were:

- a getMap request, resulting in a 2 kB jpg image, directly to MapServer
- the same request, now using the OSB adapters
- a getCapabilities request, resulting in a 10 kB capabilities document, directly to MapServer
- the same request, now using the OSB adapters.

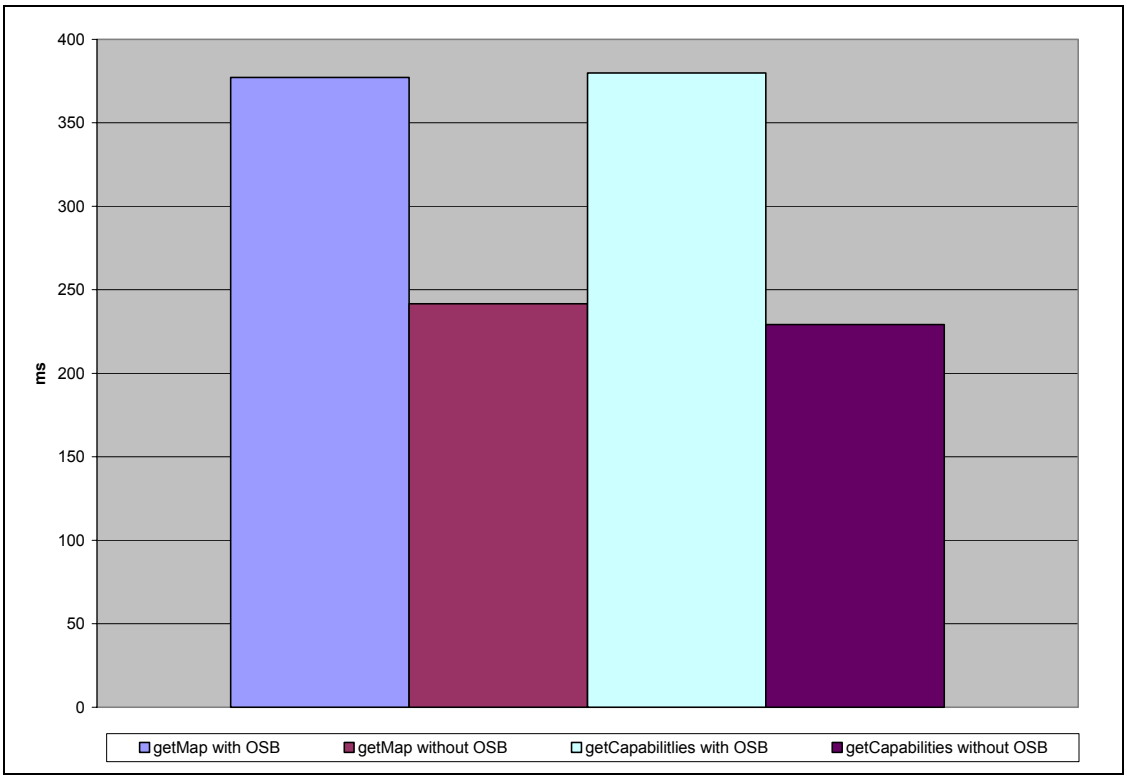

Figure 12

It can be concluded that under these circumstances (small sized messages) the 'OSB overhead' is about 150 ms, which is in this case about 50%. Base64 encoding and decoding of the image does not have a significant influence on the response time.

In order to estimate the impact of base64 encoding and decoding, a second comparative test has been carried out. Under the same conditions also a getMap request has been issued, now resulting in an image size of 50 kB.

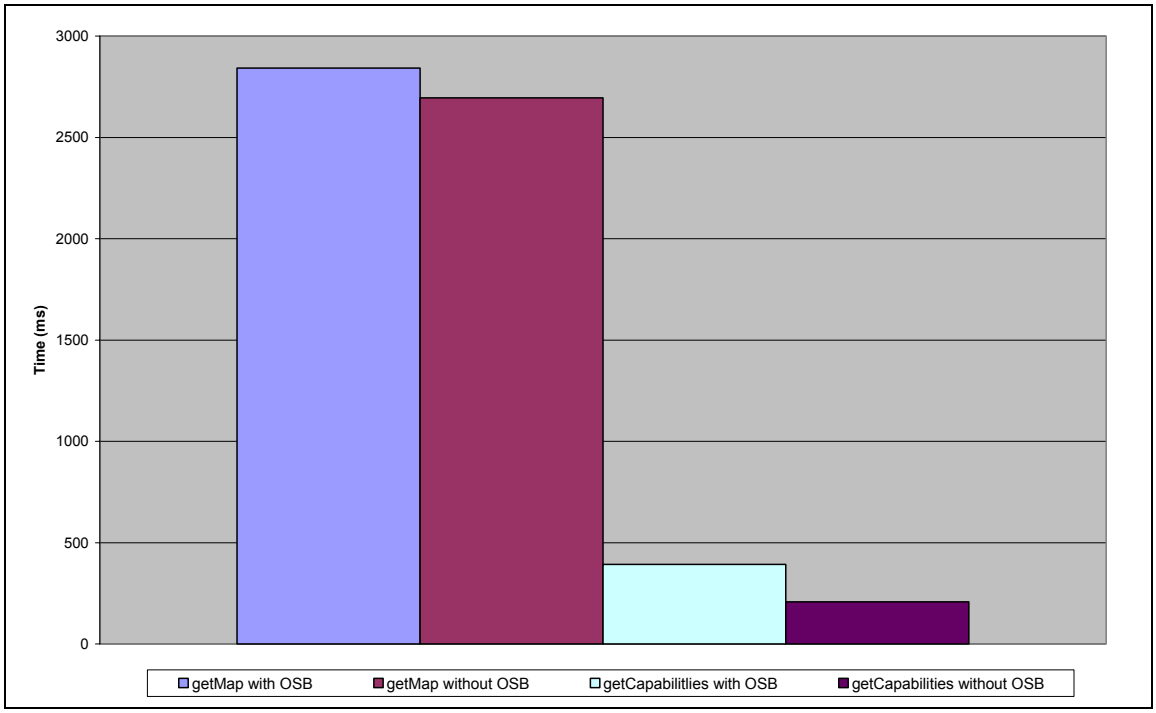

Figure 13

For this average sized image size the relative difference in response times has become very small. The absolute difference has hardly increased and is still 150ms. So, under these conditions base64 encoding does not contribute significantly to the response time. The same result was achieved with an 270 kB image map.

An alternative solution to transfer binary content, such as raster images, is using SOAP with attachments, applying MTOM / XOP. The main advantage of this approach is not as much that base64 encoding is prevented, but that the SOAP message can be transferred in parts, instead of one complete message. In that way, time-outs become less probable.

6 Research questions

The research questions are answered as if both use cases were implemented according to the functional architecture. In Appendix D, the WMS 1.3.0 scenario from use case 1 is described as if implemented in this way.

6.1 Use case 1 - research question 1

> Is it possible to invoke WMS operations according to the OGC standards WMS 1.1.1 and OWS Common 1.2 and OSB 1.1?

The following table shows that OGC and OSB refer to different versions of a number of standards / profiles. The main difference is the way binary files, important in WMS context (raster images), are transmitted. This can be derived from the difference in WS-I BP versions. Only WS-I BP version 1.2 supports the use of MTOM binary attachments. For comparison, a column for OSB version 2.0 (OSB 2010-b) is also included. It shows that the need for MTOM is already recognized in OSB context.

	OGC WMS 1.1.1	OGC OWS Common 1.2	OSB 1.1	OSB 2.0
SOAP	-	1.2	1.1	1.1
WS-I BP (WSDL, XML)	-	1.2	1.1	1.1
WS-I BP (Addressing)	-	-	1.2	1.2
WS-I BP (encoding binary data)	-	1.2	1.1	1.2
Encoding	-	document literal	document literal	document literal

Table 1

The test bed proved that the more detailed OSB requirements with respect to WS-Addressing and security can be met. An OSB compliance test has not been carried out.

On a functional level, an important issue is the lack of a standardized XML Schema that can be used to encode the GetMap and GetCapabilities requests and the resulting image from a GetMap request. Proprietary choices have to be made. Only for the WMT_MS_Capabilities document a standard XML Schema is available.

6.2 Use case 1 - research question 2

> Is it possible to invoke WMS operations according to the INSPIRE guidelines and OSB 1.1?

The same conclusion (difference in the way binary files are to be transmitted) can be drawn as in the former case.

	INSPIRE on View Services (OGC WMS 1.3.0)	OSB 1.1	OSB 2.0
SOAP	1.1	1.1	1.1
WS-I BP (WSDL, XML)	1.2	1.1	1.1
WS-I BP (Addressing)	1.2 (informative)	1.2	1.2
WS-I BP (encoding binary data)	1.2	1.1	1.2
Encoding	document literal	document literal	document literal

Table 2

The test bed proved that the more detailed OSB requirements with respect to WS-Addressing and security can be met. An OSB compliance test has not been carried out.

On a functional level, an important issue is the lack of a standardized XML Schema that can be used to encode the GetMap and GetCapabilities requests and the resulting image from a GetMap request. Proprietary choices have to be made. Only for the WMS_Capabilities document a standard XML Schema is available. In Appendix D an example is given.

6.3 Use case 1 - research question 3

What are the differences re WSDL/SOAP according to OGC, INSPIRE and OSB in the context of WMS?

