Open GIS Consortium Inc.

Date: 2003-01-15

Reference number of this OpenGIS® project document: OGC 03-014

Version: 0.8

Category: OpenGIS® Discussion Paper

Editors:

Jérôme Sonnet
Ionic Software s.a.

Charles Savage
GE Network Solutions

OWS 1.2 SOAP Experiment Report

Warning

This document is not an OGC Standard or Specification. This document presents a discussion of technology issues considered in an Interoperability Initiative of the OGC Interoperability Program. The content of this document is presented to create discussion in the geospatial information industry on this topic; the content of this document is not to be considered an adopted specification of any kind. This document does not represent the official position of the OGC nor of the OGC Technical Committee. It is subject to change without notice and may not be referred to as an OGC Standard or Specification. However, the discussions in this document could very well lead to the definition of an OGC Implementation Specification.

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

Document type:	OpenGIS® Discussion Paper
Document stage:	Publicly Available
Document language:	English

File name: 03-014.doc

Contents

i. Preface

ii. Submitting organizations

Ionic Software s.a.

GE Network Solutions

iii. Document Contributor Contact Points

All questions regarding this submission should be directed to the editor or the submitters:

CONTACT	COMPANY	ADDRESS	PHONE/FAX	EMAIL
Charles Savage	GE Network Solutions		303-464-2800	charlie1.savage@ps.ge.com
Jérôme Sonnet	Ionic Software		+32 4 364 0 364	Js@ionicsoft.com

iv. Revision history

Date	Release	Author	Paragraph modified	Description
18 November 2002	0.8	Charles Savage	Overall	Updated toolkit issues
14 November 2002	0.7	Charles Savage	Overall	Added section about issues encountered with Web Service toolkits. Also updated proposed document/literal SOAP binding
1 October 2002	0.6	Charles Savage	Overall	Changes in order to submit to the TC
16 August 2002	0.5	J. Sonnet	Overall	Additional editorial changes in order to submit to the TC
16 August 2002	0.4	J. Sonnet	Overall	Editorial changes in order to submit to the TC
5 August2002	0.3	J. Sonnet	10,11,12	Merge UML models and fix table of content

19 July 2002	0.2	J. Sonnet and Charles Savage	Third draft	Combined DIPR
19 July 2002	0.1	Charles Savage	Second draft	GE Network Solutions DIPR
17 July 2002	0.1	J. Sonnet	First draft	Ionic DIPR

v. Changes to the OpenGIS® Abstract Specification

The OpenGIS® Abstract Specification does not require changes to accommodate the technical contents of this document.

vi. Future Work

- Add WMS GetFeatureInfo operation
- Extend the UML model for SLD

Introduction

Over the last several years OGC's Interoperability Program has used a collaborative process to develop specifications for enabling the creation and interoperability of geospatially enabled services. These specifications include, but are not limited to:

- Web Map Server (WMS)
- Web Feature Server (WFS)
- Web Coverage Server (WCS)

Each of these specifications defines a number of services. Together these services are referred to as OGC Web Services (OWS). OGC Web Services allow geospatial information to be both accessed and processed over the Internet. Geospatial information is generally represented using XML that complies with the appropriate XML Schemas.

During the same period, the general computer industry has also come to the realization that there needs to be a standard way of describing, deploying, finding and calling software resources across the Internet. Collectively known as "Web Services," industry giants such as Microsoft, IBM and Sun have come to an agreement on a new set of standards in a remarkably short amount of time. Web Services offer a new platform for building loosely coupled, distributed systems.

The premise of the SOAP experiment is the belief that porting OWS services to Web Services will offer several key benefits, including:

- *Distribution* – It will be easier to distribute geospatial data and applications across platforms, operating systems, computer languages, etc.
- *Integration* – It will be easier for application developers to integrate geospatial functionality and data into their custom applications.
- *Infrastructure* - The GIS industry could take advantage of the huge amount of infrastructure that is being built to enable the Web Services architecture – including development tools, application servers, messaging protocols, security infrastructure, workflow definitions, etc.

This document will discuss how OWS services can be ported to Web Services and highlight various issues/problems that have been discovered and need further discussion.

1 Scope

The SOAP experiment had three primary goals:

- Use the Unified Modeling Language (UML) to define a platform independent model of Web Map Server interfaces
- Use the UML WMS model to define the appropriate XML Schema and Web Service Definition Language (WSDL) files that allow OWS services to be invoked using standard protocols such as HTTP GET, HTTP POST and SOAP
- Use standard COTS Web Services toolkits to invoke OGC Web Services across the Internet.

2 Conformance

Not required for an IP IPR, DIPR, or Discussion Paper.

3 Normative references

None.

4 Terms and definitions

For the purposes of this document, the following terms and definitions apply.

a) SOAP (Simple Object Access Protocol)
An XML-based remote procedure call protocol that allows messages to be exchanged between different services. SOAP is used for the bind operation described above.

b) WSDL (Web Services Description Language)
An XML-based vocabulary that provides a standard way of describing a web service's functionality. WSDL is used for the publish operation described above.

c) UDDI (Universal Description, Discovery, and Integration)
A common set of SOAP APIs that enable the publishing and finding of web services. UDDI registries provide the equivalent of a phone books white pages, yellow pages and green pages. UDDI is used to implement the service broker role and find operation described above.

d) RPC (Remote Procedure Call)
Remote Procedure Call is a protocol that one program can use to request a service from a program located in another computer in a network without having to understand network details. (A procedure call is also sometimes known as a function call or a subroutine call.)

e) Document Literal

The document literal approach consists in sending an XML document as request message and the service sends back another XML document as response message. There is no other constraint on the message encoding. (A Document Literal call is also known as a message call.)

5 Conventions

5.1 Symbols (and abbreviated terms)

None.

5.2 UML Notation

The diagrams that appear in this document are presented using the Unified Modeling Language (UML) static structure diagram. The UML notations used in this document are described in the diagram below.

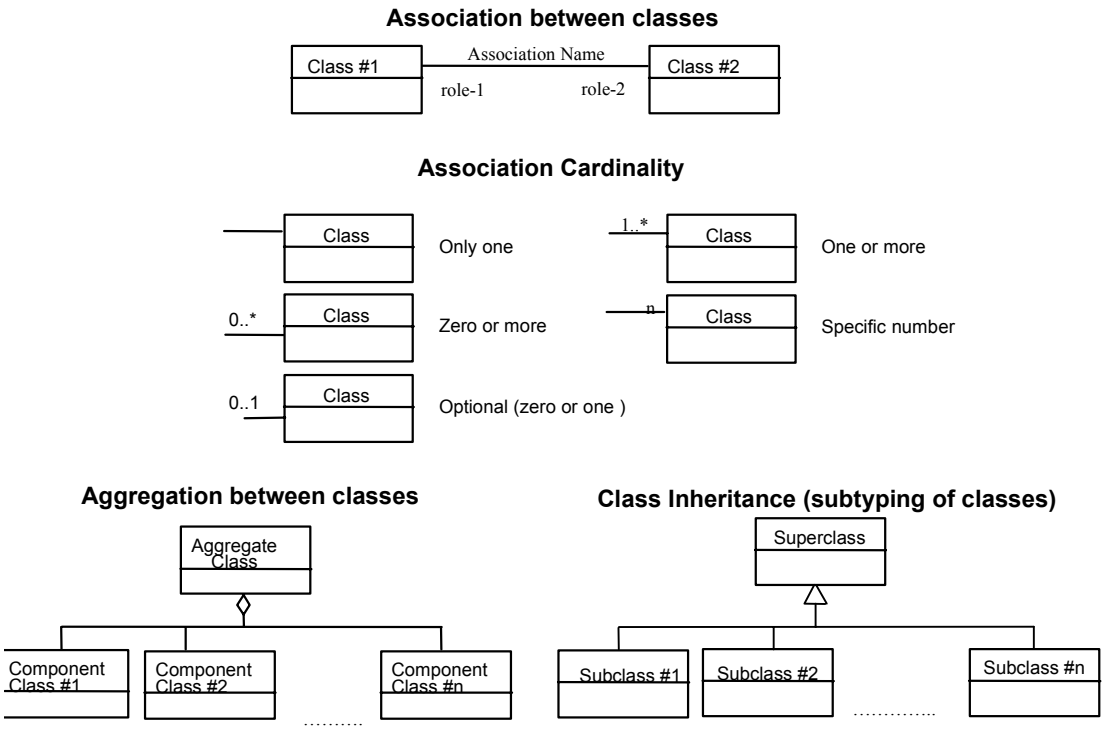


Figure 1: UML Notation

In this diagram, the following three stereotypes of UML classes are used:

a) <<Interface>> A definition of a set of operations that is supported by objects having this interface. An Interface class cannot contain any attributes.

b) <<DataType>> A descriptor of a set of values that lack identity (independent existence and the possibility of side effects). A DataType is a class with no operations whose primary purpose is to hold the information.

c) <<CodeList>> is a flexible enumeration that uses string values for expressing a list of potential values.

