

Open Geospatial Consortium Inc.

Date: 2004-08-20

Reference number of this OGC Project document: **OGC 04-060r1**

Version: 1.0.0

Category: OGC Discussion Paper

Editor: Jérôme Sonnet

OWS 2 Common Architecture: WSDL SOAP UDDI

Copyright notice

Copyright © Open Geospatial Consortium, Inc. (2005)

Warning

This document is not an OGC Specification or Standard. It is distributed for review and comment. It is subject to change without notice and may not be referred to as an OGC Standard.

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: OGC Discussion Paper
Document stage: Final
Document language: English

Contents

i.	Preface.....	i
ii.	Submitting organizations	i
iii.	Document Contributor Contact Points.....	i
iv.	Revision history.....	i
v.	Changes to the OGC Abstract Specification	ii
vi.	Future Work.....	ii
	Foreword.....	iv
	Introduction.....	v
1	Scope.....	1
2	Conformance	1
3	Normative references.....	1
4	Terms and definitions	1
4.1	SOAP	1
4.2	UDDI	2
4.3	WSDL.....	2
5	Conventions	2
5.1	Symbols (and abbreviated terms).....	2
5.2	UML Notation	3
6	WSDL.....	4
6.1	Introduction.....	4
6.1.1	About WSDL	4
6.1.2	WSDL versions.....	5
6.1.3	Abstract and implementation parts	5
6.1.4	Bindings	5
6.1.5	WS-I Profile.....	6
6.2	WSDL as a normative part to OGC implementation specification.....	6
6.3	WSDL issues	6
6.3.1	Complex types in HTTP bindings	6
6.3.2	WCS and WMS Specification problem.....	7
6.3.3	Unified WSDL for KVP and XML.....	7
6.3.4	Conclusion	7
7	WSDL for existing OGC Services	8
7.1	KVP bindings	8
7.1.1	Using existing schemas	8

7.1.2	WSDL for WCS 1.0.0.....	9
7.1.3	WSDL for WFS 1.0.0.....	10
7.1.4	WSDL for WMS 1.1.1.....	11
7.1.5	WSDL for CS-W 2.0.....	11
7.1.6	WSDL for Common Implementation Specification 1.0.0.....	11
7.1.7	WSDL Interoperability Experiments.....	11
7.2	XML & SOAP Bindings.....	12
7.3	Issues with SOAP.....	13
7.3.1	SOAP document/literal as the default OGC choice.....	13
7.3.2	SOAP for large binary transport.....	13
8	UDDI.....	13
8.1	Registration of OGC Web Services into UDDI.....	14
8.1.1	The UDDI data model.....	14
8.1.2	WSDL and UDDI registries.....	14
8.2	Two ways assertion.....	15
8.2.1	Use case.....	15
8.2.2	Implementation 1:Illustrate Scenario with no custom UDDI objects.....	15
8.2.3	Observations and Conclusions.....	18
8.3	Developing a custom validating Taxonomy.....	19
8.3.1	Technical aspects.....	19
8.3.2	Service registration with validation with user defined checked taxonomy....	23
8.3.3	Conclusions and observations.....	29
9	MapInfo - TIEs Executive Summary.....	29
10	MapInfo - Report on WSDL WFS.....	30
10.1	Problems.....	30
10.1.1	Schema Definitions.....	30
10.1.2	Problem.....	30
10.1.3	Solution.....	30
10.2	Members of Type Object.....	30
10.2.1	Problem.....	30
10.2.2	Solution.....	30
10.3	Multiple Occurrence of Elements of the Same Type.....	31
10.3.1	Problem.....	31
10.3.2	Solution.....	32
10.4	Schemas.....	32
11	MapInfo - Report on WSDL WFS.....	32
11.1	Overview.....	32
11.2	Steps.....	33
11.3	Tools.....	33
11.4	Notes.....	33
11.5	Java Use of WFS.....	33
11.5.1	WSDL Validation.....	33
11.5.2	Stub Generation.....	36
11.5.3	Using the Stub.....	36

11.5.4	Conclusions.....	36
11.6	.Net Use of WFS	36
11.6.1	WSDL Validation.....	36
11.6.2	Stub Generation	41
11.6.3	Using the Stub	41
11.6.4	Conclusions.....	41
12	MapInfo - Report on WSDL WMS	41
12.1	Overview	41
12.2	Steps	42
12.3	Tools	42
12.4	Notes	42
12.5	Java Use of WMS WSDL	42
12.5.1	WSDL Validation.....	42
12.5.2	Stub Generation	44
12.5.3	Using the Stubs.....	47
12.5.4	Conclusions.....	47
12.6	.Net Use of WMS WSDL	48
12.6.1	WSDL Validation.....	48
12.6.2	Stub Generation	50
12.6.3	Using the Stub	52
12.6.4	Conclusions.....	52
13	Galdos Simple WSDL/SOAP Experiment for WFS	53
13.1	Experiment Description.....	53
13.2	Experiment Results.....	54
14	Ionic Experiment Summary.....	63
14.1	Overview	63
14.2	Current implementations of the specifications.....	63
14.2.1	Tools	63
14.2.2	Tools compliance.....	64
14.2.3	Reference implementations	64
14.3	Test Results.....	64
15	Intergraph - WSDL/BPEL TIEs.....	65

i. Preface

This document presents the result of the OWS 2 Common Architecture thread. This group has focused on adding WSDL/SOAP/UDDI support to existing OGC Web Services.

ii. Submitting organizations

The following organizations submitted this document to the Open Geospatial Consortium Inc.

IONIC Software s.a.

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
Jerome Sonnet	IONIC Software		+32 4 364 0 364	js@ionicsoft.com
Philippe Duchesne	IONIC Software		+32 4 364 0 364	phd@ionicsoft.com
Michael Abate	MapInfo			
Will Wilbrink	MapInfo			
Ron Lake	Galdos			
Louis Reich	CSC (Nasa)			lreich@csc.com

iv. Revision history

Date	Release	Author	Paragraph modified	Description
2004-03-23	0.0.1	Jerome Sonnet	First draft	Setup a TOC

2004-04-12	0.0.2	Jerome Sonnet		Add WCS experiment output
2004-05-26	0.0.3	Philippe Duchesne		Refine WSDL chapter
2004-07-15	0.0.4	Philippe Duchesne		Add WSDL introduction and concepts
2004-08-05	0.0.5	Jerome Sonnet		Add UDDI Description
2004-08-20	0.0.6	Jerome Sonnet		Integrate other participants contribution
2007-10-01	0.0.7	Philippe Duchesne		describe SOAP issues add UDDI registration description

v. Changes to the OGC Abstract Specification

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

vi. Future Work

Future work may include but is not limited to:

- Improve the use of technologies related to WSDL/SOAP such as BPEL to create service chaining using OGC services. This has been experimented by the IH4DS thread, but experimentation can really go further.
 - Business Process Execution Language for Web Services Version 1.1
<http://www-128.ibm.com/developerworks/library/ws-bpel/>
- Use other technologies related to SOAP such as messages routing, encryption, signature...
 - SOAP Security Extensions: Digital Signature
<http://www.w3.org/TR/SOAP-dsig/>
 - XML Encryption Syntax and Processing
<http://www.w3.org/TR/xmlenc-core/>

Foreword

Attention is drawn to the possibility that some of the elements of this part of OGC 04-060r1 may be the subject of patent rights. The Open Geospatial Consortium Inc. shall not be held responsible for identifying any or all such patent rights.

This work is an Interoperability report computed by the OGC Web Services 2 Common Architecture thread.

This document presents the work done and the issue encountered during the creation of the different change proposals that add support for UDDI/WSDL/SOAP to the WMS, WFS, WCS and CS-W specifications.

This is not a normative document.

Introduction

This document reports the activity of the Common Architecture thread of the OGC Web Service 2 Testbed. The focus of the Common Architecture tread has been to add support for WSDL/SOAP/UDDI to OGC baseline services and to test these implementations using COTS development tools.

The majority of this document describes implementation issues and problems the group encountered while defining the usage of these technologies in conjunction with OGC web service interfaces. The appendices in this document describe lessons learned using WSDL and SOAP definitions with various COTS development environments.

Best use of this document requires that the reader have at least a basic knowledge of WSDL/SOAP/UDDI.

Introductory element — Main element

1 Scope

This OGC document reports the work that occurred in the OWS2 Test Bed Common Architecture thread. This thread focused on the use of UDDI/WSDL/SOAP in the OGC Web Services architecture. It also provides guidelines for the use of these technologies.

2 Conformance

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

3 Normative references

OGC 02-058, *Web Feature Service (WFS-1.0.0)*.

OGC 02-059, *Filter Encoding (Filter-1.0.0)*.

OGC 01-068r3, *Web Map Service (WMS-1.1.1)*.

OGC 02-024, *Web Coverage Service (WCS-1.0.0)*.

OGC 04-021r2, *Catalog Interface (CAT-2.0.0)*.

OGC 03-029, *Messaging Framework (DP)*.

<http://www.w3.org/TR/wsdl>, *Web Services Description Language (WSDL) 1.1*.

<http://www.w3.org/TR/soap>, *SOAP Version 1.2*.

http://uddi.org/pubs/uddi_v3.htm, *UDDI Version 3.0.1*.

4 Terms and definitions

4.1 SOAP

SOAP is a lightweight protocol for exchange of information in a decentralized, distributed environment. It is an XML based protocol that consists of three parts: an

envelope that defines a framework for describing what is in a message and how to process it, a set of encoding rules for expressing instances of application-defined datatypes, and a convention for representing remote procedure calls and responses. SOAP can potentially be used in combination with a variety of other protocols; however, the only bindings defined in this document describe how to use SOAP in combination with HTTP and HTTP Extension Framework.