See Table 1 and Table 2.

6.4 Use case 1 - research question 4

Why would you use WMS based on WSDL/SOAP for retrieving Geo-data?

Availability of Web services through SOAP/WSDL is useful when these services have to play a role in a larger ‘orchestration’ context, e.g. a business process. SOAP/WSDL based web services can easily be invoked from an ESB or BPEL context, where they become part of a service chain. In this approach, the process is leading. One of the possible services in such a chain can be a ‘human service’, which through a user interface prompts a human actor to supply the information needed for the continuation of the process. The process, including human tasks, is defined at design time.

WMS operations normally don't play a role in such a predefined process, but they are typically used in an 'ad hoc' way. A WMS session normally consists of a number of user activities in any order:

1. retrieving one or more layers from a number of WMS servers (GetCapabilities and GetMap);
2. retrieving the details of some objects by clicking on the map, thus defining the x,y coordinates as the selection criterion (GetFeatureInfo);
3. zooming in or out, hiding or showing layers, modifying layer order
4. visually analyzing the result.

This is clearly not a system defined flow. At any moment the user decides on an ad hoc basis what to do next.

The invocation of a WMS service in the context of an orchestration could be envisaged for providing the background raster image behind the results of e.g. WFS service invocations. We could argue if you should design this in an orchestration (so: in the business layer) or leave it as a responsibility for the presentation layer.

Outside the scope of such scenarios it is hard to find other sensible ways of using WMS operations. So, on the functional level, there is few added value in using WMS with WSDL/SOAP. If, however, a SOAP/WSDL platform is available, which also provides non-functional features like secure connections (e.g. OSB), this could be a decisive argument in favor of WSDL/SOAP.

6.5 Use case 2 - research question 1

Is it possible to invoke WFS operations according to ISO/TC 211 19142 and OSB 1.1?

This part of the use case has not been implemented in the test bed, because no WFS 2.0.0 servers are yet available on the market. Nevertheless, the question can be answered affirmatively, as the following table illustrates.

	ISO/TC 211 WFS 2.0.0	OSB 1.1	OSB 2.0
SOAP	1.1	1.1	1.1
WS-I BP (WSDL, XML)	1.1	1.1	1.1
WS-I BP (Addressing)	-	1.2	1.2
WS-I BP (encoding binary data)	-	1.1	1.2
Encoding	document literal	document literal	document literal

Table 3

On a functional level, standard XML Schemas are available for all WFS operations, e.g. GetCapabilities and GetFeature.

6.6 Use case 2 - research question 2

Is it possible to invoke WFS operations according to the OGC standards WFS 1.1.0 and OWS Common 1.2 and OSB 1.1?

The following table shows that the required SOAP versions are again different. However, since WFS operations don't include binary files, there is no real need for SOAP version 1.2 and MTOM or XOP. Indeed, the later WFS 2.0.0 standard only mentions version SOAP version 1.1.

	OGC WFS 1.1.0	OGC OWS Common 1.2	OSB 1.1	OSB 2.0
SOAP	1.2	1.2	1.1	1.1
WS-I BP (WSDL, XML)	1.1	1.2	1.1	1.1
WS-I BP (Addressing)	-	-	1.2	1.2
WS-I BP (encoding binary data)	-	1.2	1.1	1.2
Encoding	document literal	document literal	document literal	document literal

Table 4

The test bed proved that the more detailed OSB requirements with respect to WS-Addressing and security can be met. The services passed the OSB Compliance Test. See Appendix F for the detailed results.

For all WFS operations standard XML Schemas are available. The schemas for GetCapabilities and GetFeature are used in the test bed.

6.7 Use case 2 - research question 3

What are the differences re WSDL/SOAP according to OGC, ISO/TC 211 19142 and OSB in the context of WFS?

See Table 3 and Table 4.

6.8 Summary of differences in standards

The following table is a summary of the tables in the previous paragraphs. For sake of completeness, also a few functional differences are added.

	OGC WMS 1.1.1	OGC OWS Common 1.2	Inspire on View Services (OGC WMS 1.3.0)	OGC WFS 1.1.0	ISO/TC 211 WFS 2.0.0	OSB 1.1	OSB 2.0
Differences on SOAP / WSDL							
SOAP version	-	1.2	1.1	1.2	1.1	1.1	1.1
WS-I Basic Profile version re SOAP, WSDL, XML	-	1.2	1.2	-	1.1	1.1	1.1
WS-I Basic Profile version re Addressing	-	-	1.2 (informative)	-	-	1.2	1.2
WS-I BP (encoding binary data)	-	1.2	1.2	-	-	1.1	1.2
Encoding	-	document literal wrapped	document literal	doc literal	document literal	doc literal	doc literal
Functional differences							
Vendor specific WMS getMap parameters	Y	-	N	-	-	-	-
LANGUAGE parameter in getMap	N	-	Y (N)	-	-	-	-

Table 5 - Overview of differences in standards

6.9 Additional findings

The test bed scenarios did not include exception handling. Still, during analysis some discussion was held on the way OWS exceptions should be reported according to (OGC 2009), when using SOAP encoding:

> "If an error is detected while processing an operation request encoded in a SOAP envelope, the OWS server shall generate a SOAP response message where the content of the Body element is a Fault element containing an ExceptionReport element."

and

> "The code element shall have the Value "soap:server" indicating that this is a server exception. The Reason element shall have the Text "Server exception was encountered." This fixed string is used since the details of the exception shall be specified in the Detail element using an ows:ExceptionReport element."

Probably it is not according to the SOAP specification to report this type of exceptions through a "soap:server" SOAP Fault. The explanation of these type of faults in the SOAP 1.1 standard is:

> "The Server class of errors indicate that the message could not be processed for reasons not directly attributable to the contents of the message itself but rather to the processing of the message. For example, processing could include communicating with an upstream processor, which didn't respond. The message may succeed at a later point in time. See also section 4.4 for a description of the SOAP Fault detail sub-element.[18]"

However, OWS exceptions are raised because of incorrect contents of the message. Such a message will not succeed at a later time. Instead, SOAP fault code "soap:client" appears to be more appropriate:

> "The Client class of errors indicate that the message was incorrectly formed or did not contain the appropriate information in order to succeed. […]"

This is also the way it is interpreted in the WFS 2.0 standard (ISO/TC 211 2009-b):

> "The value in the <soap:faultcode> element shall have the content soap:Client or soap:Server indicating that this is a service exception. The value soap:Client shall be used if the message was incorrectly formed or did not contain the appropriate information in order to succeed (e.g. contained an invalid WFS request)."

[18] http://www.w3.org/TR/2000/NOTE-SOAP-20000508/#_Toc478383507

7 Conclusions and recommendations

This report presents the results of the project 'Test bed geo-services Geonovum'. Supported by an actual implementation, it was investigated to which extent it is possible to combine application level geo-standards WMS and WFS with the infrastructural communication standards based on SOAP/WSDL, as adopted by the Dutch government in its OSB WUS standard. Several versions of these standards and its global or European profiles were within scope. Also, the question was dealt with if it is useful to strive to interoperability of these standards at all.

The conclusions for WMS and WFS are different and will be given separately.

7.1 WMS

Especially for WMS, the question is justifiable if SOAP/WSDL transport is to be considered useful or not. In paragraph 6.4 it was concluded that at least on a functional level the answer is negative. In general, the availability of Web services through SOAP/WSDL is useful when one would like these services to play a role in a larger 'orchestration' context, e.g. a business process. SOAP/WSDL based web services can easily be invoked from an ESB or BPEL context, where they become part of a service chain. In this approach, the process is leading. One of the possible services in such a chain can be a 'human service', which through a user interface prompts a human actor to supply the information needed for the continuation of the process. Such a process, including human tasks, is defined at design time. However, WMS operations normally don't play a role in such a predefined process. Instead, they are typically used in an 'ad hoc', user driven, way.

Still, there could be strong non-functional arguments in favor of SOAP/WSDL based WMS services. A key OSB aspect is the provision of secure communication. If security demands are high, e.g. in the context of disaster management, one could opt for deploying WMS services in the OSB WUS environment, thus profiting from an existing security infrastructure. Also, deploying a service according to OSB WUS standards could enforced by government rules. In those cases, one will have to choose between the two available solutions to get WMS services OSB compliant.

Both approaches were applied in the test bed (see paragraph 4.1). The WMS services were implemented using an 'infrastructural architecture'. Operation requests and responses from off-the-shelf WMS products, already wrapped in their native communication protocol (HTTP), are wrapped for the second time, now in a SOAP body. In this way, it becomes completely transparent that the message contains a WMS request or response. They are one-to-one encoded as an XML CDATA block, so validation is not applicable. Indeed, in that way any message can be transmitted. The 'communication contract' is in fact a trivial one. In a purely technical sense this approach may be seen as a satisfactory solution. On a functional level, however, there are a few fundamental issues, that prevent this to be a suitable solution in a practical environment.

In the test bed, the second approach, the 'functional architecture', was only applied for WFS services, but Appendix D describes how this could work for WMS 1.3.0. Here, the operation requests and responses proper are wrapped in a SOAP body as an XML structure. This XML structure has to conform to a schema, specified in the standards. This enables a strongly typed communication contract (a WSDL description) between client and server. This is the way SOAP/WSDL is meant to be used and the architecture considered when answering the research questions.