In this document, the following standard data types are used:

f) CharacterString – A sequence of characters

g) Integer – An integer number

h) Double – A double precision floating point number

i) Float – A single precision floating point number

6 Description of the experiment

The SOAP experiment had three primary goals:

- Use the Unified Modeling Language (UML) to define a platform independent model of Web Map Server interfaces
- Use the UML WMS model to define the appropriate XML Schema and Web Service Definition Language (WSDL) files that allow OWS services to be invoked using standard protocols such as HTTP GET, HTTP POST and SOAP
- Use standard COTS Web Services toolkits to invoke OGC Web Services across the Internet.

6.1 Toolkits

A variety of standard Web Service toolkits were tested to ensure maximum interoperability. These toolkits included:

- Visual Studio .NET (http://msdn.microsoft.com/vstudio)
- Apache Axis (http://xml.apache.org/axis/index.html)
- XML Spy (http://www.xmlspy.com/)

- Systinet (http://www.systinet.com)
- Soaplite (http://www.soaplite.com)

7 Web Map Server Overview

A Web Map Service produces maps of geo-referenced data, where a "map" is defined as a visual representation of geo-referenced data; a map is not the data itself. Maps are generally rendered in a pictorial format such as PNG, GIF or JPEG, although they can also be rendered in a vector-based format such as Scalable Vector Graphics (SVG) or Web Computer Graphics Metafile (WebCGM). The WMS does not impose any output formats, allowing each implementation to offer as many formats as is deemed necessary.

WMS defines four operations, which are:

- **GetCapabilities -** Returns service-level metadata, which is a description of the service's information content and acceptable request parameters.
- **GetMap -** Returns a map image whose geospatial and dimensional parameters are well defined.
- **GetFeatureInfo -** Returns information about particular features shown on a map. This operation is optional.
- **DescribeLayer** – Returns the feature types of the layer or layers specified in a request, and the attributes can be discovered with the **DescribeFeatureType** operation of a WFS interface. This operation is added by the Styled Layer Descriptor Specification and is optional.

The WMS specification defines two types of parameters:

- Global parameters
- Operation specific parameters

Global parameters are parameters that are common to multiple operations. These parameters include:

Name	Description
Version	Specifies the protocol version number.
Request	Indicates which service operation is being invoked.
Format	Specifies the output format of the response to an operation.

Exceptions	The format in which to report errors.
Srs	The Spatial Reference System (SRS) specifies the coordinate reference system code.
bounding_box	The Bounding Box (BBOX) is a set of four comma-separated decimal, scientific notation, or integer values (if integers are provided where floating point is needed, the decimal point is assumed at the end of the number).
Time	
Elevation	

7.1 GetMap Parameters

Name	**Description**
Layers	Comma-separated list of one or more map layers. Optional if SLD parameter is present
Styles	Comma-separated list of one rendering style per requested layer. Optional if SLD parameter is present.
Width	Width in pixels of map picture.
Height	Height in pixels of map picture.
Transparent	Background transparency of map (default=FALSE).
Bgcolor	Hexadecimal red-green-blue color value for the background color (default=0xFFFFFF).

7.2 GetFeatureInfo Parameters

Name	**Description**
query_layers	Comma-separated list of one or more layers to be queried.
info_format	Return format of feature information (MIME type).
feature_count	Number of features about which to return information (default=1).

X	X coordinate in pixels of feature (measured from upper left corner=0).
Y	Y coordinate in pixels of feature (measured from upper left corner=0).

7.3 GetCapabilities Parameters

Name	Description
Updatesequence	Sequence number or string for cache control

8 UML Overview

The Unified Modeling Language (UML) is a standard graphical representation for creating logical models that can represent applications, computer systems, business processes, business-to-business communications, etc. UML allows a model to be constructed, viewed, developed, and manipulated in a standard way at analysis and design time. By providing a graphical representation of a model, UML can greatly improve the communication about its design. It is significantly easier to understand a UML model then it is to understand particular implementations of the model.

UML is one of the key specifications underlying the Object Management Group's (OMG) Model Driven Architecture (MDA). The central premise of MDA is to separate the fundamental logic behind a specification from the specifics of any particular system that implements it. Specifications modeled in this way are much easier to understand than specifications that are implemented via the use of a particular technology. As a result, MDA makes it possible to rapidly develop and deliver new interoperability specifications that can then be deployed using the latest technologies. For example, a UML specification can be implemented in any number of technologies including:

- DTD schema
- XML schema
- Packages, classes, etc. in some language such as Java
- Database schema
- WSDL documents

As a result, one of the most important parts of this experiment was developing a UML model for WMS. This logical model was then mapped to various Web Service bindings, including:

- WSDL document based on rpc/encoded formatting
- WSDL document based on document/literal encoding
- XML Schema that can be used for HTTP Post
- KVP encoding currently used by the existing GET interface (if time allows)

Note that this usage of UML is already well established within the OpenGIS Consortium. For example, the GML standard makes frequent use of UML models to explain the design decisions that underlie the standard. Note that using UML in a specification process does not mean that the UML model will be the only normative part of the specification. The different mapping should also be normative to allow « on the wire » interoperability. However, UML offers a cross mapping integrity that provides real uniformity of the different mappings.

9 UML Models of Existing WMS Vocabularies

To achieve interoperability between WMS implementations it is critical to agree on the content of messages that are sent between the systems. Currently, two different vocabularies have been developed for WMS, including one for HTTP Get and one for HTTP Post. Unfortunately, neither of these vocabularies was found sufficient for defining a UML model of WMS due to various limitations that are discussed below.

9.1 Existing HTTP Get UML Model

The WMS 1.1.1 specification defines a number of parameters that should be used when calling a WMS service. These parameters are encoded using keyword/value pairs for operation requests using HTTP Get. The UML model for these parameters is shown in Figure 2 below.

Figure 2: UML model for WMS HTTP Get Binding

The main advantage of HTTP Get vocabulary is that a request is fully represented in single URL, which can easily be bookmarked or shared. However, the HTTP Get model has various shortcomings, including:

- It does not provide an easy way of passing structured parameter values to a server – for example additional semantics must be defined to associate a list of layers with a list of styles.
- Styled Layer Descriptor (SLD) WMS requests, which include an XML description of the styling, are difficult to encode directly and require that the request URL make reference to a separate SLD URL
- Proposals to add a Filter parameter have been made more complex by the need to associate specific Filter expressions with specific Layers
- Length limitations on HTTP URLs may preclude all of the desired parameters from being included in the request.

9.2 Existing HTTP Post UML Model

To overcome the limitations of the HTTP Get binding a HTTP Post binding was developed. This binding, and its associated schemas, is described in OpenGIS Project Document 02-017r1. The binding specifies that an HTTP Post message to a WMS server should consist of an xml message that conforms to schemas that were developed as part of the effort. Using xml messages solves the issues discussed above because:

- It allows additional structure in the request message, thereby by allowing additional functionality
- It removes size restrictions
- The comma-separated list of Layer names can be replaced by a sequence of XML elements, each of which is either a named or a user-defined Layer, and directly associating Style and Filter information within each Layer. The GetFeatureInfo operation, which includes most of a GetMap request, benefits in a similar way.

A UML model of the schemas defined in the HTTP Post vocabulary are shown in Figure 3 below.