4.2 UDDI

Universal Description, Discovery and Integration, or UDDI, is the name of a group of web-based registries that expose information about a business or other entity[2] and its technical interfaces (or API's). These registries are run by multiple Operator Sites, and can be used by anyone who wants to make information available about one or more businesses or entities, as well as anyone that wants to find that information.

4.3 WSDL

WSDL is an XML format for describing network services as a set of endpoints operating on messages containing either document-oriented or procedure-oriented information. The operations and messages are described abstractly, and then bound to a concrete network protocol and message format to define an endpoint. Related concrete endpoints are combined into abstract endpoints (services). WSDL is extensible to allow description of endpoints and their messages regardless of what message formats or network protocols are used to communicate, however, the only bindings described in this document describe how to use WSDL in conjunction with SOAP 1.1, HTTP GET/POST, and MIME.

5 Conventions

5.1 Symbols (and abbreviated terms)

ISO	International Organization for Standardization
KVP	Key-Value Pair
OGC	Open Geospatial Consortium
SOAP	Simple Object Access Protocol
UDDI	Universal Description, Discovery and Integration
WCS	Web Coverage Service
WFS	Web Feature Service
WMS	Web Map Service

WSDL Web Service Description Language

XML eXtended Markup Language

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.

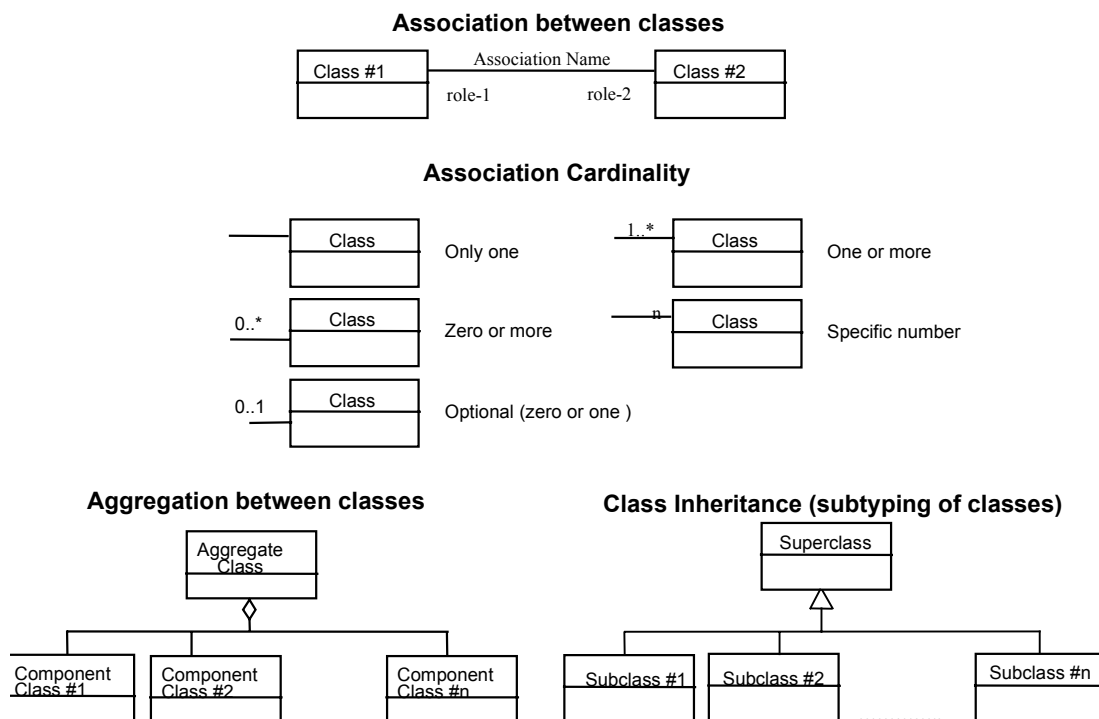


Figure 1 — UML notation

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:

- a) `CharacterString` – A sequence of characters
- b) `Integer` – An integer number
- c) `Double` – A double precision floating point number
- d) `Float` – A single precision floating point number

6 WSDL

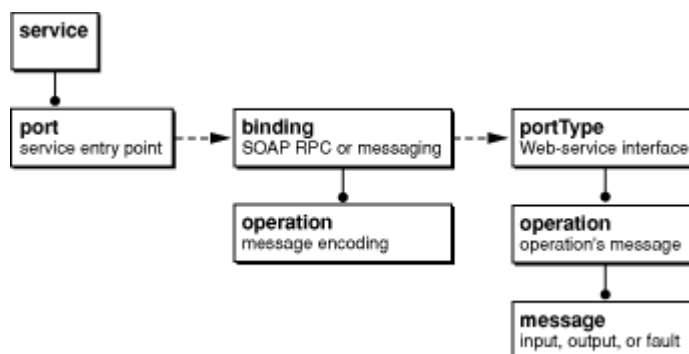
6.1 Introduction

OWS 1.1 pointed out the need for an Interface Definition Language to describe OGC services; WSDL was chosen as an instance of such a language.

6.1.1 About WSDL

The Web Services Description Language (WSDL) is an XML-based language used to describe the services a business offers and to provide a way for individuals and other businesses to access those services electronically. The WSDL document specification helps improve interoperability between applications, regardless of the protocol or the encoding scheme.

WSDL picks up where XML Schema left off by providing a way to group messages into operations and operations into interfaces (port types). It also provides a way to define bindings for each interface and protocol combination along with the endpoint address for each one. A complete WSDL definition contains all of the information necessary to invoke a Web service.



WSDL definitions make it possible to generate code that implements the given interface, on either the client or the server, making Web services accessible to the masses. Therefore, service owners that want to make it easy for others to access their services should make WSDL definitions available.

WSDL plays an important role in the overall Web services architecture since it describes the complete contract for application communication.

6.1.2 WSDL versions

WSDL 1.1 (<http://www.w3.org/TR/wsdl>) is the version currently most used and supported by existing tools; WSDL 1.2 (renamed 2.0 because of its substantial differences from 1.1) is at the moment in draft and not widely supported yet.

The work and conclusions described in this report are in respect of version 1.1.

6.1.3 Abstract and implementation parts

A WSDL document defines services as a collection of endpoints, but separates the abstract definition (the *interface*) from the concrete *implementation* :

- message, operation and portType elements provide abstract definitions for the data being exchanged and the operations being performed by a service complying with the WSDL document;
- A set of bindings is provided, each binding defining how to map the interface port types to a concrete set of ports, usually by providing encoding and connection parameters specific to an instance of the service.

KVP	XML
<ul style="list-style-type: none"> ▪ HTTP GET ▪ HTTP POST, URL-encoded 	<ul style="list-style-type: none"> ▪ HTTP POST, XML-encoded ▪ SOAP

6.1.4 Bindings

The implementation part of WSDL defines the bindings, which basically specify the encoding to use with a particular instance of a service. The most popular binding is SOAP, but other bindings include HTTP GET or POST.

Those bindings can be divided into Key-Value Pairs (KVP) or XML bindings.

As described in this diagram, the KVP messages are used in the case of HTTP GET and URL-encoded POST. The XML messages are used in the POST using XML document and in the SOAP bindings.

6.1.5 WS-I Profile

The Web Services Interoperability Organization (WS-I) is an open industry effort chartered to promote Web services interoperability across platforms, applications, and programming languages. The organization brings together several Web services leaders to respond to provide guidance, recommended practices, and supporting resources for developing interoperable Web services.

The WS-I Basic Profile provides interoperability guidance for core Web services specifications such as SOAP, WSDL, and UDDI.

OWS2 takes WS-I recommendations into consideration in the schemas design, but does not consider WS-I compliancy as a must-have. In short, we did followed all the recommendations, but it forbids the use of GET and POST binding which is not something we can afford in an OGC context.

6.2 WSDL as a normative part to OGC implementation specification

The abstract part of the WSDL descriptions will be considered as a mandatory, normative part in the OGC service implementation specifications

However, it is not required for a service to publish such a description. But if one is published, it must conform to the one of the OGC.

We probably need to work on the definition of the conformance of two WSDL description to allow integration in CITE testing

6.3 WSDL issues

6.3.1 Complex types in HTTP bindings

This dichotomy in the encoding unfortunately affects the abstract part of WSDL, because WSDL definitions using KVP bindings (i.e. HTTP GET and POST bindings) do not allow messages with complex types for message parts. Instead, each HTTP parameter must be matched by a SimpleType part on its own, thereby preventing the use of message types that do not have a flat structure.

Although some tools currently support KVP encoding on complex types, such support varies greatly from one tool to another, and is not specified in any way in the WSDL specification. Therefore using KVP with complex types is not a viable solution to consider.

Given that, it is not possible to define a unique abstract part that is independent of the implementation for HTTP Bindings.

6.3.2 WCS and WMS Specification problem

The WCS GetCoverage and the WMS GetMap (with Dimensions) cannot be fully mapped into a KVP definition. The reason is that these operations introduce arbitrary parameter names (e.g. BAND=...), and KVP definitions in WSDL must be exhaustive regarding the parameters used.

The solution is to change this organisation of the KVP encoding as

PARAMETERNAME = Comma separated list of names

PARAMETERVALUE = Comma separated list of values

Example :

```
&GroundTemperature=0/10
&AirTemperature=-5/0,5/10
```

would become something like

```
&AXISNAMES=GroundTemperature,AirTemperature
&AXISVALUES=0/10,(-5/0,5/10)
```

Two actions has been taken to resolve this issues :

1. The WCS and WMS RWG have been requested to change this encoding to allow a WSDL description of their interface.
2. The Common implementation WG has been requested to include a quote saying dynamic parameter is forbidden in OGC KVP request encoding.

6.3.3 Unified WSDL for KVP and XML

Current state of Web Services stack as defined by the W3C does not provide the required technical solution to meet this requirement. Some implementation allows specific encoding of mapping, but we have chosen not to use these “vendor specific” extensions in the OGC specifications.