As said, the infrastructural solution may be feasible, but, as indicated above, is not advised. On the other hand, before being able to use the alternative, functional approach a provision has to be made for some missing specifications in the current WMS standard. For some WMS operations, standardized SOAP bindings and XML encodings are not yet available. Partly, they are on the road

map for the next WMS version, but results probably will not be available shortly. If WMS services are indeed to be used in an OSB environment, the new Dutch WMS profile, due in a few months and based on WMS 1.3.0 (INSPIRE View Service), will have to provide for a partially local standard. Since OSB services are strictly used in a Dutch context, this should not be seen as a major drawback.

The combination of standards that is recommended is:

- WMS 1.3.0
  This is according to the next Dutch WMS profile.

- SOAP 1.1
  This version is according to both INSPIRE in the context of its View Services and the OSB WUS standards.

- WS-I Basic Profile 1.2 (Addressing)
  Also according to both INSPIRE and OSB.

- WS-I Basic Profile 1.1 (encoding binary data)
  This version is according to the OSB 1.1 standard. Binary data are transmitted as part of the SOAP body, base64 encoded. In OSB 2.0, the version will be upgraded to WS-I BP 1.2. INSPIRE in its SOAP Framework already demands version 1.2, mainly because it is relying on SOAP attachments and MTOM.
  The choice of WS-I BP version in this context is effectively a choice of OSB version.

- WS-I Basic Profile 1.1 (SOAP, WSDL, XML)
  OSB 1.1 requires this version. INSPIRE only in general mentions version 1.2, but probably merely because of the addressing and message optimization requirements. With regard to SOAP, WSDL and XML, selecting WS-I BP 1.1 will not lead to a conflict.

- Encoding: document literal

7.2 WFS

The requirement for having WFS services available using SOAP transport is far more obvious, e.g. for seamless incorporation in a BPEL orchestration. The test bed has shown that it is almost entirely possible to comply to both OGC and OSB standards. The following combination of standards is advised:

- WFS 1.1.0 → WFS 2.0.0
  WFS 1.1.0 is the most recent version that is supported by WFS products on the market, so this has to be the current choice. As soon as WFS 2.0.0 is final and supported by WFS products, it is advised to upgrade to this version.

- SOAP 1.1
  This is the standard, that is prescribed by OSB WUS. It is not according to the OGC WFS 1.1.0 standard, but this should not be seen as an important issue. Indeed, the WFS 2.0.0 standard again specifies the use of SOAP 1.1.

- WS-I Basic Profile 1.2 (Addressing)
  This profile is only mentioned by OSB WUS, not by the WFS standard.

- WS-I Basic Profile 1.1 (SOAP, WSDL, XML)
  This is according to OSB WUS. The WFS standard does not mention the WS-I BP version normatively. However, the OGC OWS Common 1.2 standard requires version WS-I BP 1.2. This is significant mainly for efficient processing of binary data, so does not apply to WFS.

- Encoding: document literal

7.3 Test bed availability

The test bed WFS service, based on the functional architecture, successfully passed the OSB compliance test. The service is publicly available and published in the OSB Service Register, as well as its compliance report. The implemented OSB WMS service, based on the infrastructural architecture, is publicly available as well. All details can be found in paragraph 4.2.5.

8 References

EU. 2009. *Commission regulation (EC) No 976/2009, implementing Directive 2007/2/EC […] as regards the Network Services*, 19 October 2009
http://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=OJ:L:2009:274:0009:0018:EN:PDF

Geonovum. 2007. *Nederlands WMS profiel*, Version 1.1, December 2007
http://www.geonovum.nl/sites/default/files/Nederlands_WMS_profiel_1_1_def.pdf

Geonovum. 2009. *Offerteaanvraag Testbed geo-services op de Overheidsservicebus en register*, versie 1.0, 30 July 2009

INSPIRE. 2008-b. *INSPIRE Network Services Architecture*, 19-07-2008
http://inspire.jrc.ec.europa.eu/reports/ImplementingRules/network/D3_5_INSPIRE_NS_Architecture_v3-0.pdf

INSPIRE. 2009. *Technical Guidance to implement INSPIRE View Services*, 2009-07-20, status Second Version
http://inspire.jrc.ec.europa.eu/documents/Network_Services/Technical%20Guidance%20View%20Services%20v%202.0.pdf

ISO/TC 211. 2005. *Text for ISO 19128 Geographic information - Web Map Server Interface,* 2005-01-25 [i.e. WMS 1.3.0]

ISO/TC 211. 2009-a. *Final text of ISO/CD 19143, Geographic information — Filter encoding,* 2009-02-18

ISO/TC 211. 2009-b. *Final text of ISO/CD 19142, Geographic information — Web feature service*, 2009-03-01 [i.e. WFS 2.0]

OGC. 2002. *Web Map Service Implementation Specification,* version 1.1.1, 2002-01-16 (OGC 01-068r3)
http://portal.opengeospatial.org/files/?artifact_id=1081&version=1&format=pdf

OGC. 2004. *OGC Web Map Service Interface,* version 1.3.0, 2004-01-20, (OGC 03-109r1)
http://portal.opengeospatial.org/files/?artifact_id=4756

OGC. 2005-a. *Web Feature Service Implementation Specification,* version 1.1.0, 03-05-2005 (OGC 04-094)
http://portal.opengeospatial.org/files/?artifact_id=8339

OGC. 2005-b. *OpenGIS Filter Encoding Implementation Specification*, version 1.1.0, 03-05-2005 (OGC 04-095)
http://portal.opengeospatial.org/files/?artifact_id=8340

OGC. 2009. *Candidate Revision: OGC Web Services Common 1.2,* 2009-04-20 (OGC 06-121r7)

OSB. 2008-a. *OSB Koppelvlakstandaard WUS,* version 1.1, 15-10-2008
http://www.logius.nl/fileadmin/logius/product/digikoppeling/koppelvlakstandaarden/Koppelvlakstandaard%20WUS%201.1.pdf

OSB. 2008-b. *OSB Best Practices WUS OSB versie 1.1*, document version 1.1, 6 October 2008
http://www.logius.nl/fileadmin/OSB/OSB_Best_Practices_WUS_1_1.pdf

OSB. 2008-c. *Gebruikershandleiding OSB Compliancevoorziening WUS*, version 1.0, 5 November 2008
http://www.logius.nl/fileadmin/OSB/WUS_CV_1_0_081120.pdf

OSB. 2010-a. *Gebruikershandleiding Digikoppeling Serviceregister*, versie 1.2, 6 January 2010
http://www.logius.nl/fileadmin/logius/product/digikoppeling/service_register/Digikoppeling%20Gebruikershandleiding%20Serviceregister%20v1.1.pdf

OSB. 2010-b. *Koppelvlakstandaard WUS Voor Digikoppeling 2.0*, (document) version 1.1, 6 January 2010
http://www.logius.nl/fileadmin/logius/product/digikoppeling/koppelvlakstandaarden/Koppelvlakstandaard%20WUS%202.0.pdf

OSB. 2010-c. *Gebruik en achtergrond van OSB certificaten*, version 1.1, 6 January 2010
http://www.logius.nl/fileadmin/logius/product/digikoppeling/service_register/Gebruik%20en%20Achtergrond%20Digikoppeling%20Certificaten%20v1.1.pdf

Ravi. 2006. *Nederlands WFS Profiel*, Version 1.0, 27 October 2006
http://www.geonovum.nl/sites/default/files/Nederlands_WFS_Profiel10.pdf

StUF. 2009-a. Protocolbindingen voor StUF, Version 3.00, 25-06-2009
https://www.surfgroepen.nl/sites/stuf/Shared%20Documents/4_StUF_Standaarden/Protocolbindingen/stuf.bindingen.030000.pdf

StUF. 2009-b. Standaard Uitwisseling Formaat, Version 3.01, 06-10-2009
https://www.surfgroepen.nl/sites/stuf/Shared%20Documents/4_StUF_Standaarden/StUF%2003.01%20(In%20Gebruik)/stuf030103.pdf

Villa, Matteo and Di Matteo, Giovanni. 2008, *INSPIRE Network Services SOAP Framework*, 2008
http://inspire.jrc.ec.europa.eu/reports/ImplementingRules/network/INSPIRE_NETWORK_SERVICES_SOAP_Framework.pdf

WS-I. 2004. *Basic Profile Version 1.1*, Final Material, 2008-08-24
http://www.ws-i.org/Profiles/BasicProfile-1.1-2004-08-24.html

WS-I. 2007. *Basic Profile Version 1.2*, Board Approval Draft, 2007-03-28
http://www.ws-i.org/Profiles/BasicProfile-1.2.html

Appendix A Abbreviations

EXI	Efficient XML Interchange
GML	Geography Markup Language
HTTP	Hypertext Transfer Protocol
ICTU	ICT Uitvoeringsorganisatie (Central ICT organization of the Dutch Government)
INSPIRE	Infrastructure for Spatial Information in the European Community
KVP	Key(word) Value Pair
MTOM	Message Transmission Optimization Mechanism
NGR	Nationaal Georegister
NEN	Nederlands Normalisatie Instituut (Netherlands Standardization Institute)
NORA	Nederlandse Overheid Referentie Architectuur (Netherlands Government Reference Architecture)
OGC	Open Geospatial Consortium
OSR	OSB Service Register
OSB	OverheidsServiceBus (Government Service Bus), now Digikoppeling
OWS	OGC Web Service
POX	Plain old XML
REST	Representational State Transfer
RD	Rijksdriehoekstelsel (Dutch national grid)
SOAP	Simple Object Access Protocol
WFS	Web Feature Service