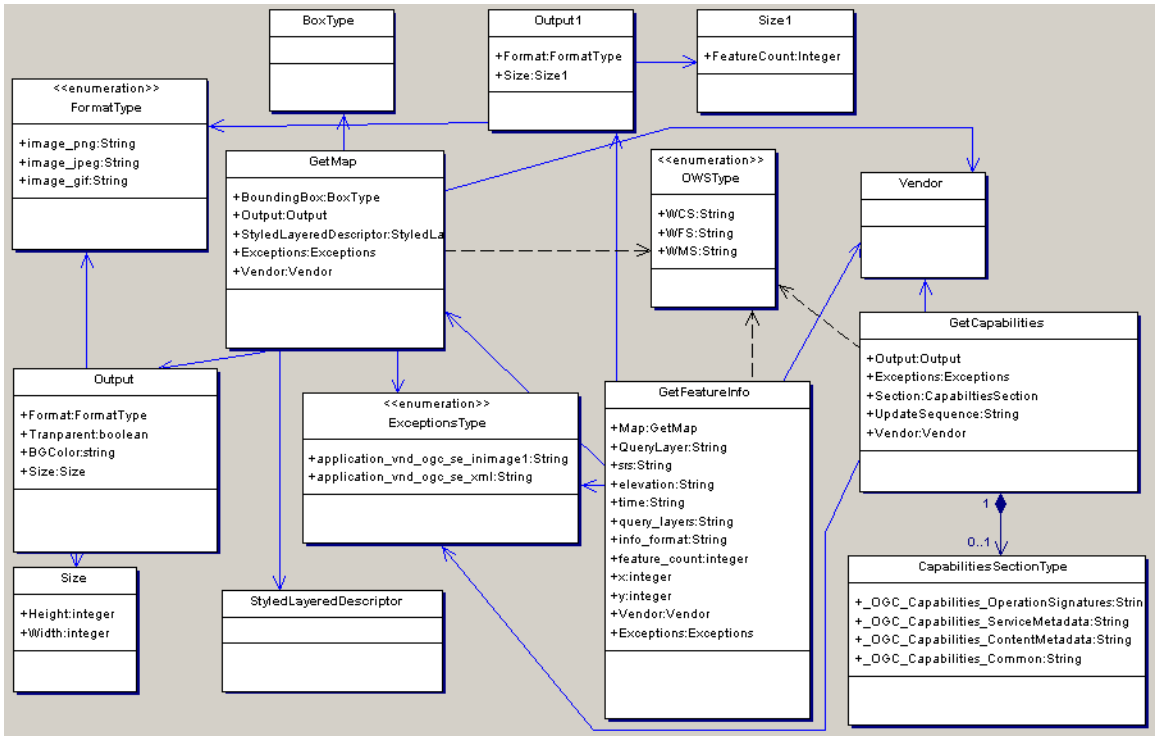


Figure 3: UML model for WMS HTTP Post Binding

Unfortunately, testing during this experiment revealed that the XML Schemas developed for the WMS HTTP Post bindings do not work correctly with existing Web Service toolkits. Issues that were encountered included:

- **version** and **service** are defined as attributes on the top-level element (GetMap, GetFeatureInfo, GetCapabilities). This is a problem because some Web Service toolkits, such as Axis, do not support the use of attributes. Note that .NET does support such attributes, but only through the use of its metadata functionality.

- the **exceptions** simple type defines an enumeration that includes two values, *application/vnd.ogc.se+inimage* and *application/vnd.ogc.se+xml*. Both Axis and .NET map these to constants defined on particular classes. However, these strings are not valid names in Java or C#. In the case of Axis, this will cause a compilation error while .NET mangles the name to make it valid. In either case, this is a critical issue.

In addition, testing also showed that the **GetFeatureInfo** and **GetMap** types are not cleanly separated. This causes problems because:

- The **exceptions** and **vendor** elements are repeated twice, once for **GetMap** and once for **GetFeatureInfo.**

- The **output** element is also repeated twice, but with different content.

- The **output** element in **GetMap** in not used for output when used in the context of **GetFeatureInfo**. The reason is that **GetFeatureInfo** requires the use of the WIDTH and HEIGHT parameters to perform its operation and thus these values must be set, although the operation is not generating maps.

10 Proposed WMS UML Model

Although the two existing UML models discussed above were found insufficient for use in the SOAP experiment, they did serve as the starting point for developing a new UML model of WMS. However, several major changes were made including:

- Separating global parameters, such as version and exceptions, into a separate **Request** class

- Addition of a **MapType** class, which defines the map area of interest. Thus it has attributes such as bounding box, srs elevation and height.

- Addition of an **ImageType** class, **w**hich defines how a map should be saved to a file. Thus it includes attributes such as image format, width, height, transparent and background color.

- The duplicated attributes between **GetFeatuteInfo** and **GetMap** have been removed.

In addition, great effort was put into reusing existing OpenGIS specifications. This included:

- Using the GML 2.1.2 **BoxType** class to specify the map bounds and spatial reference system (note that **BoxType** appears to be deprecated in GML 3.0 and thus should be replaced by the **EnvelopeType** in the future).
- Using the **StyledLayerDescriptor** class to specify how a map is styled.
- Using the **CapabilitesSectionTextType** class specified in the latest Capabilities schemas.

10.1 Data Types

The base data types of the new WMS UML model include:

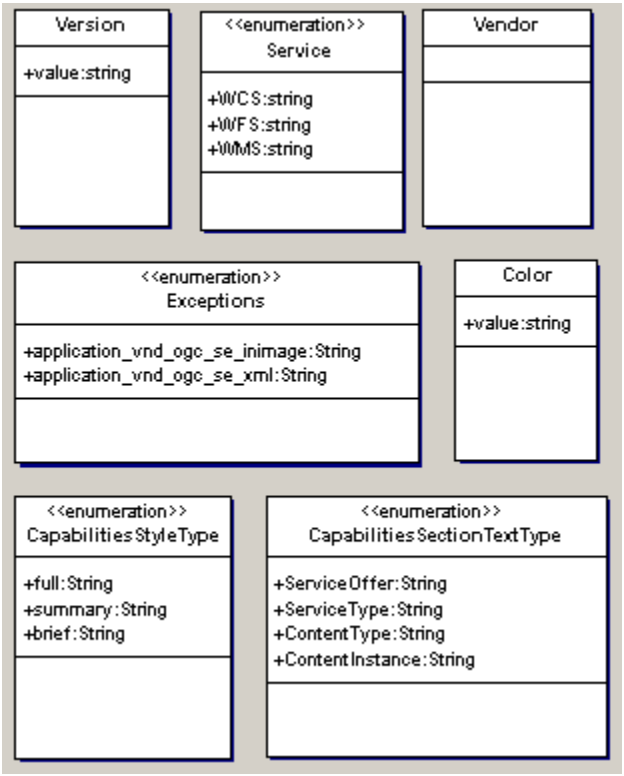


Figure 4: Base WMS types

10.2 Request Classes

The main WMS request classes are show in Figure 5 below.