This section is made available to those how may want to describe the way this issue may be solved in the future (Stephane?).

6.3.4 Conclusion

XML bindings (esp. SOAP) seem to be the way to go to; they should be used where possible and taken into account when defining new services; they offer all the flexibility needed in terms of message complexity and permit to leverage the XML schema definitions of the existing services.

However, KVP bindings cannot be overlooked, as many existing services support only such bindings, and we cannot expect all those services to be upgraded to SOAP.

Since a unique abstract part cannot be defined for both KVP and XML bindings, specific message types will have to be defined.

It was agreed not to try to merge both and that we will leave that to future work as technologies will be available and/or supported by COTS tools.

For clarity, the message definitions in WSDL documents will be followed by `_KVP` or `_XML`.

7 WSDL for existing OGC Services

This section describes the use of WSDL to describe the **existing** OGC Web Service interfaces. This includes

- WCS 1.0.0
- WFS 1.0.0
- WMS 1.1.0 (1.1.1?)
- CS-W 2.0.0

Each paragraph below will address the problems encountered to map the interfaces to WSDL (if any); this will be done first for KVP bindings, then for SOAP bindings. The WSDL documents as well as the required information will be part of the respective change requests. This document intends to contain justification of implementation choices, not results. Of course, a pointer to the appropriate OGC document numbers will be provided with each change proposal.

7.1 KVP bindings

7.1.1 Using existing schemas

As noted above, input messages in KVP bindings cannot have complex types for their parts. However, output messages can, and existing schemas can be used to define the response and fault messages of existing services.

This implies using the existing elements in the response part definition

```
<message name="GetCapaResult">
  <part name="response" element="wcs:WCS_Capabilities"/>
</message>
```

and using the mime:mimeXML in the operation binding :

```
<operation name="GetCapabilities">
  [...]
  <output>
    <mime:mimeXML/>
  </output>
</operation>
```

That way, the GetCapabilities response should be validated against the WCS_Capabilities element definition.

7.1.2 WSDL for WCS 1.0.0

There is a strong incompatibility between the current OGC WCS GET interface and the possibility of WSDL. The WCS specification 1.0.0 requires the use of arbitrary parameter names (e.g. BAND=...), and the WSDL message definition needs to be exhaustive about the possible parameter names.

The WG agreed to compute a change proposal against the specification and to suggest the Common Implementation specification to include a paragraph about this issue.

7.1.2.1 Binary response

There is an issue with operations that only provide a binary as response message. This limitation applies to WCS GetCoverage and WMS GetMap. There exists no such thing as an 'xsd:binary' type, which would be needed to define the response message part.

Actually it doesn't make much sense to define a raw, unencoded binary type within an XML schema, but then how should we map the binary response of the remote service to some message part ?

This seems to be a known but unresolved issue in the W3C note. See WSD Issues List and the post from Jeff Lansing <http://dev.w3.org/cvsweb/2002/ws/desc/issues/wsd-issues.html?rev=1.2#x7>.

7.1.2.2 Optional parts (and by extension, Get parameters)

According to the WSDL note, The 'minOccurs' attribute cannot be set on a wsdl:part element, meaning we cannot define an optional parameter in the KVP encoding.

There seems to be no way to get around this limitation; all message parts will be mandatory.

See <http://groups.yahoo.com/group/wSDL/message/454>.

7.1.3 WSDL for WFS 1.0.0

Some changes has been necessary to be able to describe the WFS using WSDL. Because of these problems, the conclusion is that no 1.0.0 version of WFS can be described using WSDL, so we made some schema fixes to version 1.0.1.

Here is a summary of the changes that we had to do in the WFS 1.0.0 schemas,

WFS-basic

- included WFS-capabilities.xsd to incorporate capabilities elements
- moved wfs:WFS_Capabilities element to the section for response messages

WFS-transaction

- removed duplicate type definition, EmptyType (also defined in WFS-capabilities.xsd)

WFS-capabilities

- import filterCapabilities.xsd from "http://www.opengis.net/ows" namespace
- renamed duplicate element declarations to resolve name collisions, substituting lower case for first letter: query, insert, update, delete, lock (also the references in OperationType)
- renamed component elements of RequestType in the same manner to resolve duplicates: getCapabilities, describeFeatureType, etc.
- renamed duplicate type definitions to resolve name collisions: getCapabilitiesType, transactionType, etc.

filterCapabilities.xsd

- changed target namespace to "http://www.opengis.net/ows" namespace from "http://www.opengis.net/ogc", as this was the simplest way to resolve the many name collisions (most filter elements are redefined for inclusion in capabilities documents)

Moreover, when binding an interface to an HTTP method, the WS-I Basic profile (and consequently most tools) doesn't allow any operations to be omitted (R2718), so all of them have to be bound to that method (e.g. GET).

XML Spy 2004 rel. 3 (Enterprise ed.) does not process HTTP bindings. However, if the HTTP endpoints and bindings are stubbed out it does construct and submit SOAP request messages; the target service responds appropriately.

7.1.4 WSDL for WMS 1.1.1

The WMS service is subject to the same problems as described in the WCS part, namely the binary response problem, and the lack of a minOccurs attribute. Besides that, there's also a problem due to the lack of WMS XMLSchema, as described below.

7.1.4.1 WSDL description of Services using DTDs

Current WMS services return XML responses validated with a DTD document, ignoring any namespace definition.

This prevents the validation of WSDL response messages against a XML schema for the WMS, since the schema definitions must be in the scope of some namespace (this is not mandatory in the schema spec, but in practice having a WSDL document using schemas without namespaces leads to many more problems).

There are two solutions to this issue:

- Specify the part encoding as text/xml without specifying the kind of element returned. This will lower the granularity of the code generated by tools, but it will work with existing services.
- Change the implementation specification to use XML Schema definition

7.1.5 WSDL for CS-W 2.0

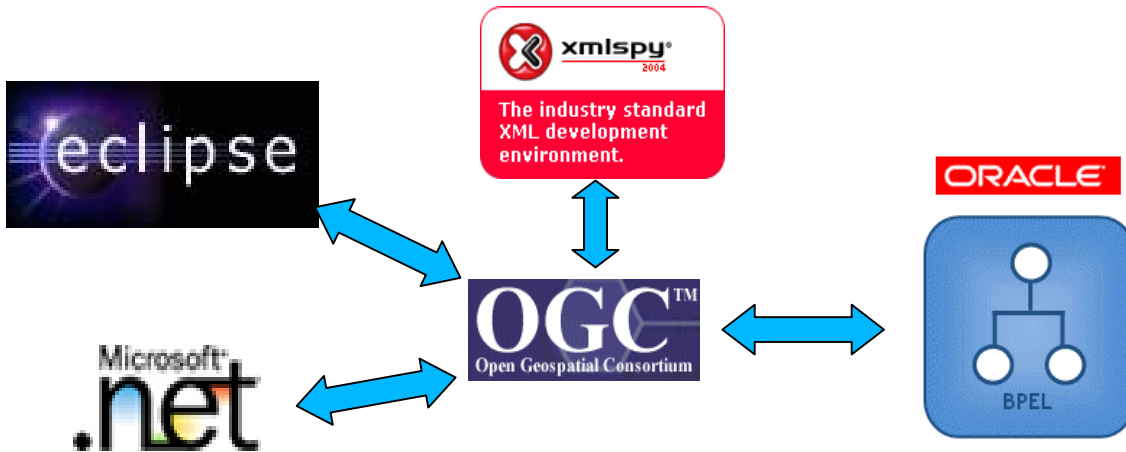
The only problem we encounter in Catalog Service-Web is that a GetRecords request may be processed asynchronously, but a WSDL interface definition allows only one output message to be specified. A type definition was added to the wrapper schema to provide a choice between a normal GetRecordsResponse and an asynchronous response (i.e. an acknowledgement).

7.1.6 WSDL for Common Implementation Specification 1.0.0

The group recommend that the Common Implementation Specification specify a common usage for WSDL. A formal change proposal has been developed to include a section that describes the normative impact of including WSDL description in a specification and the way it shall be used.

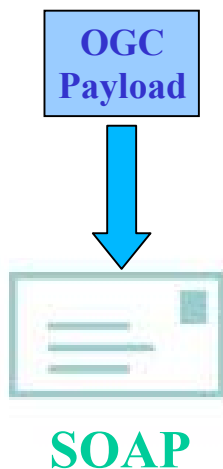
7.1.7 WSDL Interoperability Experiments

Various tools has been used to experiment the produced WSDL. In particular, the IH4DS thread have used the XSDL description of the OWS to integrate them with a workflow engine provided by Oracle (was Collaxa BPEL). See the IH4DS IPRs and the Annexes for more details about these experiments.



7.2 XML & SOAP Bindings

This section will describes the WSDL part for SOAP access to existing OGC Web Service interfaces. This means, we will reject any try to change message semantic sor encoding sthat are not an absolute requirement to make a SOAP message workcorrectly.



The path chose reduce sthe impact on existing specification sand existing implementations. It requires only theembed dingof the existing XML message in the body of the SOAP envelopes, using document/literal option sof the SOAP protocol. This form of a SOAP envelope is the one that has the most support from the industry. This choice has proven to be very efficient for existing implementations and has working quite well with existing COTS SOAP Tools.

Nevertheless, we envision that more SOAP functionalities may be used by future OGC Services. An interesting document that provides more information on this areaof research is the Messaging Framework Discussion Paper (See Section 3).

We have identified some general issues regarding SOAP as a transport mechanism. This section will address them and provide a comprehensive answer to them.

7.3 Issues with SOAP

7.3.1 SOAP document/literal as the default OGC choice

When using SOAP binding in a WSDL document, one has to choose the style and encoding of the SOAP messages. A WSDL SOAP binding can be either an RPC style binding or a document style binding. Furthermore, A SOAP binding can have an encoded use or a literal use. Of the four resulting combinations, document/literal binding was chosen for OGC messages, because it is the one that matches best the way transport layers are designed in current OGC services (XML messages validated by OGCschemas) and because it allows for direct use of OGC schemas in the SOAP messaging scheme.