WMS	Web Map Service
WSDL	Web Services Description Language
WUS	WSDL/UDDI/SOAP
XOP	XML-binary Optimized Packaging

Appendix B WSDL files - use case 1

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- Published by JAX-WS RI at http://jax-ws.dev.java.net. RI's version is JAX-WS RI 2.1.7-hudson-48-. -->
<wsdl:definitions xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/" xmlns:tns="http://atosorigin.com/wms"
        xmlns:wsa="http://www.w3.org/2005/08/addressing" xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
        xmlns:err="http://atosorigin.com/Error" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
        name="geonovum-wms-abstract-definition.wsdl" targetNamespace="http://atosorigin.com/wms">
    <!--### TYPES DEFINITIONS ### -->
    <wsdl:types>
        <xsd:schema elementFormDefault="qualified">
            <xsd:import namespace="http://atosorigin.com/Error"
                    schemaLocation="http://212.159.196.130:80/tgmServer/RelayService?xsd=1"/>
            <xsd:import namespace="http://atosorigin.com/wms"
                    schemaLocation="http://212.159.196.130:80/tgmServer/RelayService?xsd=2"/>
            <xsd:import namespace="http://www.w3.org/2005/08/addressing"
                    schemaLocation="http://www.w3.org/2006/03/addressing/ws-addr.xsd"/>
        </xsd:schema>
    </wsdl:types>
    <!--### MESSAGES DEFINITIONS ###-->
    <wsdl:message name="RelayRequestMessage">
        <wsdl:part element="tns:RelayRequest" name="RelayRequest"/>
    </wsdl:message>
    <wsdl:message name="RelayResponseMessage">
        <wsdl:part element="tns:RelayResponse" name="RelayResponse"/>
    </wsdl:message>
    <wsdl:message name="wsaHeaders">
        <wsdl:part element="wsa:To" name="wsaTo"/>
        <wsdl:part element="wsa:Action" name="wsaAction"/>
        <wsdl:part element="wsa:MessageID" name="wsaMessageID"/>
        <wsdl:part element="wsa:RelatesTo" name="wsaRelatesTo"/>
        <wsdl:part element="wsa:ReplyTo" name="wsaReplyTo"/>
    </wsdl:message>
    <!--Fault messages-->
    <wsdl:message name="FoutMelding">
        <wsdl:part element="err:Fout" name="parameters"/>
    </wsdl:message>
    <!--### PORTTYPES DEFINITIONS ###-->
    <wsdl:portType name="RelayServicePortType">
        <wsdl:operation name="Relay">
            <wsdl:input message="tns:RelayRequestMessage"
                    wsa:Action="https://www.atosorigin.com/osbtestbed/RelayServiceRequest"/>
            <wsdl:output message="tns:RelayResponseMessage"
                    wsa:Action="https://www.atosorigin.com/osbtestbed/RelayServiceResponse"/>
            <wsdl:fault message="err:FoutMelding" name="FoutMelding"/>
        </wsdl:operation>
    </wsdl:portType>
</wsdl:definitions>
```

Listing 22 – Use case 1, WSDL Abstract Definition[19]

[19] http://212.159.196.130/tgmServer/RelayService?wsdl=1

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- Published by JAX-WS RI at http://jax-ws.dev.java.net. RI's version is JAX-WS RI 2.1.7-hudson-48-. -->
<wsdl:definitions xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/" xmlns:tns="http://atosorigin.com/wms"
        xmlns:wsaw="http://www.w3.org/2006/05/addressing/wsdl" xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
        xmlns:wsi="http://ws-i.org/schemas/conformanceClaim/" name="geonovum-wfs-servicebinding.wsdl"
        targetNamespace="http://atosorigin.com/wms">
    <wsdl:import location="http://212.159.196.130:80/tgmServer/RelayService?wsdl=1"
                namespace="http://atosorigin.com/wms"/>
    <!--### BINDINGS DEFINITIONS ### -->
    <wsdl:binding name="RelayServicePortBinding" type="tns:RelayServicePortType">
        <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>
        <wsaw:UsingAddressing wsdl:required="true"/>
        <!-- WMS Relay operation-->
        <wsdl:operation name="Relay">
            <soap:operation soapAction="https://www.atosorigin.com/osbtestbed/RelayService"/>
            <wsdl:input>
                <soap:header message="tns:wsaHeaders" part="wsaTo" use="literal"/>
                <soap:header message="tns:wsaHeaders" part="wsaAction" use="literal"/>
                <soap:header message="tns:wsaHeaders" part="wsaMessageID" use="literal"/>
                <soap:header message="tns:wsaHeaders" part="wsaRelatesTo" use="literal"/>
                <soap:header message="tns:wsaHeaders" part="wsaReplyTo" use="literal"/>
                <soap:body use="literal"/>
            </wsdl:input>
            <wsdl:output>
                <soap:header message="tns:wsaHeaders" part="wsaTo" use="literal"/>
                <soap:header message="tns:wsaHeaders" part="wsaAction" use="literal"/>
                <soap:header message="tns:wsaHeaders" part="wsaMessageID" use="literal"/>
                <soap:header message="tns:wsaHeaders" part="wsaRelatesTo" use="literal"/>
                <soap:header message="tns:wsaHeaders" part="wsaReplyTo" use="literal"/>
                <soap:body use="literal"/>
            </wsdl:output>
            <wsdl:fault name="FoutMelding">
                <soap:fault name="FoutMelding" use="literal"/>
            </wsdl:fault>
        </wsdl:operation>
    </wsdl:binding>
    <!--### SERVICE DEFINITIONS ###-->
    <wsdl:service name="RelayService">
        <wsdl:documentation>
            <wsi:Claim conformsTo="http://ws-i.org/profiles/basic/1.1"/>
        </wsdl:documentation>
        <wsdl:port binding="tns:RelayServicePortBinding" name="RelayServicePort">
            <soap:address location="http://212.159.196.130:80/tgmServer/RelayService"/>
        </wsdl:port>
    </wsdl:service>
</wsdl:definitions>
```

Listing 23 - Use case 1, WSDL Service binding[20]

[20] http://212.159.196.130/tgmServer/RelayService?wsdl

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- Published by JAX-WS RI at http://jax-ws.dev.java.net. RI's version is JAX-WS RI 2.1.7-hudson-48-. -->
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:atos="http://www.atosorigin.com/wms"
            targetNamespace="http://www.atosorigin.com/wms">
    <xsd:element name="request" type="atos:RelayRequestType"/>
    <xsd:complexType name="RelayRequestType">
        <xsd:sequence>
            <xsd:element name="headers" type="atos:headersType"/>
            <xsd:element name="method" type="xsd:string"/>
            <xsd:element name="protocol" type="xsd:string"/>
            <xsd:element name="queryString" type="xsd:string"/>
            <xsd:element name="requestURL" type="xsd:string"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType name="headersType">
        <xsd:sequence>
            <xsd:element name="header" type="atos:headerType" maxOccurs="unbounded"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType name="headerType">
        <xsd:sequence>
            <xsd:element name="headerName" type="xsd:string"/>
            <xsd:element name="headerValue" type="xsd:string"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:element name="response" type="atos:RelayResponseType"/>
    <xsd:complexType name="RelayResponseType">
        <xsd:sequence>
            <xsd:element name="content-length" type="xsd:string"/>
            <xsd:element name="content-type" type="xsd:string"/>
            <xsd:element name="headers" type="atos:headersType"/>
            <xsd:element name="payload" type="xsd:string"/>
            <xsd:element name="payload-type" type="xsd:string"/>
        </xsd:sequence>
    </xsd:complexType>
</xsd:schema>
```

Listing 24 – Use case 1, geonovum-wms.xsd[21]

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- Published by JAX-WS RI at http://jax-ws.dev.java.net. RI's version is JAX-WS RI 2.1.7-hudson-48-. -->
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:err="http://atosorigin.com/Error"
            targetNamespace="http://atosorigin.com/Error">
    <xsd:element name="Fout" type="err:FoutType"/>
    <xsd:complexType name="FoutType">
        <xsd:sequence>
            <xsd:element name="foutMelding" nillable="true" type="xsd:string"/>
            <xsd:element name="foutMeldingCode" nillable="true" type="xsd:string"/>
        </xsd:sequence>
    </xsd:complexType>
</xsd:schema>
```

Listing 25 – Use case 1, geonovum-foutmelding.xsd[22]