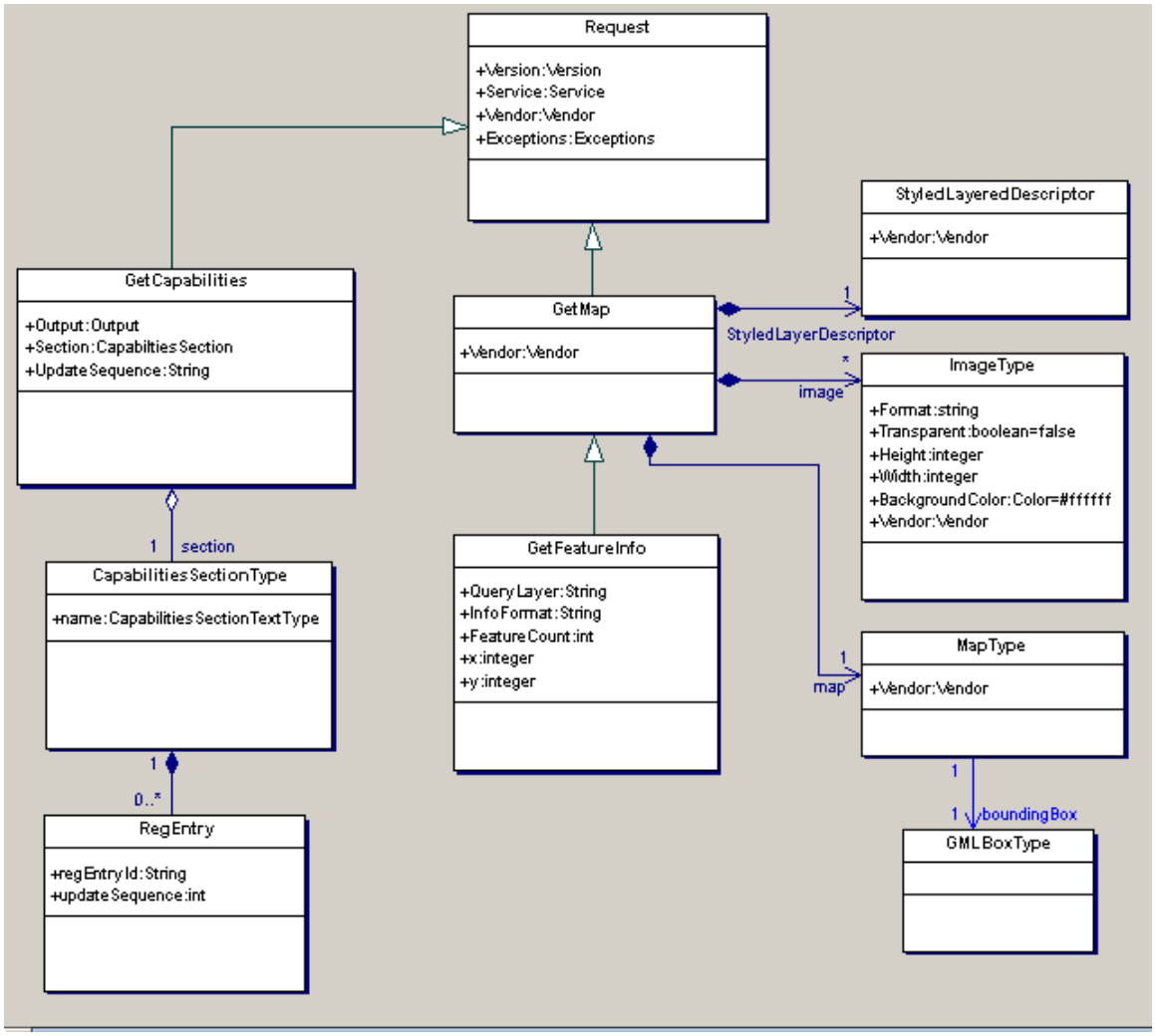


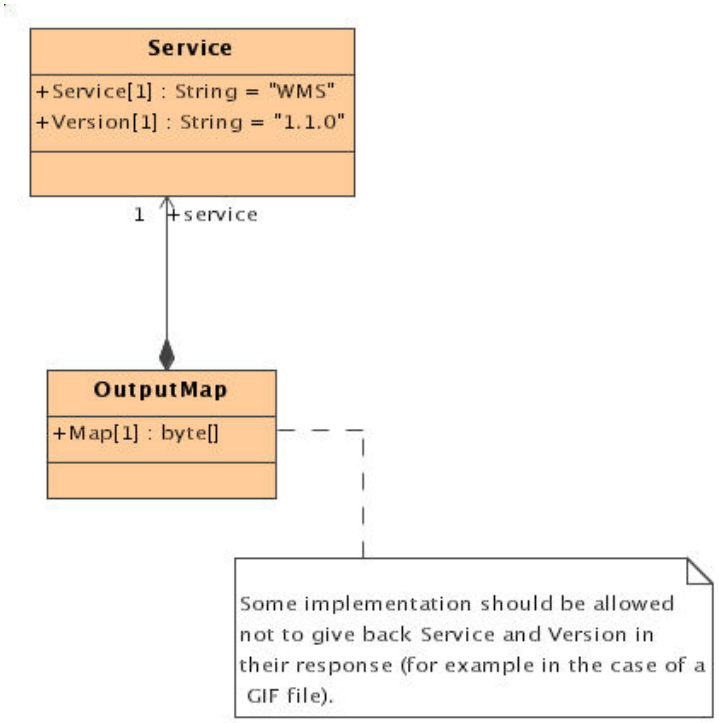
Figure 5: WMS UML Model

10.3 Response Classes

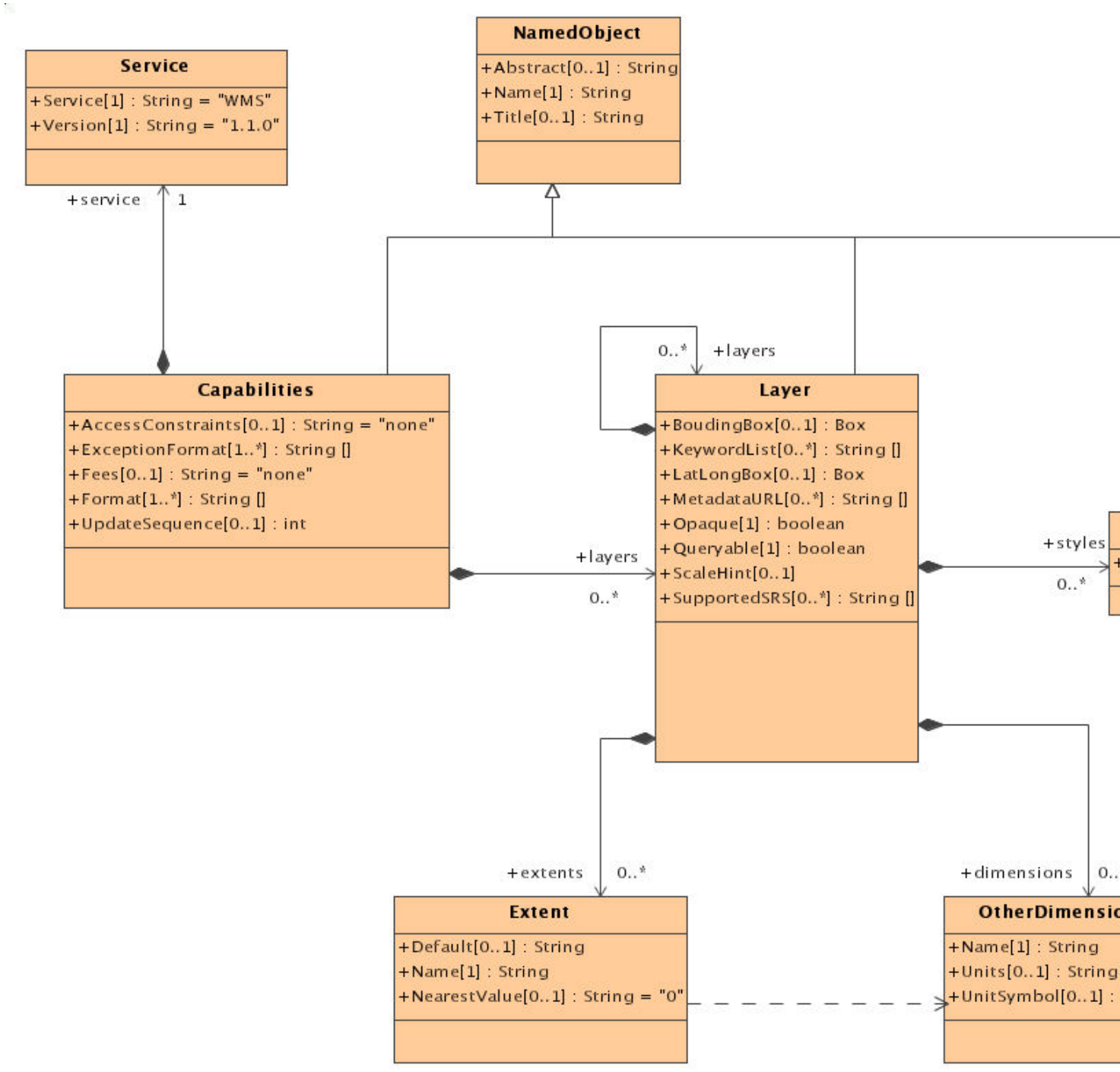
[Note that this section has not been updated yet]

Next, the main WMS response classes are show in Figure 5 below.

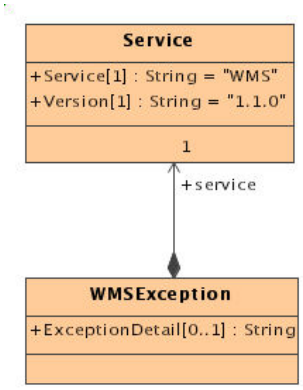
10.3.1 GetMapResponse



10.3.2 GetCapabilities Response



10.3.3 Exception handling



11 Mapping UML to Web Services

The next step in the experiment was to create new WMS Web Services based on the proposed UML model. This involved three steps, including:

- Creating the appropriate XML Schema documents from the UML WMS model
- Creating the appropriate Web Service Description Language (WSDL) documents that make use of the XML Schemas
- Defining the bindings that clients can use to send requests and responses to a server

Each of these steps will be discussed below.

11.1 Creating XML Schema Documents

There are many possible mappings between UML and XML, and a significant amount of research has taken place in this area. Of particular interest is OMG's work on version 2.0 of the XML Metadata Interchange (XMI) specification (see http://www.jeckle.de/files/XMI2.pdf), which includes standard mappings from UML/MOF models to XML Schemas.

However, XMI Version 2.0 had not been finalized at the time of this experiment. As a result, the UML was manually converted to XML Schema following the general rules laid out by David Carlson in his book "Modeling XML Applications with UML: Practical e-Business Applications (see http://www.amazon.com/exec/obidos/tg/detail/-/0201709155/qid=1037327601/sr=8-1/ref=sr_8_1/104-0318871-9800714?v=glance&s=books&n=507846).

11.2 Creating WSDL Documents

One of the key goals of the Web Services Experiment was to define the appropriate WSDL documents. A WSDL document consists of five main sections, including:

- Type definitions that are used to describe the data being exchanged. By default WSDL uses XML Schemas, although any schema system can be used.
- Message definitions that refer to the types defined in the type definitions section.
- Interface definitions (which are called portTypes). Each interface consists of multiple operations that have input, output and fault messages (which are defined in the message definitions section).
- Binding definitions that define how a specific interface can be accessed via either the HTTP Get, HTTP Post or SOAP protocols.
- Service definitions that consist of multiple ports. Each port is associated with a URL and a binding that define how a client can call a particular interface.

As part of the experiment two WSDL files were developed. The first WSDL file was developed by Ionic and is included in Appendix A. The second WSDL file was developed by GE Network Solutions and is included in Appendix B.