See <http://www-106.ibm.com/developerworks/webservices/library/ws-whichwsdl/> for more information on encodings, styles and how to use them.

7.3.2 SOAP for large binary transport

Transport of large binary data as part of SOAP messages can be an issue, and is involved in several OGC services (e.g. WMS maps, WCS coverage data).

Two approaches can be taken to cope with binary data inside SOAP messages. One is to put data inline in the messages; but since SOAP messages are XML-encoded and can't contain raw binary data, data has to be encoded (using hex- or base64-encoding) to get valid XML documents. To achieve this, extra encoding and decoding steps are needed. Therefore, that approach needs much more processing and bandwidth and puts performance at stake. The other approach is to put binary data as attachment to the SOAP messages, using a standard such as MIME; this combination (SOAP with MIME attachments, a.k.a. SwA) is the subject of a W3C Note (<http://www.w3.org/TR/SOAP-attachments>) and is part of the WS-I guidelines. It has the advantage of avoiding encoding of data; however, resulting messages, as a whole, are no longer valid XML messages, and may cause problems if routed through XML-based frameworks that do not recognize MIME messages.

Using SOAP with attachments seems the way to go, and WSDL documents created in the scope of OWS-2 were designed with that approach in mind. However, support for MIME messages in existing XML tools (e.g. BPEL) should be investigated.

The Attachments Profile (<http://www.ws-i.org/Profiles/Basic/2003-08/AttachmentsProfile-1.0.htm>), part of the WS-I Profile, provides guidelines on how to use MIME or DIME attachments within SOAP messages.

8 UDDI

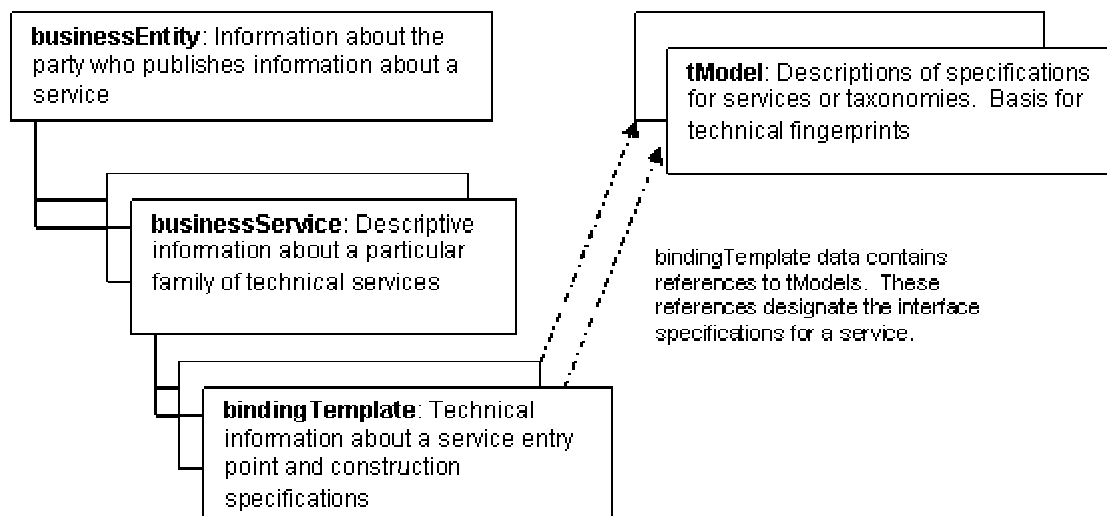
The overall goal of this initiative is to use OGC web services in conjunction with the industry standards WSDL/SOAP/UDDI. This part will focus on the registration process of OGC web service into public UDDI Registries.

8.1 Registration of OGC Web Services into UDDI

This first step will provide a methodology to register a W*S into a public UDDI registry. It will cover the most common use of UDDI, the registration of Web Services.

8.1.1 The UDDI data model

As far as web services registration is concerned, the UDDI data model is made up of several main entities :



8.1.2 WSDL and UDDI registries

The Oasis note on how to use WSDL in a UDDI registry (<http://www.oasis-open.org/committees/uddi-spec/doc/tn/uddi-spec-tc-tn-wsdl-v2.htm>) provides a good starting point to describe how one should register a web service into a UDDI registry, using its WSDL documents.

That document proposes a methodology for mapping WSDL data model entities onto UDDI data model entities. Basically, that methodology maps each building block of a WSDL document to a separate UDDI entity : a tModel for portTypes, another tModel for bindings, a UDDI businessService for the WSDL service, and a bindingTemplate for the WSDL Port.

One remark about the Oasis guidelines is that they assume the WSDL bindings to be part of the reusable portion of WSDL that will be shared between services; in other words, they consider bindings as part of the interface. This does not invalidate the methodology described in the note, but raises the issue of whether bindings should be considered normative or not.

8.2 Two ways assertion

This second step will exercise another concept of UDDI registries, it is the ability to manage and to assert relationships between business entities.

8.2.1 Use case

In this experiment the use case is as is,

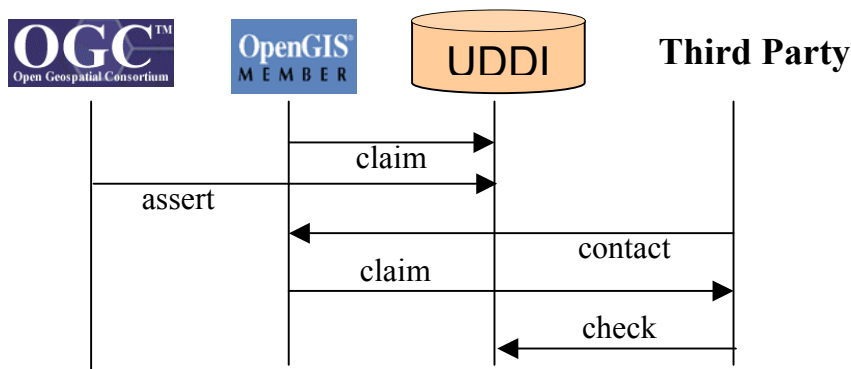
- An OGC member claims compliance of its product in UDDI.

The assertion is visible to the member and to the OGC. All other parties does not see it yet. It must be validated by the OGC.

- The OGC assert the claim is right.

Since the OGC recognise the claim as true, it becomes available to all parties that accesses the UDDI registry.

- A third party contact the OGC member and ask for compliance.
- The OGC member claims again compliance of its product.
- The third party can check using the UDDI registry that the claim is recognised as valid by the OGC.



This case raise some questions about the security in UDDI registries, which is one major goal of the industry leaders that support UDDI. And also whether the OGC is interested in publishing the compliance of its member into a public registry.

8.2.2 Implementation 1: Illustrate Scenario with no custom UDDI objects

The first implementation used the Systinet UDDI User interface and the predefined unchecked relationship taxonomy provided in the UDDI specification.

The following snapshots of the Systinet UI screens illustrate the process:

Figure 1: Illustrates OGC member adding assertion on WCS Compliance

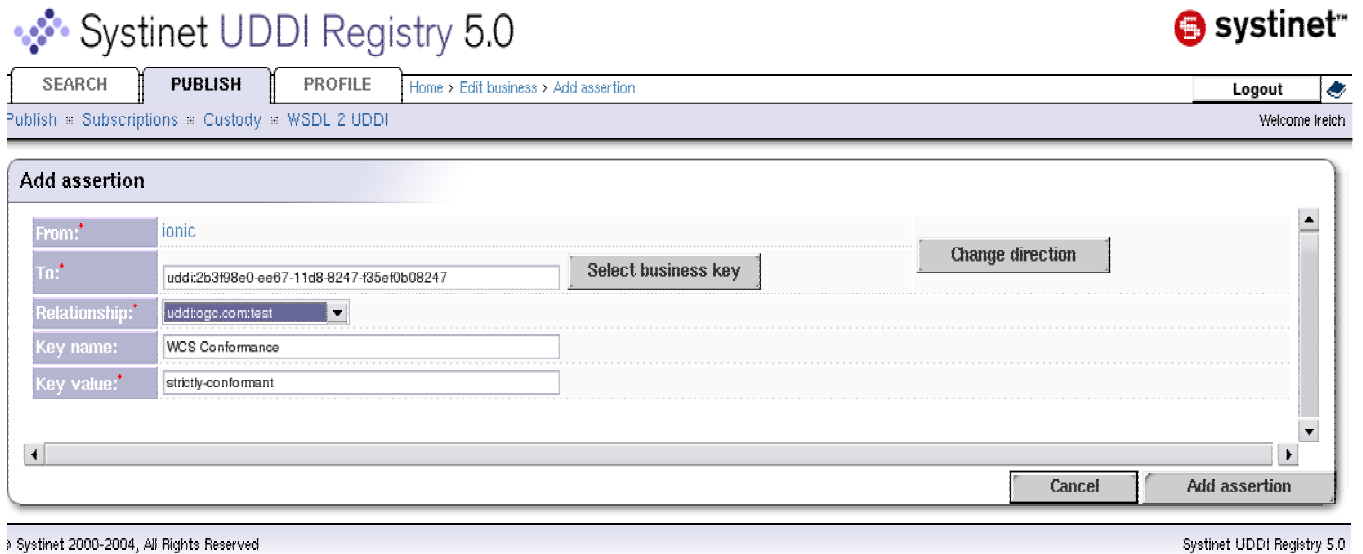


Figure 2: Illustrates UDDI response to add assertion request

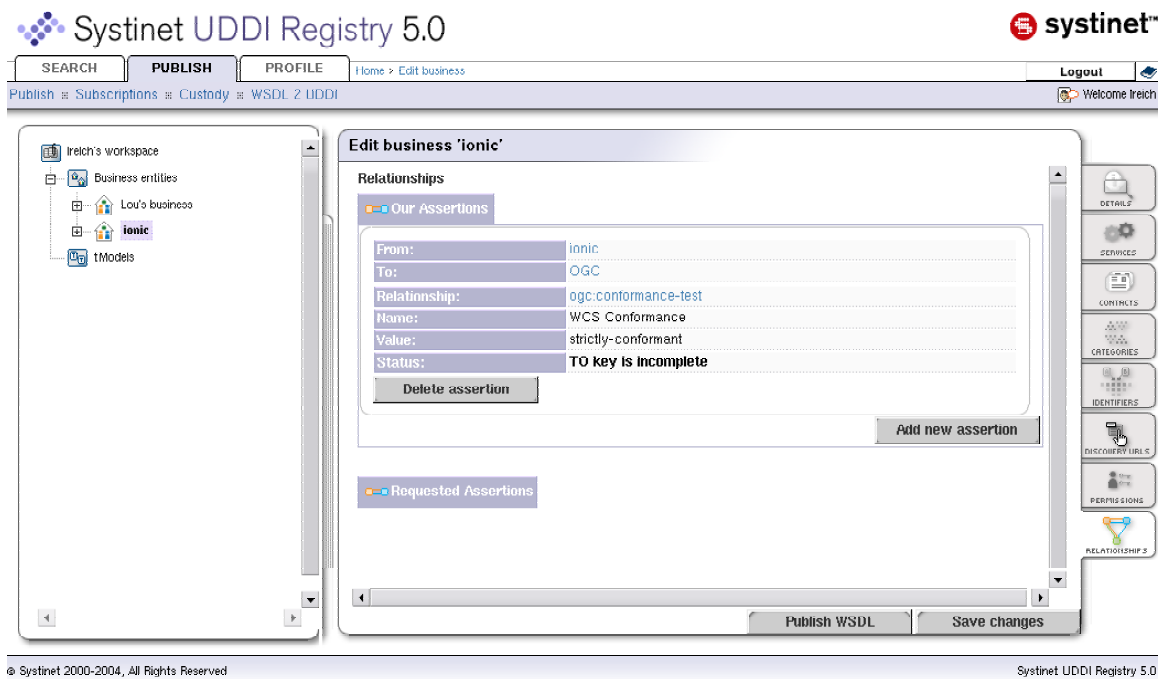


Figure 3: Illustrates the alerting of the OGC Administrator of the assumption

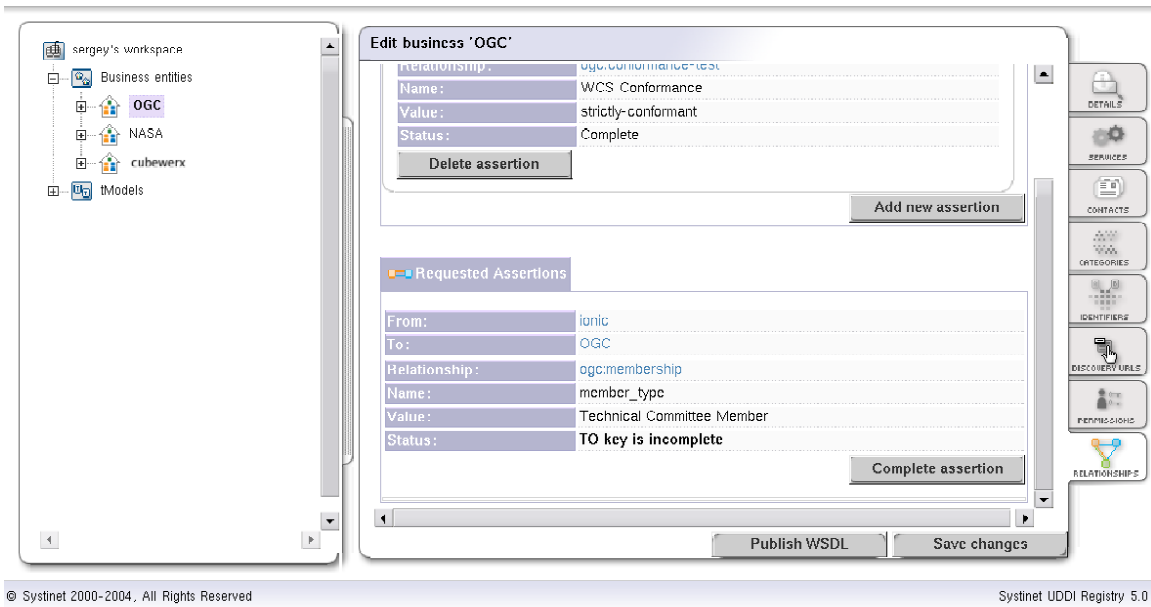


Figure 4: Shows the OGC view after approving the assumption

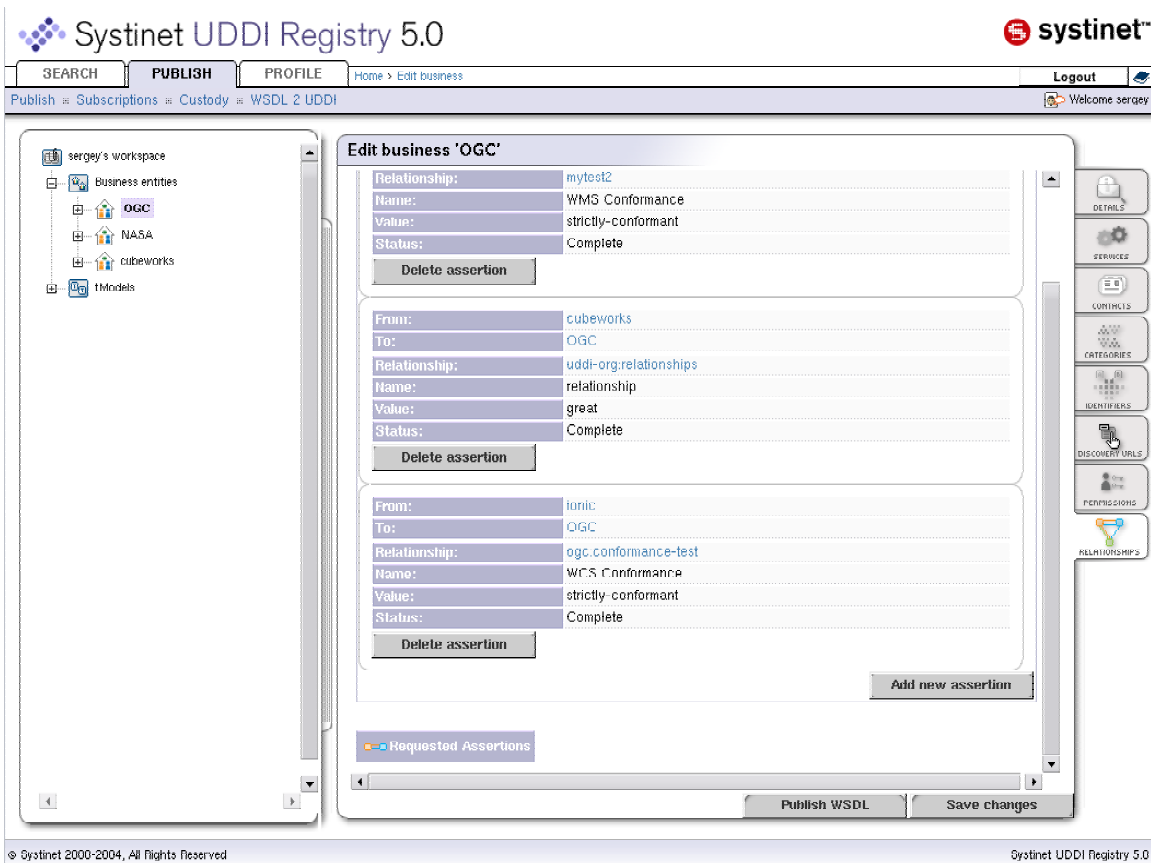


Figure 5 shows a user requesting a business that are related to OGC:

The screenshot shows the Systinet UDDI Registry 5.0 interface. At the top, there is a navigation bar with tabs for SEARCH, PUBLISH, and PROFILE. The current page is 'Find related businesses'. Below the navigation bar, there is a breadcrumb trail: Find business :: Find service :: Find binding :: Find tModel :: Find related businesses :: Browse taxonomies :: Direct get. A 'Logout' button is visible in the top right corner.

The main content area is titled 'Find related businesses' and contains a section for 'Find related businesses - Search results'. It displays 'Displaying results 1 - 1 of 1'. The search results are shown in a table-like format with the following data:

From:	cubewerx	To:	OGC
Relationship:	ogc:conformance-test		
Name:	WMS Conformance		
Value:	strictly-conformant		
Relationship:	ogc:conformance-test		
Name:	WCS Conformance		
Value:	profile		
Relationship:	ogc:membership		
Name:	member_type		
Value:	Technical Committee Member		
Relationship:	ogc:conformance-test		
Name:	WFS Conformance		
Value:	conformant		

At the bottom of the search results area, there are two buttons: 'Search' and 'Results'. The footer of the page contains the copyright notice: © Systinet 2000-2004, All Rights Reserved and Systinet UDDI Registry 5.0.

8.2.3 Observations and Conclusions

This experiment showed that it was easy to add assertions at a business entity level in UDDI using a standard taxonomy and UI. However, the scope of the experiment made it impossible to address some questions about the security in UDDI registries, which is one major goal of the industry leaders that support UDDI.

Also given the fact that the assertion capability can only be made at the business entity layer and there are no current plans to extend the capability to include service descriptions, it is questionable whether there is significant value to this functionality. For example, it is questionable whether the OGC is interested in publishing either its membership list or the compliance of its member into a public Web Service registry.

However the background work in setting up the test UDDI registry led us to a better understanding of the validation services provided by UDDI for various classes.