[21] http://212.159.196.130/tgmServer/RelayService?xsd=2

[22] http://212.159.196.130/tgmServer/RelayService?xsd=1

Appendix C WSDL files - use case 2

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- Published by JAX-WS RI at http://jax-ws.dev.java.net. RI's version is JAX-WS RI 2.1.7-hudson-48-. -->
<wsdl:definitions xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:tns="http://atosorigin.com"
    xmlns:osbxsd="http://service.voorbeeld.osb.gbo.overheid.nl/200706"
    xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/" xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
    xmlns:wsa="http://www.w3.org/2005/08/addressing" xmlns:wsaw="http://www.w3.org/2006/05/addressing/wsdl"
    xmlns:wfs="http://www.opengis.net/wfs" xmlns:wsam="http://www.w3.org/2007/05/addressing/metadata"
    xmlns:err="http://www.w3.org/2007/05/addressing/metadata" targetNamespace="http://atosorigin.com"
    name="geonovum-wfs-abstract-definition.wsdl">
    <!--### TYPES DEFINITIONS ### -->
    <wsdl:types>
        <xsd:schema elementFormDefault="qualified">
            <xsd:import namespace="http://atosorigin.com/Error"
                    schemaLocation="http://212.159.196.130:80/tgmWebservice/WFSService?xsd=21"/>
            <xsd:import namespace="http://www.opengis.net/wfs"
                    schemaLocation="http://212.159.196.130:80/tgmWebservice/WFSService?xsd=47"/>
        </xsd:schema>
    </wsdl:types>
    <!--### MESSAGES DEFINITIONS ###-->
    <!--GetCapabilities messages-->
    <wsdl:message name="wfsGetCapabilitiesRequest">
        <wsdl:part name="wfsGetCapabilities" element="wfs:GetCapabilities"/>
    </wsdl:message>
    <wsdl:message name="wfsGetCapabilitiesResponse">
        <wsdl:part name="wfsWFS_Capabilities" element="wfs:WFS_Capabilities"/>
    </wsdl:message>
    <!--GetFeature messages-->
    <wsdl:message name="wfsGetFeatureRequest">
        <wsdl:part name="wfsGetFeature" element="wfs:GetFeature"/>
    </wsdl:message>
    <wsdl:message name="wfsGetFeatureResponse">
        <wsdl:part name="wfsFeatureCollection" element="wfs:FeatureCollection"/>
    </wsdl:message>
    <!--WSA Headers message-->
    <wsdl:message name="wsaHeaders">
        <wsdl:part name="wsaTo" element="wsa:To"/>
        <wsdl:part name="wsaAction" element="wsa:Action"/>
        <wsdl:part name="wsaMessageID" element="wsa:MessageID"/>
        <wsdl:part name="wsaRelatesTo" element="wsa:RelatesTo"/>
        <wsdl:part name="wsaReplyTo" element="wsa:ReplyTo"/>
    </wsdl:message>
    <!--Fault messages -->
    <wsdl:message name="FoutMelding">
        <wsdl:part name="parameters" element="err:FoutMelding"/>
    </wsdl:message>
    <!--### PORTTYPES DEFINITIONS ###-->
    <!--WFS PortType-->
    <wsdl:portType name="wfsServicePortType">
        <!--WFS GetCapabilities-->
        <wsdl:operation name="getCapabilities">
            <wsdl:input message="tns:wfsGetCapabilitiesRequest"
                wsaw:Action="https://www.atosorigin.com/osbtestbed/wfsgetCapabilitiesRequest"/>
            <wsdl:output message="tns:wfsGetCapabilitiesResponse"
                wsaw:Action="https://www.atosorigin.com/osbtestbed/wfsgetCapabilitiesResponse"/>
            <wsdl:fault name="FoutMelding" message="tns:FoutMelding"/>
```

```
        </wsdl:operation>
        <!--WFS GetFeature-->
        <wsdl:operation name="getFeature">
            <wsdl:input message="tns:wfsGetFeatureRequest"
                wsaw:Action="https://www.atosorigin.com/osbtestbed/wfsgetFeatureRequest"/>
            <wsdl:output message="tns:wfsGetFeatureResponse"
                wsaw:Action="https://www.atosorigin.com/osbtestbed/wfsgetFeatureResponse"/>
            <wsdl:fault name="FoutMelding" message="tns:FoutMelding"/>
        </wsdl:operation>
    </wsdl:portType>
</wsdl:definitions>
```

Listing 26 – Use case 2, WSDL Abstract Definition[23]

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- Published by JAX-WS RI at http://jax-ws.dev.java.net. RI's version is JAX-WS RI 2.1.7-hudson-48-. -->
<wsdl:definitions xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
        xmlns:osb="http://atosorigin.com" xmlns:wsi="http://ws-i.org/schemas/conformanceClaim/"
        xmlns:tns="http://service.compliance.osb.gbo.overheid.nl/200707/osbcomplianceservice"
        xmlns:wsaw="http://www.w3.org/2006/05/addressing/wsdl"
        targetNamespace="http://service.compliance.osb.gbo.overheid.nl/200707/osbcomplianceservice"
        name="geonovum-wfs-servicebinding.wsdl">
    <wsdl:import namespace="http://atosorigin.com"
            location="http://212.159.196.130:80/tgmWebservice/WFSService?wsdl=1"/>
    <!--### BINDINGS DEFINITIONS ### -->
    <wsdl:binding name="wfsServicePortBinding" type="osb:wfsServicePortType">
        <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>
        <wsaw:UsingAddressing wsdl:required="true"/>
        <!--WFS GetCapabilities-->
        <wsdl:operation name="getCapabilities">
            <soap:operation soapAction="hhttps://www.atosorigin.com/osbtestbed/wfsgetCapabilities"/>
            <wsdl:input>
                <soap:header message="osb:wsaHeaders" part="wsaTo" use="literal"/>
                <soap:header message="osb:wsaHeaders" part="wsaAction" use="literal"/>
                <soap:header message="osb:wsaHeaders" part="wsaMessageID" use="literal"/>
                <soap:header message="osb:wsaHeaders" part="wsaRelatesTo" use="literal"/>
                <soap:header message="osb:wsaHeaders" part="wsaReplyTo" use="literal"/>
                <soap:body use="literal"/>
            </wsdl:input>
            <wsdl:output>
                <soap:header message="osb:wsaHeaders" part="wsaTo" use="literal"/>
                <soap:header message="osb:wsaHeaders" part="wsaAction" use="literal"/>
                <soap:header message="osb:wsaHeaders" part="wsaMessageID" use="literal"/>
                <soap:header message="osb:wsaHeaders" part="wsaRelatesTo" use="literal"/>
                <soap:header message="osb:wsaHeaders" part="wsaReplyTo" use="literal"/>
                <soap:body use="literal"/>
            </wsdl:output>
            <wsdl:fault name="FoutMelding">
                <soap:fault name="FoutMelding" use="literal"/>
            </wsdl:fault>
        </wsdl:operation>
        <!--WFS GetFeature-->
```

[23] http://212.159.196.130/tgmWebservice/WFSService?wsdl=1 . Note: compliance operations toUpperCase and ping are not shown here.

```
        <wsdl:operation name="getFeature">
            <soap:operation soapAction="https://www.atosorigin.com/osbtestbed/wfsgetFeature"/>
            <wsdl:input>
                <soap:header message="osb:wsaHeaders" part="wsaTo" use="literal"/>
                <soap:header message="osb:wsaHeaders" part="wsaAction" use="literal"/>
                <soap:header message="osb:wsaHeaders" part="wsaMessageID" use="literal"/>
                <soap:header message="osb:wsaHeaders" part="wsaRelatesTo" use="literal"/>
                <soap:header message="osb:wsaHeaders" part="wsaReplyTo" use="literal"/>
                <soap:body use="literal"/>
            </wsdl:input>
            <wsdl:output>
                <soap:header message="osb:wsaHeaders" part="wsaTo" use="literal"/>
                <soap:header message="osb:wsaHeaders" part="wsaAction" use="literal"/>
                <soap:header message="osb:wsaHeaders" part="wsaMessageID" use="literal"/>
                <soap:header message="osb:wsaHeaders" part="wsaRelatesTo" use="literal"/>
                <soap:header message="osb:wsaHeaders" part="wsaReplyTo" use="literal"/>
                <soap:body use="literal"/>
            </wsdl:output>
            <wsdl:fault name="FoutMelding">
                <soap:fault name="FoutMelding" use="literal"/>
            </wsdl:fault>
        </wsdl:operation>
    </wsdl:binding>
    <!--### SERVICE DEFINITIONS ###-->
    <wsdl:service name="wfsService">
        <wsdl:documentation>
            <wsi:Claim conformsTo="http://ws-i.org/profiles/basic/1.1"/>
        </wsdl:documentation>
        <wsdl:port name="wfsServicePort" binding="tns:wfsServicePortBinding">
            <soap:address location="http://212.159.196.130:80/tgmWebservice/WFSService"/>
        </wsdl:port>
    </wsdl:service>
</wsdl:definitions>
```

Listing 27 – Use case 2, WSDL Service binding[24]