11.3 Bindings

WSDL bindings describe how an abstract service description should be mapped to a specific access protocol. Although most people equate Web Services only with SOAP, WSDL allows any number of different bindings. During the experiment the following bindings were investigated:

- HTTP Get
- HTTP Post
- SOAP RPC/Encoded
- SOAP Document/Literal

Although the focus of this experiment was on the SOAP bindings, it is deemed crucial to support the HTTP Get binding in any future WMS specifications that are developed. The reasoning for this recommendation is that the HTTP Get binding:

- Provides backwards compatibility with WMS 1.1.1 by mapping the UML model defined in Section 10 above to the appropriate name/value pairs

- Provides a mechanism for clients that do not support SOAP, such as older Web Browsers, to call a WMS server

12 SOAP Bindings

SOAP and WSDL provide a fair amount of flexibility in defining how SOAP messages should be formatted. This flexibility is controlled by how the overall SOAP document is constructed and how parameters are encoded. The table below shows the four possible combinations.

	Document-based SOAP messages	**RPC-based SOAP messages (According to SOAP Section 7)**
Literal Parameters (based on an XSD schema)	*Document/Literal*	RPC/Literal
Encoded Parameters (based on SOAP Section 5 encoding rules)	Document/Encoded	*Rpc/Encoded*

Document and *RPC* control how an operation call within a SOAP **Body** element is formatted, while *literal* and *encoded* control how individual parameters are encoded. These values are defined in the WSDL SOAP binding extension via the use of the **style** attribute and **use** attribute. Of these four combinations, the *Document/Literal* and *RPC/Encoded* styles are the most common, and therefore the most interoperable. As a result, the experiment focused on using both of these styles.

12.1 RPC / Encoded Binding

The concept behind RPC/Encoded style is to use messages to make remote procedure calls (RPCs) over the Internet to a specific operation on a service. In this model the operation is the important thing, not the message delivering the data.

In this style, the XML document that makes up the SOAP request follows the formatting rules outlined in sections 5 and 7 of the SOAP specification. This means that the structure of the documents that are sent on the wire is well defined and is not defined by the service. The service only defines the content of these messages.

Note that sections 5 and 7 were included in the SOAP specification ([2]) because SOAP had to create its own type system to guarantee web-based services, written in any

language, could communicate with one another. The use of XML Schema is only to define the types, and not necessarily the real encoding

SOAP specification section 5 assumes an object-centric view of the world –the underlying XML format being used to transfer messages is not important. Thus, the XML itself is given second-class status. So, the UML model that we develop during this experiment is actually defining the message content, the SOAP specification is responsible to define how to encode this in XML.

Note that WSDL specification recommends as well of using the SOAP data structure to specify the service definitions (see section 2.2 of [1]).

12.2 Document / Literal Binding

The concept behind Document/Literal formatting is passing messages between servers. In this model, the message is the most important thing, not the operation. The advantage of this style is that in theory it allows looser coupling. The disadvantage is that it presents a programming model that many developers are less familiar with.

In this style, the XML document that makes up the SOAP request follows a well-defined XML schema. This view assumes that XML Schemas is the dominant type system for all messages, not strongly typed objects defined in particular programming language. This XML-centric approach considers objects as mere implementation details and does not make use of SOAP section 5 or 7.

13 Clients

This experiment tested a broad range of Web Service toolkits to ensure maximum interoperability. These toolkits included:

- Visual Studio .NET (http://msdn.microsoft.com/vstudio)
- Apache Axis (http://xml.apache.org/axis/index.html)
- XML Spy (http://www.xmlspy.com/)
- Systinet (http://www.systinet.com)
- Soaplite (http://www.soaplite.com)

Each toolkit will be discussed below.

13.1 Visual Studio.NET

Visual Studio.NET includes a tool called *wsdl.exe* that generates client proxies in C# or Visual Basic from a WSDL document. The tool was successfully used to generate proxies for all bindings of interest.

13.2 Axis

Axis includes with a tool called *Wsdl2Java* that generates client proxies in Java from a WSDL document. The Axis tool currently only supports the SOAP bindings (both rpc/encoded and document/literal).

13.3 XML Spy

XML Spy, version 5, includes a SOAP debugging tool. The SOAP debugging tool was used to generate a SOAP request based on wms.wsdl.

XML Spy currently only supports SOAP bindings (both rpc/encoded and document/literal).

14 Issues

One of the major goals of the experiment was to identify issues involved with using Web Services standards and Web Service toolkits to call a WMS server. Issues that were identified during testing are summarized in Table 1 below.

Issue	Toolkit	Description
1	.NET	Does not support importing multiple XML Schema files that have the same target namespace.
2	Axis	Does not support the xs:ID, xs:NMTOKENS, xs:positiveInteger or xs:language types.
3	.NET and Axis	Do not support top-level elements in SOAP doc/lit messages to be derived by extension or restriction.
4	.NET and Axis	Do not support attributes on top-level elements in SOAP doc/lit messages.

		SOAP doc/lit messages.
5	Axis	Class name / namespace conflicts
6	Axis	Does not support enumerations.
7	Axis	Does not support enumerated values that contain special characters, such as "+"
8	.NET	Does not support complex types as parameters for HTTP Get and non-SOAP HTTP Post services.

Table 1: Issues identified during the SOAP Experiment

14.1 Issue #1

.NET does not support importing multiple XML Schema files that have the same namespace. This issue is easy to reproduce by following these steps:

1. Create a new WSDL file called Wsdl1.wsdl with the following content:

```
<definitions xmlns="http://schemas.xmlsoap.org/wsdl/"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:HTTP="http://schemas.xmlsoap.org/wsdl/HTTP/"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/"
xmlns:test="http://www.test.com"
xmlns:y="http://new.webservice.namespace"
targetNamespace="http://new.webservice.namespace">
   <import namespace="http://www.test.com" location="./schema1.xsd"/>
   <message name="testMessage">
      <part name="parameters" element="test:test"/>
   </message>
   <portType name="testPortType">
      <operation name="testOperation">
         <input message="y:testMessage"/>
      </operation>
   </portType>
   <binding name="testBinding" type="y:testPortType">
      <soap:binding style="document"
transport="http://schemas.xmlsoap.org/soap/HTTP"/>
      <operation name="testOperation">
         <soap:operation
soapAction="http://www.opengis.net/ows/testOperation"
style="document"/>
         <input>
            <soap:body use="literal"/>
         </input>
      </operation>
   </binding>
```

```
    <service name="testService">
      <port name="testPort" binding="y:testBinding">
        <soap:address location="http://localhost/test"/>
      </port>
    </service>
  </definitions>
```

2. Create a new XML Schema file called Schema1.xsd with the following content:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema targetNamespace="http://www.test.com"
xmlns:test="http://www.test.com"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified" attributeFormDefault="unqualified">
  <xs:import namespace="http://www.test.com"
schemaLocation="../directory2/schema2.xsd"/>
  <xs:element name="test" type="test:testType"/>
</xs:schema>
```

3. Create a new XML Schema file called Schema2.xsd with the following content:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema targetNamespace="http://www.test.com"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:test="http://www.test.com" elementFormDefault="qualified"
attributeFormDefault="unqualified">
  <xs:import namespace="http://www.test.com"
schemaLocation="schema3.xsd"/>
</xs:schema>
```

4. Create a new XML Schema file called Schema.3xsd with the following content:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema targetNamespace="http://www.test.com"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:test="http://www.test.com" elementFormDefault="qualified"
attributeFormDefault="unqualified">
  <xs:complexType name="testType"/>
</xs:schema>
```

5. Place the newly created files into the following directory structure:

- some_directory

  - schema_test

    - directory1

      wsdl1.xsd

schema1.xsd

- directory2

schema2.xsd

schema3.xsd

Attempting to import the schema files developed above using the wsdl.exe .NET command line tool will result in the following error:

```
Warning: Ignoring duplicate schema with
TargetNamespace='http://www.test.com' fr
om 'http://localhost/schemas/schema_test/directory1/schema1.xsd'.
Warning: Ignoring duplicate schema with
TargetNamespace='http://www.test.com' fr
om 'http://localhost/schemas/schema_test/directory2/schema2.xsd'.
Warning: Ignoring duplicate schema with
TargetNamespace='http://www.test.com' fr
om 'http://localhost/schemas/schema_test/directory1/schema1.xsd'.
Warning: Ignoring duplicate schema with
TargetNamespace='http://www.test.com' fr
om 'http://localhost/schemas/schema_test/directory2/schema2.xsd'.
Error: Unable to import binding 'testBinding' from namespace
'http://new.webserv
ice.namespace'.
  - Unable to import operation 'testOperation'.
  - The element 'http://www.test.com:test' is missing.
```

The only apparent workaround is to place all elements and types defined for a particular namespace into a single file. As a result, it is necessary to combine all the filter (expr.xsd and filter.xsd) schemas together and Style Layer Descriptor schemas (common.xsd, FeatureStyle.xsd, Symbol.xsd and StyledLayerDescriptor.xsd) .

14.2 Issue #2

Axis does not support the xs:ID, xs:NMTOKENS or xs:language types. As a result, the following changes to various schemas had to be made:

Schemas\gml\2.1.1\Geometry.xsd Changes			
Line	**15 Original**	**New**	**Comment**
64	<attribute name="fid" type="ID" use="optional"/>	<attribute name="fid" type="**string**" use="optional"/>	Axis RC1 does not support the xs:id type

83	<attribute name="fid" type="ID" use="optional"/>	<attribute name="fid" type="**string**" use="optional"/>	Axis RC1 does not support the xs:id type

Schemas\gml\2.1.1\Feature.xsd Changes			
Line	**16 Original**	**New**	**Comment**
51	<attribute name="gid" type="ID" use="optional"/>	<attribute name="gid" type=" **string** " use="optional"/>	Axis RC1 does not support the xs:id type
65	<attribute name="gid" type="ID" use="optional"/>	<attribute name="gid" type="**string**" use="optional"/>	Axis RC1 does not support the xs:id type
275	<attribute name="gid" type="ID" use="optional"/>	<attribute name="gid" type="ID" use="optional"/>	Axis RC1 does not support the xs:id type
292	<attribute name="gid" type="ID" use="optional"/>	<attribute name="gid" type=" **string** " use="optional"/>	Axis RC1 does not support the xs:id type
309	<attribute name="gid" type="ID" use="optional"/>	<attribute name="gid" type=" **string** " use="optional"/>	Axis RC1 does not support the xs:id type

Schemas\filter\0.1.20\filter.xsd Changes			
Line	**17 Original**	**New**	**Comment**
62	<xsd:attribute name="fid" type="xsd:ID" use="required"/>	<xsd:attribute name="fid" type=" **xsd:string** " use="required"/>	Axis RC1 does not support the xs:id type
67	<xsd:attribute name="rid" type="xsd:ID" use="required"/>	<xsd:attribute name="rid" type="**xsd:string**" use="required"/>	Axis RC1 does not support the xs:id type

17.1 Issue #3

Neither Axis or .NET support top level elements in SOAP doc/literal messages that are derived by extension or restriction. As a result, the following mapping from the proposed UML model to XML Schema does not work.

```
   <complexType name="RequestType">
      <element  name="version" type="ows:VersionType" default="1.1.1"/>
      <element name="service" type="ows:ServiceType" default="wms"/>
      <element name="exceptions" type="ows:ExceptionType"
default="application_vnd_ogc_se_xml"/>
   </complexType>

   <element name="GetMap" type="ows:GetMap"/>

   <complexType name="GetMap">
      <complexContent>
         <extension base="RequestType">
            <sequence>
               <element name="Map" type="ows:MapType"/>
               <element name="Image" type="ows:ImageType"/>
               <element ref="sld:StyledLayerDescriptor" minOccurs="0"/>
            </sequence>
            <attributeGroup ref="ows:RequestAttributeGroup"/>
         </extension>
      </complexContent>
   </complexType>
```

An alternative approach would be to use the XML Schemas group particle, as shown in the example below:

```
      <group name="RequestModelGroup">
         <sequence>
            <element name="version" type="ows:WmsVersionType"
   default="1.1.1"/>
            <element name="service" type="ows:ServiceType"
   default="wms"/>
            <element name="exceptions" type="ows:ExceptionType"
   default="application_vnd_ogc_se_xml"/>
         </sequence>
      </group>

      <complexType name="GetMap">
         <sequence>
            <group ref="ows:RequestModelGroup"/>
            <element name="Map" type="ows:MapType"/>
            <element name="Image" type="ows:ImageType"/>
            <element ref="sld:StyledLayerDescriptor" minOccurs="0"/>
         </sequence>
      </complexType>
```

Unfortunately, this approach works for .NET but not for Axis. Thus at the moment, the version, service and exceptions element are repeated for each top-level request element.

17.2 Issue #4

Neither Axis nor .NET support attributes on top-level elements in SOAP doc/literal messages. As a result, the following schema does not work:

```
    <element name="GetMap" type="ows:GetMap"/>
    <complexType name="GetMap">
      <complexContent>
        <extension base="RequestType">
          <sequence>
            <element name="Map" type="ows:MapType"/>
            <element name="Image" type="ows:ImageType"/>
            <element ref="sld:StyledLayerDescriptor" minOccurs="0"/>
          </sequence>
          <attributeGroup ref="ows:RequestAttributeGroup"/>
        </extension>
        <attribute name="version" type="ows:VersionType"
default="1.1.1"/>
        <attribute name="service" type="ows:ServiceType"
default="wms"/>
        <attribute name="exceptions" type="ows:ExceptionType"
default="application_vnd_ogc_se_xml"/>
      </complexContent>
    </complexType>
```

.NET simply ignores the attributes while Axis drops the GetMap element entirely and uses “non-wrapped” doc/literal messages. A “non-wrapped” SOAP doc/literal message for the GetMap operation looks like this:

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soapenv:Body>
    <Map xmlns="http://www.opengis.net/ows">
      <BoundingBox xsi:nil="true"/>
    </Map>
    <Image xmlns="http://www.opengis.net/ows">
      <Height>0</Height>
      <Width>0</Width>
      <Format xsi:nil="true"/>
    </Image>
  </soapenv:Body>
</soapenv:Envelope>
```

In contrast, a “wrapped” SOAP doc/literal message for GetMap looks like this:

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soapenv:Body>
    <GetMap xmlns="http://www.opengis.net/ows">
      <Map>
        <BoundingBox xsi:nil="true"/>
      </Map>
      <Image>
        <Height>0</Height>
        <Width>0</Width>
        <Format xsi:nil="true"/>
      </Image>
    </GetMap>
  </soapenv:Body>
</soapenv:Envelope>
```

Note the adddition of the **GetMap** element, which wraps the parameters passed to the actual **GetMap** operation. By default, .NET uses "wrapped" doc/literal messages. Axis, on the other hand, will use "wrapped" doc/literal messages when the following conditions are met:

- An input message has a single part
- The single part is an element
- The element has the same name as the operation
- The element's complex type has no attributes

Unfortunately, this approach works for .NET but not for Axis. Thus at the moment, the version, service and exceptions element are repeated for each top level request element.

17.3 Issue #5

Axis flattens namespaces, which results in some class name conflicts. For example, the various OpenGIS schemas used by wms.wsdl have the following namespaces:

- http://www.opengis.net/gml
- http://www.opengis.net/sld
- http://www.opengis.net/ows
- http://www.opengis.net/wms

Axis maps all of these XML namespaces to the same Java package www.opengis.net. As a result, a name conflict occurs between SLD and WMS because both define a **VersionType** complexType.

The workaround was to rename the WMS **VersionType** to **WmsVersionType**. Note that in the long run it probably makes sense to develop a set of shared ComplexTypes that can be used across all schemas (WMS, WCS, SLD, WFS, etc.).

17.4 Issue #6

Although Axis can import simpleTypes, it does not seem to support enumerated simpleTypes.

17.5 Issue #7

Axis does not support enumerated values that contain special characters, such as "+"

17.6 Issue #8

.Net does not support complex types as parameters for HTTP Get and non-SOAP HTTP Post services. Only SOAP-enabled services can utilize complex types as parameters. HTTP Get and non-SOAP HTTP Post services are limited to simple types, which are interpreted as key/value pairs, for request and response parameters.

This is due to the fact that there is not a standard way of submitting XML via HTTP Get or HTTP Post (assuming SOAP is not being used). This limitation likely applies to other Web Service Toolkits as well for this reason.

The effect of the limitation is that WSDL that describes a SOAP-enabled service can't generate a proxy for a non-SOAP version of the service, even if the services are otherwise identical. For Non-SOAP services to work with proxy generators, they must only use simple types as parameters.

18 Appendix A – Ionic Proposal

[Jerome, can you provide the final WSDL and Schema files you developed?]

19 Appendix B – GE Network Solutions Proposal

This Appendix includes GE Network Solution's proposed WMS XML Schema and WSDL documents. As recommended by Section 2.1.2 of the WSDL 1.1 specification, the WSDL document is divided into three files:

- wms_schema.xsd
- wms_definitions.wsdl
- wms.wsdl

This divides service definitions according to their level of abstraction. It also allows greater reuse of the WSDL documents and makes them easier to use and maintain.

19.1 wms_schema.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<schema targetNamespace="http://www.opengis.net/ows"
xmlns:sim="http://www.opengis.net/sim"
xmlns:sld="http://www.opengis.net/sld"
xmlns="http://www.w3.org/2001/XMLSchema"
xmlns:ows="http://www.opengis.net/ows"
xmlns:ogc="http://www.opengis.net/ogc"
xmlns:gml="http://www.opengis.net/gml"
elementFormDefault="qualified">
  <!--

****************************************************************
                                Imports
****************************************************************
    -->
  <!-- Note that we cannot import GML 3.0 Base because SLD imports
GML 2.1.1 already and thus we would end up with conflicts
  <import namespace="http://www.opengis.net/gml"
schemaLocation="../../gml/3.0/base/feature.xsd"/> -->
  <!-- Import Style Layer Descriptor.  Note that we use a custom
version, StyledLayerDescriptor_all, to get workaround
          .NET's problems with using xsd:include
    -->
  <import namespace="http://www.opengis.net/sld"
schemaLocation="../../sld/1.0.20/StyledLayerDescriptor_all.xsd"/>
  <!--

****************************************************************
                            Request Attribute Group
****************************************************************
    -->
  <group name="RequestModelGroup">
    <sequence>
```

```
        <element name="version" type="ows:WmsVersionType"
default="1.1.1"/>
        <element name="service" type="ows:ServiceType"
default="wms"/>
        <element name="exceptions" type="ows:ExceptionType"
default="application_vnd_ogc_se_xml"/>
      </sequence>
   </group>
   <!--

************************************************************
                                    Capabilities

************************************************************
      -->
   <element name="GetCapabilities">
      <complexType>
         <sequence>
            <!-- It would be preferrable to reuse the RequestModelGroup
here but
               Axis RC1 does not support groups. -->
            <element name="version" type="ows:WmsVersionType"
default="1.1.1"/>
            <element name="service" type="ows:ServiceType"
default="wms"/>
            <element name="exceptions" type="ows:ExceptionType"
default="application_vnd_ogc_se_xml"/>
            <!-- Now add additional elements -->
            <element name="section" type="ows:CapabilitiesSectionType"
minOccurs="0"/>
            <element name="style" type="ows:CapabilitiesStyleType"/>
         </sequence>
      </complexType>
   </element>
   <element name="GetCapabilitiesResponse">
      <complexType>
         <sequence>
            <!--  To be completed
                    <element ref="sim:OGC_Capabilities"/> -->
         </sequence>
      </complexType>
   </element>
   <!--  Old-style CapabilitiesSectionType with just identifiers-->
   <complexType name="CapabilitiesSectionType">
      <sequence minOccurs="0" maxOccurs="unbounded">
         <element name="RegEntry">
            <complexType>
               <attribute name="regEntryId" type="string"
use="required"/>
               <attribute name="updateSequence" type="int"
use="optional"/>
            </complexType>
         </element>
      </sequence>
      <attribute name="name" type="ows:CapabilitiesSectionTextType"
default="/"/>
```

```
   </complexType>
   <simpleType name="CapabilitiesSectionTextType">
      <restriction base="token">
         <enumeration value="/"/>
         <enumeration value="ServiceOffer"/>
         <enumeration value="ServiceType"/>
         <enumeration value="ContentType"/>
         <enumeration value="ContentInstance"/>
      </restriction>
   </simpleType>
   <simpleType name="CapabilitiesStyleType">
      <restriction base="token">
         <enumeration value="full"/>
         <enumeration value="summary"/>
         <enumeration value="brief"/>
      </restriction>
   </simpleType>
   <!--

*****************************************************************
                                  Map

*****************************************************************
      -->
   <complexType name="GetMap">
      <sequence>
         <!-- It would be preferrable to reuse the RequestModelGroup
here but
              Axis RC1 does not support groups. -->
         <element name="version" type="ows:WmsVersionType"
default="1.1.1"/>
         <element name="service" type="ows:ServiceType"
default="wms"/>
         <element name="exceptions" type="ows:ExceptionType"
default="application_vnd_ogc_se_xml"/>
         <!-- Now add additional elements -->
         <element name="Map" type="ows:MapType"/>
         <element name="Image" type="ows:ImageType"/>
         <element ref="sld:StyledLayerDescriptor" minOccurs="0"/>
      </sequence>
   </complexType>
   <element name="GetMap" type="ows:GetMap"/>
   <element name="GetMapResponse">
      <complexType>
         <sequence>
            <element name="GetMapResult" type="base64Binary"
minOccurs="0"/>
         </sequence>
      </complexType>
   </element>
   <complexType name="MapType">
      <sequence>
         <element name="BoundingBox" type="gml:CoordinatesType"/>
         <element name="Elevation" type="int" minOccurs="0"/>
         <element name="Time" type="string" minOccurs="0"/>
      </sequence>
```

```
   </complexType>
   <complexType name="ImageType">
      <sequence>
         <element name="Height" type="int"/>
         <element name="Width" type="int"/>
         <element name="Format" type="string"/>
         <element name="Transparent" type="boolean" default="false"
minOccurs="0"/>
         <element name="BGColor" type="ows:ColorType"
default="0xFFFFFF" minOccurs="0"/>
      </sequence>
   </complexType>
   <!--

************************************************************************
                                      FeatureInfo

************************************************************************
      -->
   <element name="GetFeatureInfo">
      <complexType>
         <annotation>
            <documentation>
           GetFeatureInfo should be extended from GetMap except that
.NET does not
           appear to support complexTypes derived by extension for the
            top level element of a SOAP request -->
        </documentation>
         </annotation>
         <sequence>
            <!-- It would be preferrable to reuse the RequestModelGroup
here but
               Axis RC1 does not support groups. -->
            <element name="version" type="ows:WmsVersionType"
default="1.1.1"/>
            <element name="service" type="ows:ServiceType"
default="wms"/>
            <element name="exceptions" type="ows:ExceptionType"
default="application_vnd_ogc_se_xml"/>
            <!-- It would be preferrable to extend GetMap but neither
.NET or Axis allow this
               for a top level item in a SOAP doc/literal request -->
            <element name="Map" type="ows:MapType" minOccurs="0"/>
            <element name="Image" type="ows:ImageType" minOccurs="0"/>
            <element ref="sld:StyledLayerDescriptor" minOccurs="0"/>
            <!-- Now add additional elements -->
            <element name="QueryLayer" type="string" minOccurs="0"/>
            <element name="InfoFormat" type="string" minOccurs="0"/>
            <element name="FeatureCount" type="int" minOccurs="0"/>
            <element name="x" type="int" minOccurs="0"/>
            <element name="y" type="int" minOccurs="0"/>
            <element name="Vendor" type="ows:Vendor" minOccurs="0"/>
         </sequence>
      </complexType>
   </element>
   <element name="GetFeatureInfoResponse">
```

```
      <complexType>
        <sequence>
          <element name="GetFeatureInfoResult" type="string"
minOccurs="0"/>
        </sequence>
      </complexType>
    </element>
    <!--

*********************************************************************
                                    BasicTypes

*********************************************************************
      -->
    <complexType name="Vendor">
      <sequence>
        <element name="test" type="string"/>
      </sequence>
    </complexType>
    <simpleType name="ExceptionType">
      <restriction base="token">
        <enumeration value="application_vnd_ogc_se_inimage"/>
        <enumeration value="application_vnd_ogc_se_xml"/>
        <enumeration value="application_vnd_ogc_se_blank"/>
      </restriction>
    </simpleType>
    <!-- ColorType -->
    <simpleType name="ColorType">
      <restriction base="token">
        <pattern value="0[xX][0-9a-fA-F]{6}"/>
      </restriction>
    </simpleType>
    <!-- WmsVersionType -->
    <simpleType name="WmsVersionType">
      <restriction base="token">
        <enumeration value="1.1.1"/>
      </restriction>
    </simpleType>
    <!-- ServiceType -->
    <simpleType name="ServiceType">
      <restriction base="token">
        <enumeration value="wms"/>
        <enumeration value="wfs"/>
        <enumeration value="wcs"/>
      </restriction>
    </simpleType>
    <!--

*********************************************************************
                                    Other

*********************************************************************
      -->
    <!-- These element are needed by .NET to generate HTTP Get binding
-->
    <element name="base64Binary" type="base64Binary" nillable="true"/>
```

```
    <element name="string" type="string" nillable="true"/>
  </schema>
```