This functionality became the focus of the remainder of the experiment.

8.3 Developing a custom validating Taxonomy

In the previous experiment we felt a small set of term/classes that were valid values of the conformance class would be valuable. However, the relationship taxonomy was not validated so any test could be inserted for the value. This is shown in the following figure where “great” is given as a value to relationship.

The screenshot shows the Systinet UDDI Registry 5.0 interface. The main content area displays search results for 'Find related businesses'. The results are as follows:

Find related businesses - Search results			
Displaying results 1 - 3 of 3			
From:	Lou's business	To:	OGC
Relationship:	mytest2		
Name:	WMS Conformance		
Value:	strictly-conformant		
From:	cubeworks	To:	OGC
Relationship:	uddi-org:relationships		
Name:	relationship		
Value:	great		
From:	ionic	To:	OGC
Relationship:	ogc:conformance-test		
Name:	WCS Conformance		
Value:	strictly-conformant		

At the bottom of the results table, there is a page indicator showing '1'.

After much experimenting we were able to use the Systinet UDDI and a Systinet method which allows keyword value pairs to be validated against an XML list in the tmodel.

While this made the assertion experiment more interesting since a controlled vocabulary could be entered, the restriction of business entity relationships only still could not be overcome.

However, the fact that the validation service could be a user defined web service led to a very interesting sequence of experiments discussed in the remainder of this section.

8.3.1 Technical aspects

In order to create a custom validating taxonomy in Systinet UDDI Registry 5.0 the following steps need to be taken:

1. A checked taxonomy should be created with the specification of external Java class that would perform actual validation.
2. The validating class needs to be implemented. The class must implement *org.systinet.uddi.client.valueset.validation.v3.UDDI_ValueSetValidation_PortType* interface. This interface declares jut one method:

```
public DispositionReport
    validate_values(Validate_values body) throws
        UDDIException
```

The `Validate_values` object contains at least one `tModel`, `businessEntity`, `businessService`, `bindingTemplate` or `publisherAsertion`, which contains reference to the taxonomy validated by this web service. No matter what exact UDDI structures are passed into this method, eventually `KeyedReferences` that are contained in these structures will be validated:

```
private void validate(KeyedReferenceArrayList
    keyedReferenceArrayList, DispositionReport
    report) throws UDDIException {
    for (Iterator iter =
        keyedReferenceArrayList.iterator();
        iter.hasNext();) {
        KeyedReference keyedReference =
            (KeyedReference) iter.next();
        if
            (TMODEL_KEY.equalsIgnoreCase(keyedReference.
                getTModelKey())) {
            if
                (!isWithinBoundingBox(keyedReference.getKeyV
                    alue())) {
                    String message = "Given point
                        is not within Bounding Box " +
                            keyedReference.toXML();

                    report.addResult(createResult(UDDIErrorCodes
                        .E_INVALID_VALUE, message));
                }
            }
        }
    }
}
```

3. Once the taxonomy and the class are created, the taxonomy needs to be uploaded to the registry and the class needs to be put on the registry's classpath.

In this example, a pair of (X,Y) coordinates is validated against predefined bounding box. In other words, if the pair is inside of the box it is considered to be valid:

```
public static boolean
    isWithinBoundingBox(String input)
    {
        boolean answer = false;
        try
```

```

    {
        StringTokenizer tokenPoint = new
StringTokenizer(input, ", ");
        Point point = new Point
(Double.parseDouble(tokenPoint.nextToken()),
Double.parseDouble(tokenPoint.nextToken()));
        answer = BB.isPointWithin(point);
    }
    catch (Throwable e)
    {
        e.printStackTrace();
    }
    return answer;
}

```

So, for example, if a tModel created containing the X,Y pair and it needs to be validated against the taxonomy, Java code that does it may look like this:

```

    TModel tModel = new TModel();
    tModel.setName(new Name(name));
    tModel.addDescription(new
Description(description));

    tModel.setCategoryBag(new CategoryBag());

    tModel.getCategoryBag().addKeyedReference(new
KeyedReference(BoundingBoxValidation.TMODEL_
KEY, keyValue));

    Save_tModel save = new Save_tModel();
    save.addTModel(tModel);
    save.setAuthInfo(authInfo);
    UDDI_Publication_PortType publishing =
getPublishingStub();
    System.out.print("Save in progress ...");
    TModelDetail tModelDetail =
publishing.save_tModel(save);\

```

If validation of the values contained in the tModel successful, the tModel will be saved, otherwise *DispositionReport* containing the failure information will be returned from the validation service. This code from the validation service illustrates that:

```

    ResultArrayList results =
report.getResultArrayList();
    if (results == null || results.size() ==
0)
        return
DispositionReport.DISPOSITION_REPORT_SUCCESS
;
    throw new UDDIException(report);

```

Validation can be performed via Registry's Web UI. First, the taxonomy cab browsed:

Browse taxonomy

Name:	demo:BoundingBox
tModel key:	uddi:csc.com:demo:BoundingBox
Description:	A category system used to connect tModel with a Bounding Box, that describes its API.
Compatibility:	tModel
Categorization:	categorization
Validation:	external class:demo.uddi.validation.BoundingBoxValidation
Unvalidatable:	no

Key name:
Key value:

Then, if values (5,5) that are valid according to the business logic of the taxonomy's validation service are input, the previously published tModel can be found. Thus, the values are validated via the service:

Find tModel

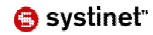
Search results
Displaying results 1 - 1 of 1

Coordinates values	Name	Description
	Demonstrates Bounding Box validation service	1

Search:

However, if values that are not valid are input (11,11), different result is returned:

Systinet UDDI Registry 5.0



[ARCH](#) | [PUBLISH](#) | [PROFILE](#) | [MANAGE](#) | Home > Find taxonomy > Browse taxonomy | [Logout](#)
[Business](#) > [Find service](#) > [Find binding](#) > [Find tModel](#) > [Find related businesses](#) > [Browse taxonomies](#) > [Direct get](#) | Welcome admin

Find taxonomy

name:	demo:BoundingBox
del key:	uddi.csc.com:demo:BoundingBox
description:	A category system used to connect tModel with a Bounding Box, that describes its API.
compatibility:	tModel
categorization:	categorization
validation:	external class:demo.uddi.validation.BoundingBox:Validation
validatable:	no

by name:
 by value:

Systinet 2000-2004, All Rights Reserved | Systinet UDDI Registry 5.0

Systinet UDDI Registry 5.0



[SEARCH](#) | [PUBLISH](#) | [PROFILE](#) | [MANAGE](#) | Home > Find taxonomy > Browse taxonomy > Find tModel | [Logout](#)
[Find business](#) > [Find service](#) > [Find binding](#) > [Find tModel](#) > [Find related businesses](#) > [Browse taxonomies](#) > [Direct get](#) | Welcome admin

Find tModel

Search results

name	description
No tModels found.	

|

Systinet 2000-2004, All Rights Reserved | Systinet UDDI Registry 5.0

In this case, the bounding box is defined as 4 points:

(0,0),(0,10),(10,10),(10,0).

8.3.2 Service registration with validation with user defined checked taxonomy

In this example, previously created is modified and used to validate a service. Taxonomy's validator class is modified to validate whether a four given points constitute a valid bounding box and if that box is within the bounding box defined by taxonomy:

```

public static boolean isWithinBoundingBox(String input)
{
    boolean answer = false;
    try
    {

```

```

StringTokenizer tokenPoint = new StringTokenizer(input, " ");
Point pointLL = new Point (Double.parseDouble(tokenPoint.nextToken()),
                           Double.parseDouble(tokenPoint.nextToken()));
Point pointUL = new Point (Double.parseDouble(tokenPoint.nextToken()),
                           Double.parseDouble(tokenPoint.nextToken()));
Point pointUR = new Point (Double.parseDouble(tokenPoint.nextToken()),
                           Double.parseDouble(tokenPoint.nextToken()));
Point pointLR = new Point (Double.parseDouble(tokenPoint.nextToken()),
                           Double.parseDouble(tokenPoint.nextToken()));

    answer = BB.isValidBox(pointLL,pointUL,pintUR,pointLR) &&
    BB.isPointWithin(pointLL) && BB.isPointWithin(pointUL) &&
    BB.isPointWithin(pointUR) && BB.isPointWithin(pointLR);
}
catch (Throwable e)
{
    e.printStackTrace();
}

return answer;
}

```

Lets say someone wishes to publish a service that is described by a bounding box. Additionally that bounding box must be valid according to earlier created taxonomy. Thus, it needs to be validated against the taxonomy. Java code that does it may look like this:

```

BusinessService service = new BusinessService();
service.setBusinessKey("uddi:b2844320-f072-11d8-8249-f35ef0b08247");
service.setNameArrayList(new NameArrayList());
service.getNameArrayList().add(new Name(name));
service.addDescription(new Description(description));
service.setCategoryBag(new CategoryBag());
service.getCategoryBag().addKeyedReference(new
KeyedReference(BoundingBoxValidation.TMODEL_KEY, keyValue));

Save_service save = new Save_service();
save.addBusinessService(service);
save.setAuthInfo(authInfo);

UDDI_Publication_PortType publishing = getPublishingStub();
System.out.print("Save in progress ...");
ServiceDetail serviceDetail = publishing.save_service(save);
System.out.println(" done");
return serviceDetail;

```

If validation of the values associated with the service successful, the service will be saved, otherwise *DispositionReport* containing the failure information will be returned from the validation service. This code from the validation service illustrates that:

```

ResultArrayList results = report.getResultArrayList();

if (results == null || results.size() == 0)

    return DispositionReport.DISPOSITION_REPORT_SUCCESS;

throw new UDDIException(report);

```

Log 1 is a record adding service with category value that is valid by bounding box taxonomy

Running ValidationDemo demo...