[24] http://212.159.196.130/tgmWebservice/WFSService?wsdl

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:err="http://atosorigin.com/Error"
    targetNamespace="http://atosorigin.com/Error">
    <xsd:element name="Fout" type="err:FoutType"/>
    <xsd:complexType name="FoutType">
        <xsd:sequence>
            <xsd:element name="foutMelding" type="xsd:string" nillable="true"/>
            <xsd:element name="foutMeldingCode" type="xsd:string" nillable="true"/>
        </xsd:sequence>
    </xsd:complexType>
</xsd:schema>
```

Listing 28 – Use case 2, geonovum-foutmelding.xsd[25]

[25] http://212.159.196.130/tgmWebservice/WFSService?xsd=21 . Exception handling was out of scope for this project, so in use case 2 the same straightforward exception reporting is used as in use case 1. However, (OGC 2009) prescribes a specific ExceptionReport element to be used here. In the test bed implementation, an ExceptionReport is returned as an alternative 'legal response' in the SOAP body, not as as a SOAP fault. So, with respect to exception handling the test bed implementation does not conform to the standards.

Appendix D WMS 1.3.0 functional approach

For a detailed comparison between the two architectures, this appendix partially shows the results if we would have implemented the WMS 1.3.0 message exchange according to the alternative, functional solution. Only the SOAP messages for the GetCapabilities Request are shown.

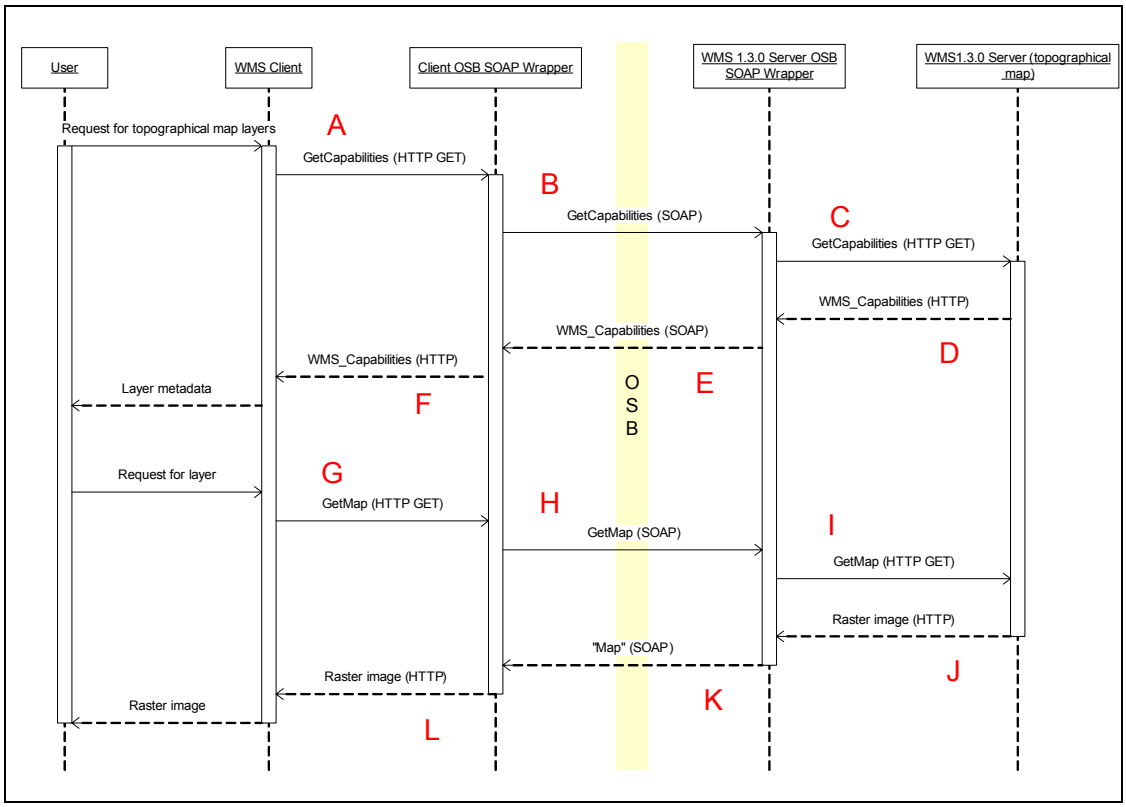

Figure 14 - Sequence diagram - Use case 1, part 2 (alternative solution)

A/C. GetCapabilities (HTTP/GET)

This request is still the same as the one that is implemented in the test bed scenario.

```
http://212.159.196.130:81/cgi-bin/mapserv.exe?MAP=D:%5CData%5CTOP10NL.map&VERSION=1.3.0&
SERVICE=WMS&REQUEST=GetCapabilities&LANGUAGE=dut
```

Listing 29

B. GetCapabilities (SOAP)

For this, an ad hoc XML-encoding had to be specified for the keyword/value pairs in the HTTP/GET request. This specification can be recognized by the use of is the “atos” namespace. Here, a straightforward approach has been chosen: all keywords are simply transformed into attributes, except for the MAP parameter which is not a WMS parameter, but effectively a part of the address. The referred schema (atoswms.xsd) has not been specified.

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<SOAP-ENV:Envelope
        xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
        xmlns:wsa="http://www.w3.org/2005/08/addressing">
    <SOAP-ENV:Header>
        <wsa:To>http://212.159.196.130:81/cgi-bin/mapserv.exe?map=D:\Data\TOP10NL.map</wsa:To>
        <wsa:ReplyTo>
            <wsa:Address>http://www.w3.org/2005/08/addressing/anonymous</wsa:Address>
        </wsa:ReplyTo>
        <wsa:MessageID>http://www.atosorigin.com/174655018110122</wsa:MessageID>
        <wsa:Action>https://www.atosorigin.com/osbtestbed/wmsgetCapabilitiesRequest</wsa:Action>
    </SOAP-ENV:Header>
    <SOAP-ENV:Body>
        <GetCapabilities xmlns:atos="http://www.atosorigin.com/wms13"
                xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                xsi:schemaLocation="http://www.atosorigin.com/wms13
                                    http://212.159.196.130:81/schemas/atoswms.xsd"
                version="1.3.0"  service="WMS" language="dut">
        </GetCapabilities>
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Listing 30

D/F. WMS_Capabilities (HTTP)

The resulting WMS_Capabilities document is the same as shown in use case 1, part 2 (Listing 12).

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<WMS_Capabilities version="1.3.0" xmlns="http://www.opengis.net/wms"
    xmlns:sld="http://www.opengis.net/sld" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:ms="http://mapserver.gis.umn.edu/mapserver"
    xsi:schemaLocation="
        http://www.opengis.net/wms http://schemas.opengis.net/wms/1.3.0/capabilities_1_3_0.xsd
        http://www.opengis.net/sld http://schemas.opengis.net/sld/1.1.0/sld_capabilities.xsd
        http://mapserver.gis.umn.edu/mapserver
            http://212.159.196.130:81/cgi-bin/mapserv.exe?map=D:\Data\TOP10NL.map&amp;
                service=WMS&amp;version=1.3.0&amp;request=GetSchemaExtension">
  …
  <Capability>
        …
        <INSPIRE:ViewCapabilities xmlns:INSPIRE="http://www.inspire.org"
                xsi:schemaLocation="http://www.inspire.org http://212.159.196.130:81/schemas/inspireview.xsd">
            <INSPIRE:Languages>
                <INSPIRE:Language default="true">eng</INSPIRE:Language>
                <INSPIRE:Language>dut</INSPIRE:Language>
            </INSPIRE:Languages>
            <INSPIRE:CurrentLanguage>dut</INSPIRE:CurrentLanguage>
        </INSPIRE:ViewCapabilities>
    …
  </Capability>
  …
</WMS_Capabilities>
```

Listing 31

E. WMS_Capabilities (SOAP)

The WMS_Capabilities document is transmitted in the same way as the WFS_Capablities document in use case 2:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
    <SOAP-ENV:Header>
        <wsa:To>http://www.w3.org/2005/08/addressing/anonymous</wsa:To>
        <wsa:RelatesTo>http://www.atosorigin.com/174655018110122</wsa:RelatesTo>
        <wsa:MessageID>http://www.atosorigin.com/174656534220131</wsa:MessageID>
        <wsa:From>http://www.w3.org/2005/08/addressing/anonymous</wsa:From>
        <wsa:Action>https://www.atosorigin.com/osbtestbed/wmsgetCapabilitiesResponse</wsa:Action>
    </SOAP-ENV:Header>
    <SOAP-ENV:Body>
        <WMS_Capabilities version="1.3.0" xmlns="http://www.opengis.net/wms"
            xmlns:sld="http://www.opengis.net/sld" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
            xmlns:ms="http://mapserver.gis.umn.edu/mapserver"
            xsi:schemaLocation="
                http://www.opengis.net/wms http://schemas.opengis.net/wms/1.3.0/capabilities_1_3_0.xsd
                http://www.opengis.net/sld http://schemas.opengis.net/sld/1.1.0/sld_capabilities.xsd
                http://mapserver.gis.umn.edu/mapserver
                    http://212.159.196.130:81/cgi-bin/mapserv.exe?map=D:\Data\TOP10NL.map&amp;
                        service=WMS&amp;version=1.3.0&amp;request=GetSchemaExtension">
            …
        </WMS_Capabilities>
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Listing 32