19.2 wms_definitions.wsdl

```
  <definitions xmlns="http://schemas.xmlsoap.org/wsdl/"
  xmlns:wms_defs="http://www.opengis.net/wms_definitions"
  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
  xmlns:HTTP="http://schemas.xmlsoap.org/wsdl/HTTP/"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
  xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/"
  xmlns:ows="http://www.opengis.net/ows"
  targetNamespace="http://www.opengis.net/wms_definitions">
    <import namespace="http://www.opengis.net/ows"
  location="./wms_schemas.xsd"/>
    <message name="GetMapIn">
      <part name="parameters" element="ows:GetMap"/>
    </message>
    <message name="GetMapOut">
      <part name="parameters" element="ows:GetMapResponse"/>
    </message>
    <message name="GetFeatureInfoIn">
      <part name="parameters" element="ows:GetFeatureInfo"/>
    </message>
    <message name="GetFeatureInfoOut">
      <part name="parameters" element="ows:GetFeatureInfoResponse"/>
    </message>
    <message name="GetCapabilitiesIn">
      <part name="parameters" element="ows:GetCapabilities"/>
    </message>
    <message name="GetCapabilitiesOut">
      <part name="parameters" element="ows:GetCapabilitiesResponse"/>
    </message>
    <message name="GetMapHTTPGetIn">
      <part name="request" type="xs:string"/>
      <part name="version" type="ows:WmsVersionType"/>
      <part name="service" type="ows:ServiceType"/>
      <part name="exceptions" type="ows:ExceptionType"/>
      <part name="format" type="xs:string"/>
      <part name="srs" type="xs:string"/>
      <part name="bbox" type="xs:string"/>
      <part name="transparent" type="xs:boolean"/>
      <part name="height" type="xs:integer"/>
      <part name="width" type="xs:integer"/>
      <part name="bgcolor" type="ows:ColorType"/>
      <part name="layers" type="xs:string"/>
      <part name="styles" type="xs:string"/>
      <part name="time" type="xs:string"/>
      <part name="elevation" type="xs:integer"/>
    </message>
    <message name="GetMapHTTPGetOut">
      <part name="Body" element="ows:base64Binary"/>
    </message>
    <message name="GetCapabilitiesHTTPGetIn">
```

```
      <part name="request" type="xs:string"/>
      <part name="version" type="ows:WmsVersionType"/>
      <part name="service" type="ows:ServiceType"/>
      <part name="exceptions" type="ows:ExceptionType"/>
      <part name="Section" type="ows:CapabilitiesSectionType"/>
      <part name="updatesequence" type="xs:string"/>
   </message>
   <message name="GetCapabilitiesHTTPGetOut">
      <part name="Body" type="xs:string"/>
   </message>
   <portType name="WmsPortType">
      <operation name="GetMap">
         <input message="wms_defs:GetMapIn"/>
         <output message="wms_defs:GetMapOut"/>
      </operation>
      <operation name="GetFeatureInfo">
         <input message="wms_defs:GetFeatureInfoIn"/>
         <output message="wms_defs:GetFeatureInfoOut"/>
      </operation>
      <operation name="GetCapabilities">
         <input message="wms_defs:GetCapabilitiesIn"/>
         <output message="wms_defs:GetCapabilitiesOut"/>
      </operation>
   </portType>
   <portType name="WmsPortTypeHTTPGet">
      <operation name="GetMap">
         <input message="wms_defs:GetMapHTTPGetIn"/>
         <output message="wms_defs:GetMapHTTPGetOut"/>
      </operation>
      <!--          <operation name="GetFeatureInfo">
               <input message="ows:GetFeatureInfoHTTPGetIn"/>
               <output message="ows:GetFeatureInfoHTTPGetOut"/>
          </operation> -->
      <operation name="GetCapabilities">
         <input message="wms_defs:GetCapabilitiesHTTPGetIn"/>
         <output message="wms_defs:GetCapabilitiesHTTPGetOut"/>
      </operation>
   </portType>
   <!--

*****************************************************************
                              Import Types

*****************************************************************
     -->
   <!--

*********************************************************************
******
                             HTTP Post and SOAP (Doc/Lit) Messages

*********************************************************************
*****
     -->
   <!--
```

```
**********************************************************************
                                HTTP Get Messages
**********************************************************************
     -->
   <!-- <message name="GetFeatureInfoHTTPGetIn">
      <part name="version" type="xs:string"/>
      <part name="request" type="xs:string"/>
      <part name="format" type="xs:string"/>
      <part name="exceptions" type="xs:string"/>
      <part name="srs" type="xs:string"/>
      <part name="bbox" type="xs:string"/>
      <part name="time" type="xs:string"/>
      <part name="elevation" type="xs:string"/>
      <part name="transparent" type="xs:boolean"/>
      <part name="height" type="xs:integer"/>
      <part name="width" type="xs:integer"/>
      <part name="bgcolor" type="xs:string"/>
      <part name="querylayer" type="xs:string"/>
      <part name="infoformat" type="xs:string"/>
      <part name="featurecount" type="xs:integer"/>
      <part name="x" type="xs:integer"/>
      <part name="y" type="xs:integer"/>
   </message>
   <message name="GetFeatureInfoHTTPGetOut">
      <part name="Body" type="ows:string"/>
   </message> -->
   <!--

**********************************************************************
******
                                HTTP Post and SOAP Port

**********************************************************************
*****
      -->
   <!--

**********************************************************************
******
                                HTTP Get Port

**********************************************************************
*****
     -->
</definitions>
```

19.3 wms.wsdl

```
<definitions xmlns="http://schemas.xmlsoap.org/wsdl/"
xmlns:wms="http://www.opengis.net/wms"
xmlns:wms_defs="http://www.opengis.net/wms_definitions"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:HTTP="http://schemas.xmlsoap.org/wsdl/HTTP/"
```

```
xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/"
xmlns:ows="http://www.opengis.net/ows"
targetNamespace="http://www.opengis.net/wms">
  <import namespace="http://www.opengis.net/wms_definitions"
location="./wms_definitions.wsdl"/>
  <binding name="WmsServiceSoapBinding" type="wms_defs:WmsPortType">
    <soap:binding style="document"
transport="http://schemas.xmlsoap.org/soap/HTTP"/>
    <operation name="GetMap">
      <soap:operation
soapAction="http://www.opengis.net/ows/GetMap" style="document"/>
      <input>
        <soap:body use="literal"/>
      </input>
      <output>
        <soap:body use="literal"/>
      </output>
    </operation>
    <operation name="GetFeatureInfo">
      <soap:operation
soapAction="http://www.opengis.net/ows/GetFeatureInfo"
style="document"/>
      <input>
        <soap:body use="literal"/>
      </input>
      <output>
        <soap:body use="literal"/>
      </output>
    </operation>
    <operation name="GetCapabilities">
      <soap:operation
soapAction="http://www.opengis.net/ows/GetCapabilities"
style="document"/>
      <input>
        <soap:body use="literal"/>
      </input>
      <output>
        <soap:body use="literal"/>
      </output>
    </operation>
  </binding>
  <binding name="WmsServiceHTTPPostBinding"
type="wms_defs:WmsPortType">
    <HTTP:binding verb="POST"/>
    <operation name="GetMap">
      <HTTP:operation location="/GetMap"/>
      <input>
        <mime:content type="application/x-www-form-urlencoded"/>
      </input>
      <output>
        <mime:mimeXml part="Body"/>
      </output>
    </operation>
    <operation name="GetFeatureInfo">
```

```
        <HTTP:operation location="/GetFeatureInfo"/>
        <input>
          <mime:content type="application/x-www-form-urlencoded"/>
        </input>
        <output>
          <mime:mimeXml part="Body"/>
        </output>
      </operation>
      <operation name="GetCapabilities">
        <HTTP:operation location="/GetCapabilities"/>
        <input>
          <mime:content type="application/x-www-form-urlencoded"/>
        </input>
        <output>
          <mime:mimeXml part="Body"/>
        </output>
      </operation>
    </binding>
    <binding name="WmsServiceHTTPGetBinding"
type="wms_defs:WmsPortTypeHTTPGet">
      <HTTP:binding verb="GET"/>
      <operation name="GetMap">
        <HTTP:operation location="/GetMap"/>
        <input>
          <HTTP:urlEncoded/>
        </input>
        <output>
          <mime:mimeXml part="Body"/>
        </output>
      </operation>
      <!-- <operation name="GetFeatureInfo">
        <HTTP:operation location="/GetFeatureInfo"/>
        <input>
          <HTTP:urlEncoded/>
        </input>
        <output>
          <mime:mimeXml part="Body"/>
        </output>
      </operation> -->
      <operation name="GetCapabilities">
        <HTTP:operation location="/GetCapabilities"/>
        <input>
          <HTTP:urlEncoded/>
        </input>
        <output>
          <mime:mimeXml part="Body"/>
        </output>
      </operation>
    </binding>
    <service name="WmsService">
      <!-- SOAP binding -->
      <port name="WmsServiceSoap" binding="wms:WmsServiceSoapBinding">
        <soap:address location="http://localhost/sias/cgi-
bin/siscgi.exe/wms/soap"/>
      </port>
      <!-- HTTP Post binding -->
```

```
      <port name="WmsServiceHTTPPost"
binding="wms:WmsServiceHTTPPostBinding">
         <HTTP:address location="http://localhost/sias/cgi-
bin/siscgi.exe/wms/post"/>
      </port>
      <!-- HTTP Get binding -->
      <port name="WmsServiceHTTPGet"
binding="wms:WmsServiceHTTPGetBinding">
         <HTTP:address location="http://localhost/sias/cgi-
bin/siscgi.exe/wms"/>
      </port>
   </service>
</definitions>
```

Bibliography

[1] WSDL 1.1 Specification, W3C, 15 March 2001

http://www.w3.org/TR/wsdl

[2] SOAP 1.1 Specification, W3C, 08 May 2000

http://www.w3.org/TR/soap

[3] Microsoft presentation of Web Services

http://msdn.microsoft.com/webservices/understanding/readme/default.asp

[4] Modeling XML Applications With UML. David Carlson. February 2002.

http://www.sdmagazine.com/documents/s=2398/sdm0202f/0202f.htm.