*** Systinet Registry Demo - BoundingBoxValidationDemo ***

Saving tModel where

Enter name [Coordinates values]:

Enter description [Demonstrates Bounding Box validation service]:

Default bounding box is used. It will be validated against the taxonomy. In this case it is outside of the bounds defined by taxonomy.

BBox [1,1,1,4,4,4,4,1]:

Using Security at url <https://sindbad.gsfc.nasa.gov:8443/uddi/security> ..

done

Logging in .. done

name = Coordinates values

description = Demonstrates Bounding Box validation service

keyValue = 1,1,1,4,4,4,4,1

Using Publishing at url <https://sindbad.gsfc.nasa.gov:8443/uddi/publishing> .. done

Save in progress ... done

Service 1 : uddi:7497d270-1318-11d9-9ec3-4b0f99d09ec3

```
<businessService serviceKey="uddi:7497d270-1318-11d9-9ec3-4b0f99d09ec3"
businessKey="uddi:b2844320-f072-11d8-8249-f35ef0b08247" xmlns="urn:uddi-org:api_v3">
```

```
<name>Coordinates values</name>
```

```
<description>Demonstrates Bounding Box validation service</description>
```

```
<categoryBag>
```

```
<keyedReference tModelKey="uddi:csc.com:demo:BoundingBox" keyValue="1,1,1,4,4,4,4,1"/>
```

```
</categoryBag>
```

```
</businessService>
```

```
*****
```

Logging out .. done

The run shows that business service was saved and its XML representation printed to the screen.

Log 2 shows adding service with category value that is not valid by bounding box taxonomy

Running ValidationDemo demo...

*** Systinet Registry Demo - BoundingBoxValidationDemo ***

Saving tModel where

Enter name [Coordinates values]:

Enter description [Demonstrates Bounding Box validation service]:

BBox [1,1,1,4,4,4,1]: 11,11,11,14,14,14,14,11

Using Security at url <https://sindbad.gsfc.nasa.gov:8443/uddi/security> .. done

Logging in .. done

name = Coordinates values

description = Demonstrates Bounding Box validation service

Key value is the value of bounding box that is going to be validated against the taxonomy. In this case it is outside of the bounds defined by taxonomy.

keyValue = 11,11,11,14,14,14,14,11

Using Publishing at url <https://sindbad.gsfc.nasa.gov:8443/uddi/publishing> .. done

Save in progress ...Exception in thread "main"
org.systinet.uddi.client.v3.UDDIException:

```
<dispositionReport xmlns="urn:uddi-org:api_v3">
  <result errno="20200">
    <errInfo errCode="E_invalidValue">
      Given box is not within Bounding Box &lt;keyedReference
        tModelKey="uddi:csc.com:demo:BoundingBox"
        keyValue="11,11,11,14,14,14,14,11"
        xmlns="urn:uddiorg:api_v3"/>
    </errInfo>
  </result>
</dispositionReport>
```

The run shows that business service was not saved. The XML representation of Disposition Report was printed to the screen. The report encapsulates the error given by the validation service.

Note:

- in these example the bounding box represented by these 4 points is used inside of the taxonomy: (0,0),(0,10),(10,10),(10,0).
- When value are input via WEB UI, they are provided as 8 comma separated numbers. Each two of these numbers represent one of the corners of the bounding box in this order: lower left, upper left, upper right, lower right.

Previously added service can be browsed through the registry UI:

The screenshot shows the Systinet UDDI Registry 5.0 interface. The main content area displays the 'View service' page for 'Coordinates values'. The page includes a navigation menu on the left with 'Lou's business', 'boundingboxservice', and 'Coordinates values'. The main content area shows a table with columns 'tModel', 'key name', and 'key value'. The table contains one row with 'demo:RoundingBox' as the key name and '1,1,1,4,4,4,4,1' as the key value. Below the table, there is a section for 'Category groups' with the message 'No category groups found.' At the bottom of the page, there are buttons for 'Back', 'View as XML', 'Delete', and 'Edit'. The footer of the page contains the copyright notice '© Systinet 2000-2004, All Rights Reserved' and the version 'Systinet UDDI Registry 5.0'.

tModel	key name	key value
	demo:RoundingBox	1,1,1,4,4,4,4,1

8.3.3 Conclusions and observations

I think this is a significant result and should be investigated further in our discussions of UDDI and ebxml registry/repository. This experiment took less than 5 mandays to implement but we had already gathered a fair amount of UDDI v3 experience from the previous portions of the OWS 2 CA experience. The goal was to discover if there were obvious limits on the use of UDDI version 3 validation web services using the Systinet v5 UDDI server.

The key point is that the item to be validated is passed to the specified validation web service as a string, the string is then parsed by the validation service. This means we might be able to validate GML bounding shapes (assuming a GML validator exists) for services in a taxonomy/class. I believe that it would be relatively easy to implement complex validation services.

9 MapInfo - TIEs Executive Summary

During the OWS-2 kick-off in Washington, MapInfo volunteered to write .NET and Java WFS and WMS clients. Attached are reports on our efforts. Also included is a document with tips for .NET development that the OWS-2 common architecture team might find useful.

We used Visual Studio .Net and Eclipse with the plug-ins for WSDL. These two IDE's are the most popular in the software development industry. The testing can be summarized as follows:

WFS

WMS

Validate WSDL	fail	pass
Generate Java Stubs	not possible	pass with changes
Execute Java Demo	not possible	not possible
Generate .NET Stubs	not possible	fail
Execute .NET Demo	not possible	not possible

It appears that the current stub generators are limited and not able to use the current WSDL files for WFS and WMS. Another way of looking at it -- maybe the current GML, WFS and WMS schemas are too advanced and perhaps OGC should investigate a basic profile for web services support.

10 MapInfo - Report on WSDL WFS

10.1 Problems

10.1.1 Schema Definitions

10.1.2 Problem

Microsoft expects all XML Schema definitions within one namespace to be placed into a single file.

10.1.3 Solution

Collapse all types within the same namespace into a single file.

10.2 Members of Type Object

10.2.1 Problem

Wsd.exe utility will generate members of type 'object' for definitions like:

```
<element ref="xls:_RequestParameters" minOccurs="0"/>
```

10.2.2 Solution

Use the following format for definitions instead:

```
<element name="_RequestParameters" type="xls:AbstractRequestParametersType"/>
```


10.3 Multiple Occurrence of Elements of the Same Type

10.3.1 Problem

For multiple occurrences of elements of the same type wsdl.exe generates erroneous mappings.

For example:

For:

```
<complexType name="PointToPointDistanceType">
  <sequence>
    <element name="startPoint" type="gml:Point"/>
    <element name="endPoint" type="gml:Point"/>
  </sequence>
  <attribute name="distanceType" type="mixs:DistanceCalculationType"/>
  <attribute name="distanceUnit" type="xls:DistanceUnitType" default="KM"/>
</complexType>
```

The mapping generated is:

```
    /// <remarks/>

[System.Xml.Serialization.XmlElementAttribute (Namespace="http://www.opengis.net/gml" )]

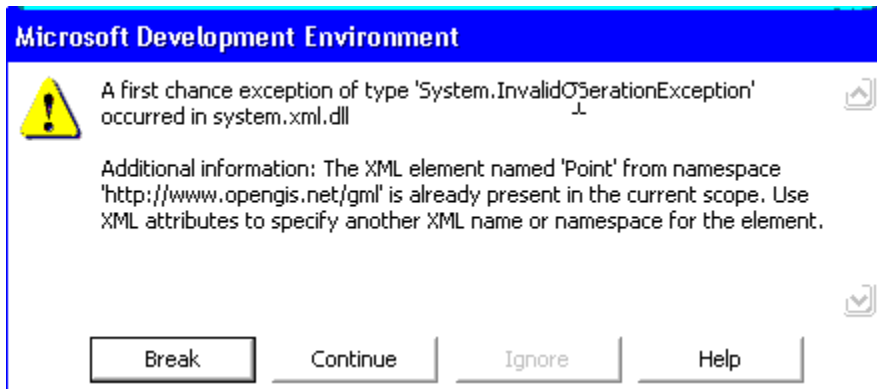
    public PointType Point;

    /// <remarks/>

[System.Xml.Serialization.XmlElementAttribute ("Point",
Namespace="http://www.opengis.net/gml" )]

    public PointType Point1;
```

And will produce the following exception:



10.3.2 Solution

The class generated should be changed to

```

/// <remarks/>

[System.Xml.Serialization.XmlElementAttribute("Point",
Namespace="http://www.opengis.net/gml")]

public PointType Point;

/// <remarks/>

[System.Xml.Serialization.XmlElementAttribute("Point1",
Namespace="http://www.opengis.net/gml")]

public PointType Point1;

```

10.4 Schemas

The utilities `wSDL.exe` and `xsd.exe` generate a single C# class as an output for as many schemas as you specify. In other words, if one service references some types or elements of the schema the output class will contain all types for namespaces specified.

11 MapInfo - Report on WSDL WFS

11.1 Overview

This section reviews testing of the WFS WSDL contribution. The files from the folder *WFS-1.1* posted to the OGC Web Services, Phase 2 section of the OGC portal was used for this test.