Appendix E Sequence diagrams test bed components

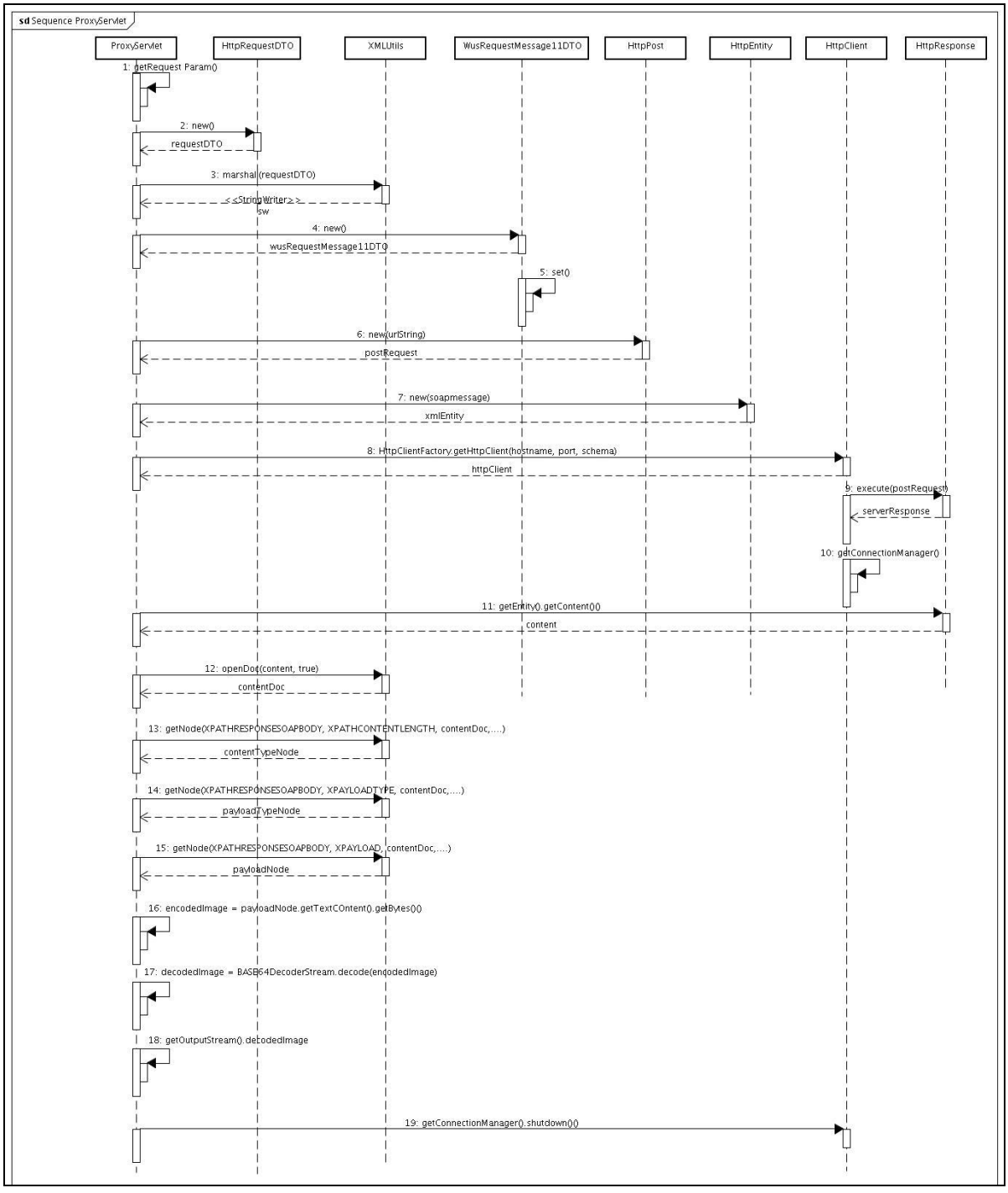

Figure 15 – Sequence diagram ProxyServlet (use case 1)

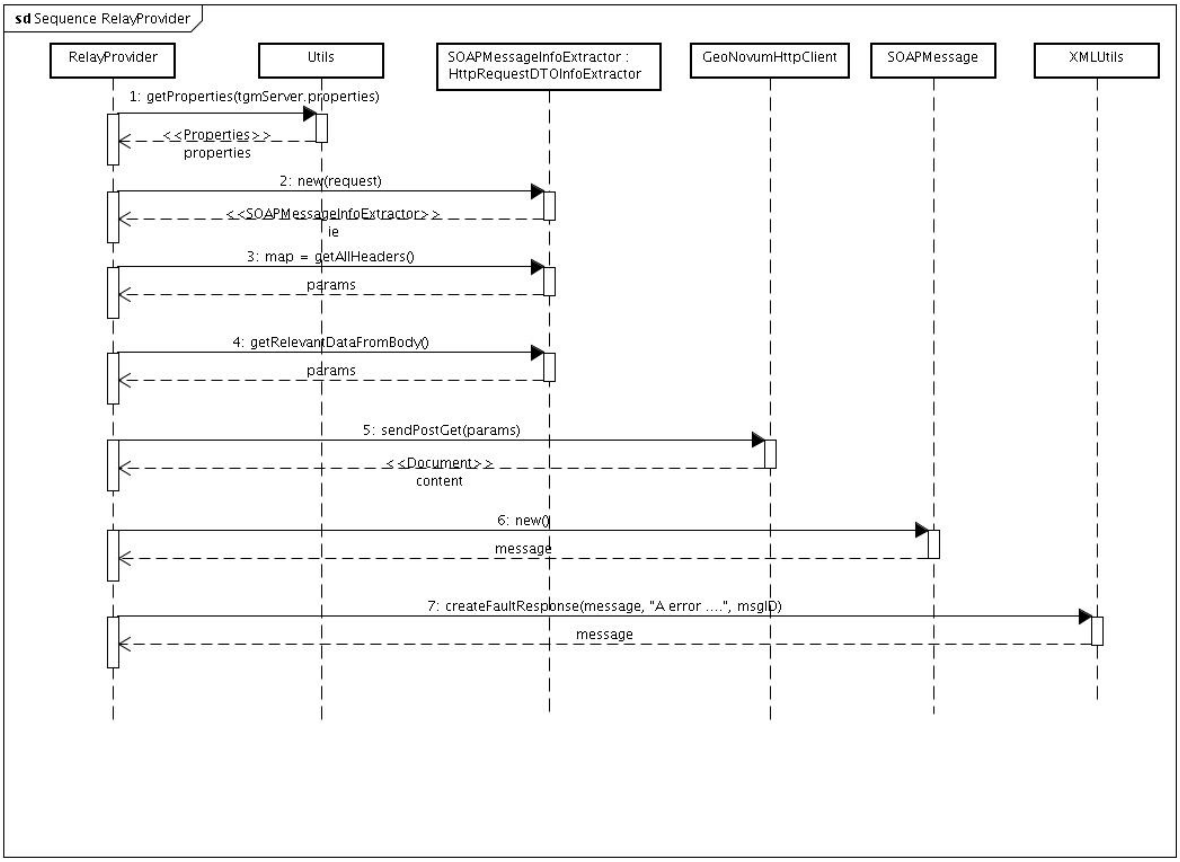

Figure 16 - Sequence diagram RelayProvider (use case 1)

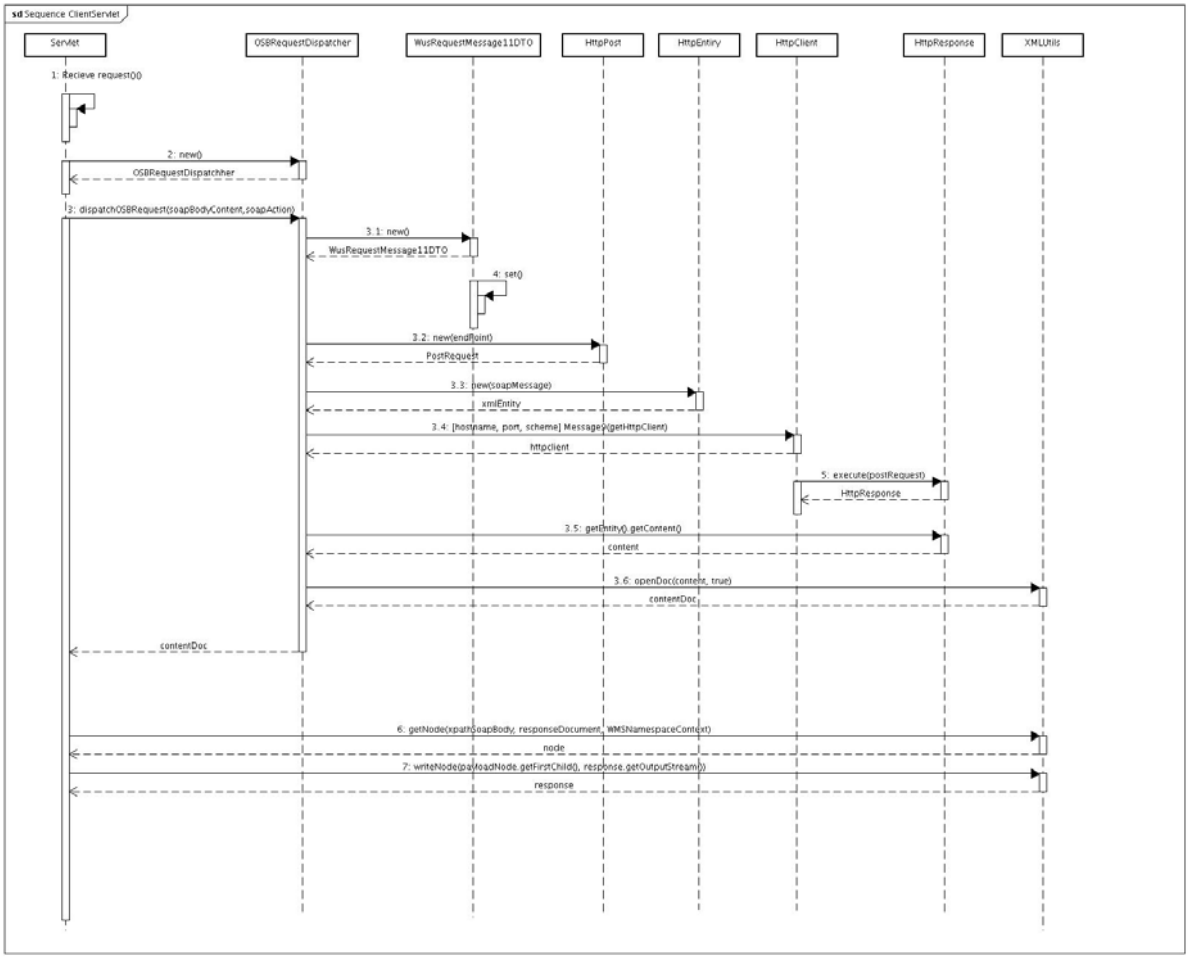

Figure 17 – Sequence diagram ClientServlet (use case 2)

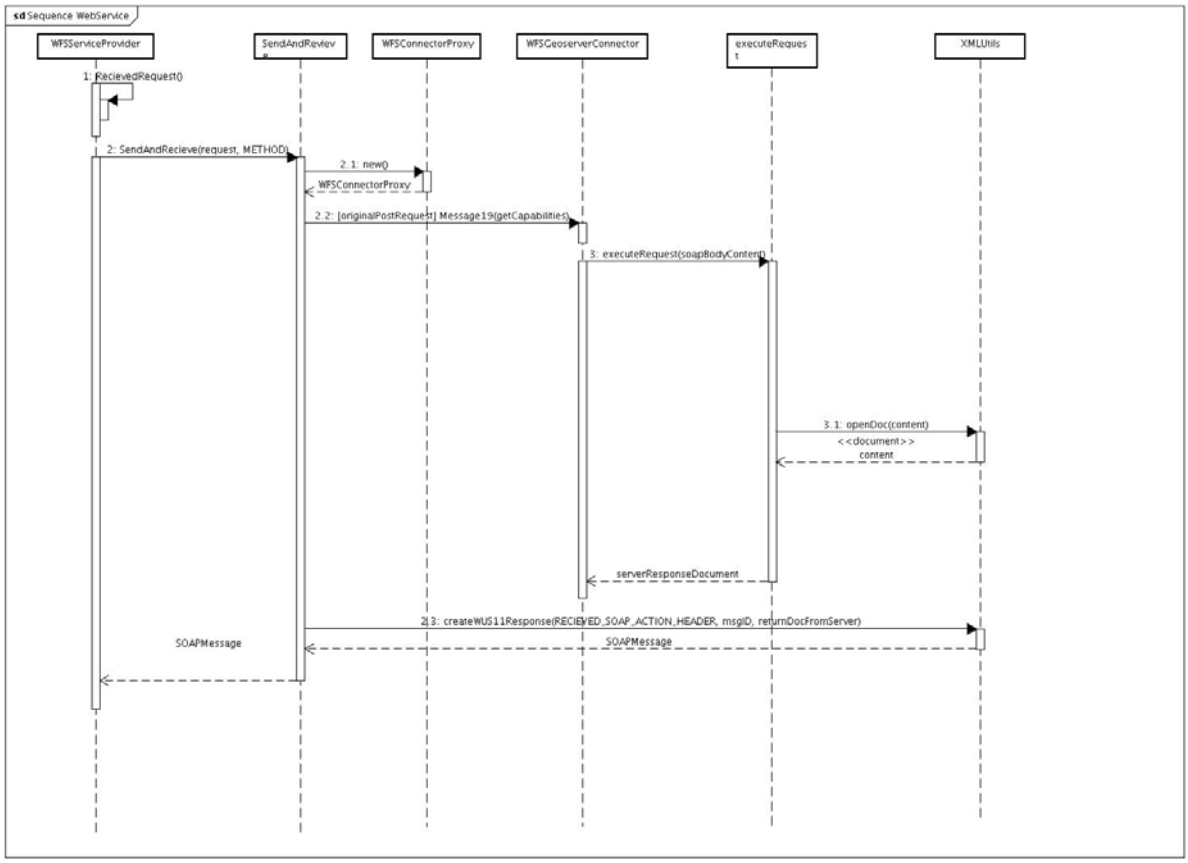

Figure 18 – Sequence diagram Webservice (use case 2)

Appendix F OSB Compliance Report

This appendix contains a summary from the complete report. For the time being, this report can be found at: http://www.wus.cv.prod.osb.overheid.nl/ictu-ocvwus-war/viewWSIReport.html?id=25304

The report is divided into 4 sections. These sections handle different parts of the SOAP message. The sections are:

- discovery
- description
- message
- envelope

The OSB only concentrates on the last 2. Discovery and description are not tested and thus not part of the OSB Compliance test.

Message

The complete message has been checked against the following points. A short description is provided what exactly was tested. The checks that were not applicable were not checked and thus not explained here. All applicable tests passed successfully.

ID	Applicable?	Description
BP1001	YES	The request or response message should contain the protocol HTTP 1/1. Also the exact position of this protocol is checked.
BP1002	YES	The request or response message should contain the protocol HTTP 1/1 or HTTP 1/0. Also the exact position of this protocol is checked.
BP1004	YES	Checks if the message is a POST message.
BP1006	NO	
BP1010	NO	
BP1101	NO	
BP1103	NO	
BP1116	NO	
BP4103	NO	
BP4104	YES	The message contains an HTTP Header field that is not from the following list of specified header fields: (http://www.mnot.net/drafts/draft-nottingham-http-header-reg-00.txt)
BP4105	NO	
BP4106	NO	
BP4107	NO	
SSBP1003	YES	The logged SOAP envelope is a UTF-8 transcript of an envelope originally encoded as UTF-8 or UTF-16. The HTTP Content-Type header's charset value is either UTF-8 or UTF-16.
SSBP5100	YES	The SOAP envelope is the exclusive payload of the HTTP entity-body.
SSBP5101	YES	A message must have a "Content-Type" HTTP header field. The "Content-Type" HTTP header field must have a field-value whose media type is "text/xml".

Envelope

The SOAP envelope is checked against the following points. A short description is provided what exactly was tested. The checks that were not applicable were not checked and thus not explained here. All applicable tests passed successfully.

ID	Applicable?	Description
BP1005	NO	
BP1007	YES	DTDs relating to soap:header or soap:body documents, are not present in the envelope: no DOCTYPE element is present.
BP1008	NO	
BP1009	NO	
BP1011	NO	
BP1012	NO	
BP1013	NO	
BP1031	NO	
BP1032	YES	The soap:Envelope, soap:Header, and soap:Body elements do not have any attributes in the namespace "http://schemas.xmlsoap.org/soap/envelope/"
BP1033	YES	The SOAP envelope does not contain the namespace declaration xmlns:xml="http://www.w3.org/XML/1998/namespace".
BP1100	YES	The message uses a "200 OK" HTTP status code
BP1107	NO	
BP1201	YES	The envelope has a document element named "Envelope" with namespace http://schemas.xmlsoap.org/soap/envelope/.
BP1202	YES	Each child element (if any) of the soap:Body element is namespace qualified (not the grandchildren).
BP1203		
BP1204	NO	
BP1208	YES	The SOAP envelope does not include XML processing instructions.
BP1211	NO	
BP1212	NO	
BP1213	NO	
BP1214	NO	
BP1301	NO	
BP1302	NO	
BP1305	NO	
BP1306	NO	
BP1307	YES	The envelope contains elements that are namespaced "http://schemas.xmlsoap.org/soap/envelope/" do not have a soap:encodingStyle attribute.
BP1308	YES	No child element of soap:Body has a soap:encodingStyle attribute.
BP1309	YES	The soap:Envelope does not have direct children after the soap:Body element

BP1316	NO	
BP1318	NO	
BP1600	YES	The envelope conforms to the structure specified in SOAP 1.1 Section 4.
BP1601	YES	The soap:envelope and soap:body are well-formed XML 1.0 documents.
BP1701	YES	The envelope conforms to the SOAP schema located at http://schemas.xmlsoap.org/soap/envelope/
BP1755	NO	
BP4100	NO	
BP4101	NO	
BP4102	NO	
BP4109	NO	
OSB01	YES	wsa:To must represent an absolute IRI representing the address of the intended receiver of this message.
OSB02	YES	wsa:Action is required and must represent an absolute IRI that uniquely identifies the semantics implied by this message.
OSB03	YES	wsa:MessageId uniquely identifies this message.
OSB04	YES	A pair of values that indicate how this message relates to another message. The type of the relationship is identified by an absolute IRI. The related message is identified by an absolute IRI that corresponds to the related message's [message id] property.
OSB05	YES	Reference of the endpoint where the message originated from. The value of this element must be http://www.w3.org/2005/08/addressing/anonymous or http://www.w3.org/2005/08/addressing/none
OSB06	YES	Checks if there are user headers used next to the ws-addressing headers
SSBP1601	YES	The soap:envelope in the message is a well-formed XML 1.0 document.
SSBP9704	YES	The ENVELOPE does not contain the namespace declaration xmlns:xml="http://www.w3.org/XML/1998/namespace".