11.2 Steps

1. Validation of WSDL
2. Generation of stubs to call services
3. Use of stubs in a program to test services

11.3 Tools

- Eclipse 3.0.0 (<http://www.eclipse.org/platform>)
- WSDL/SOAP Validation Plug-in for Eclipse (<http://www.eclipse.org/wsvt/>)
- WSDL2Java (<http://ws.apache.org/axis>)
- Visual Studio .NET 2003 Architect Edition
- XMLSPY 5 Professional Edition

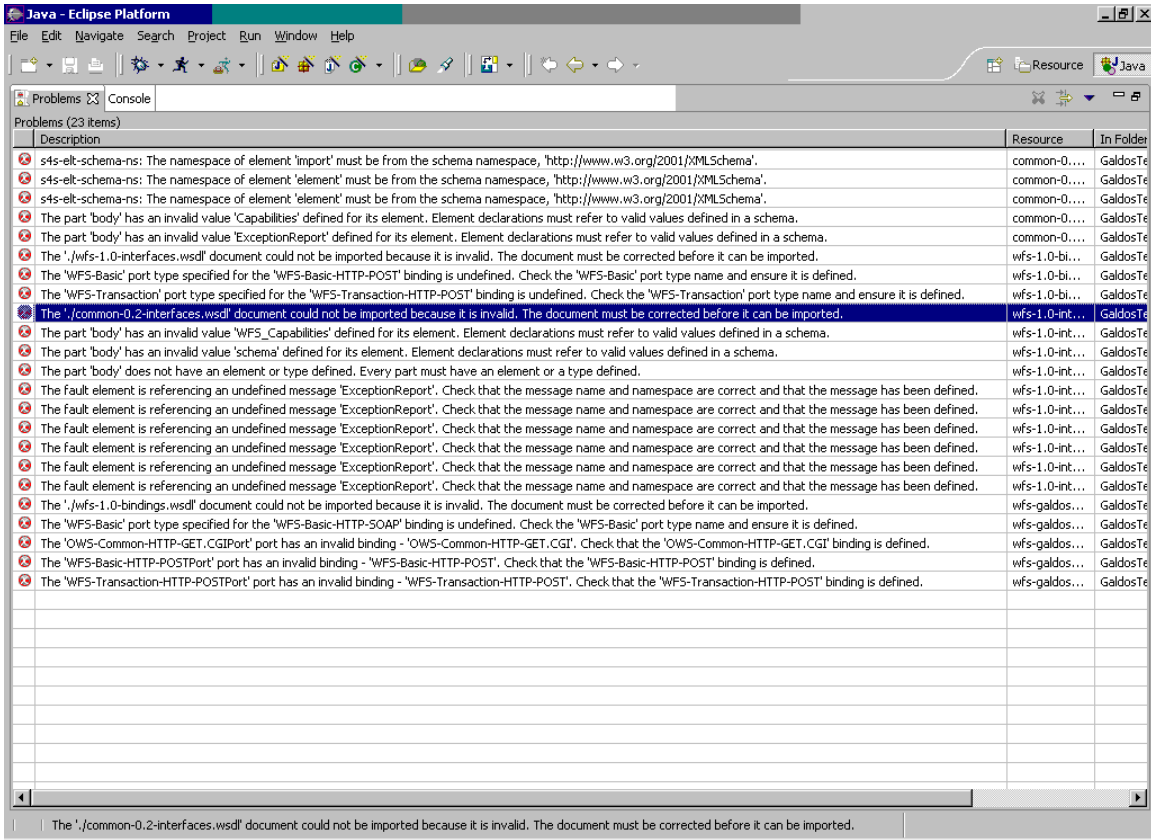
11.4 Notes

- Validation was done using the WSDL validation plug-in for Eclipse
- Generation of Java stubs was attempted using WSDL2Java

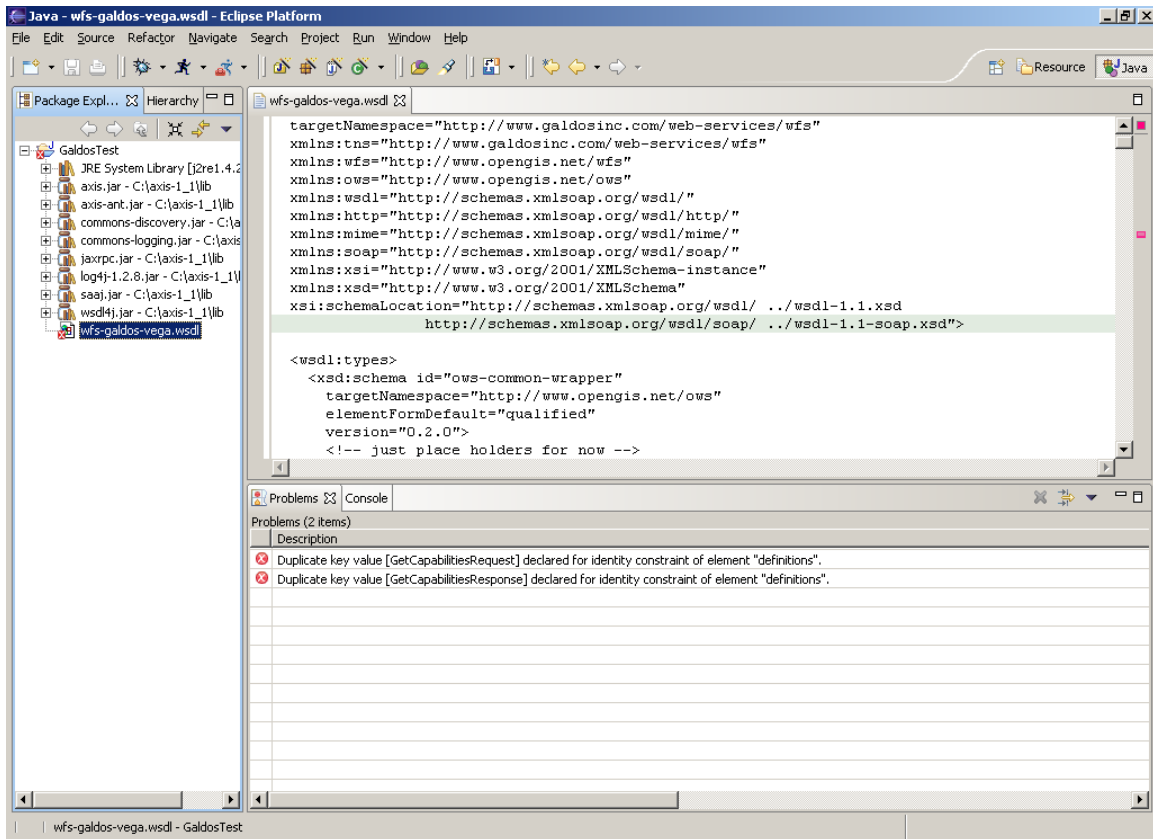
11.5 Java Use of WFS

11.5.1 WSDL Validation

The first attempt at validation was done on the WSDL files as they came from the zip file. Each WSDL file was put through the validation process and the following errors were the result.

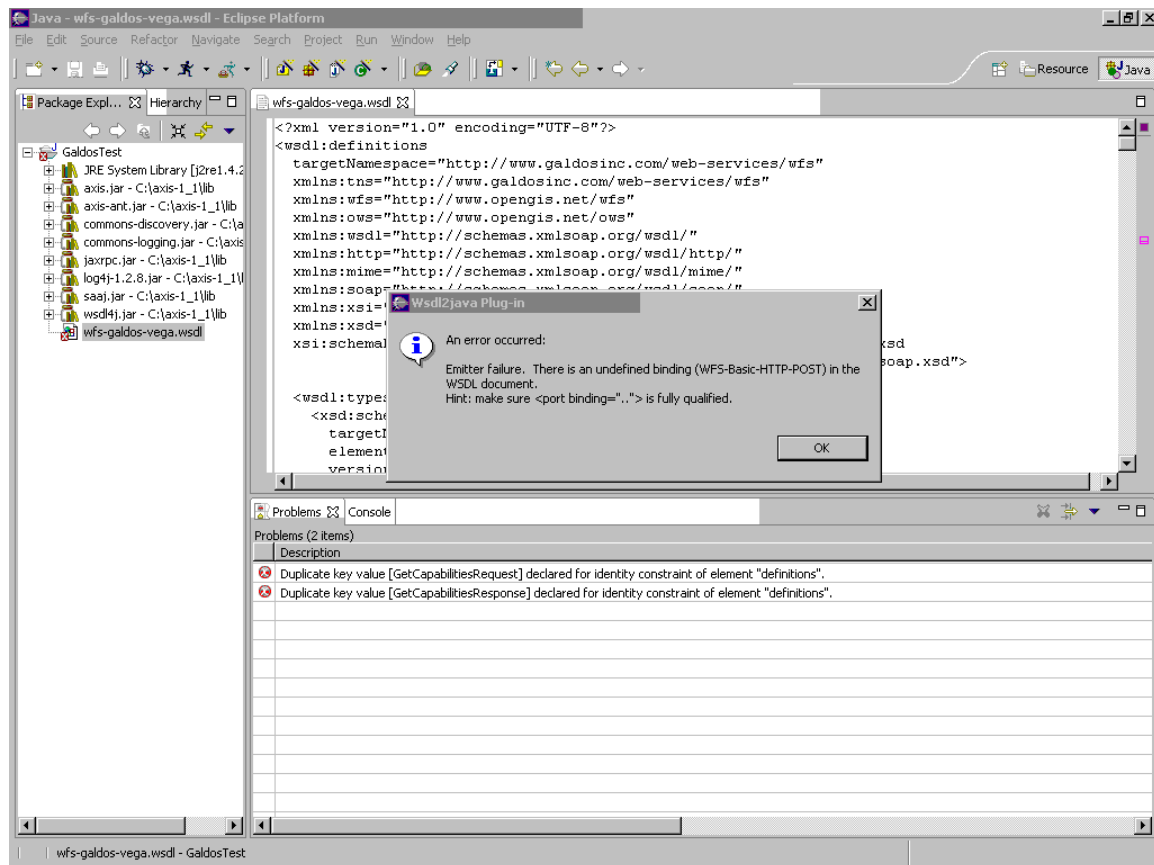


Next all of the WSDL files were combined in order to avoid any problems due to imports. Also, the validation tool reported errors on the documentation tags in the WSDL so these were removed. This step removed all but two of the errors, and these two errors are accounted for in the Issues section of the README.txt that accompanied the zip file.



11.5.2 Stub Generation

Stub generation was attempted but failed.



11.5.3 Using the Stub

Stub testing was not possible.

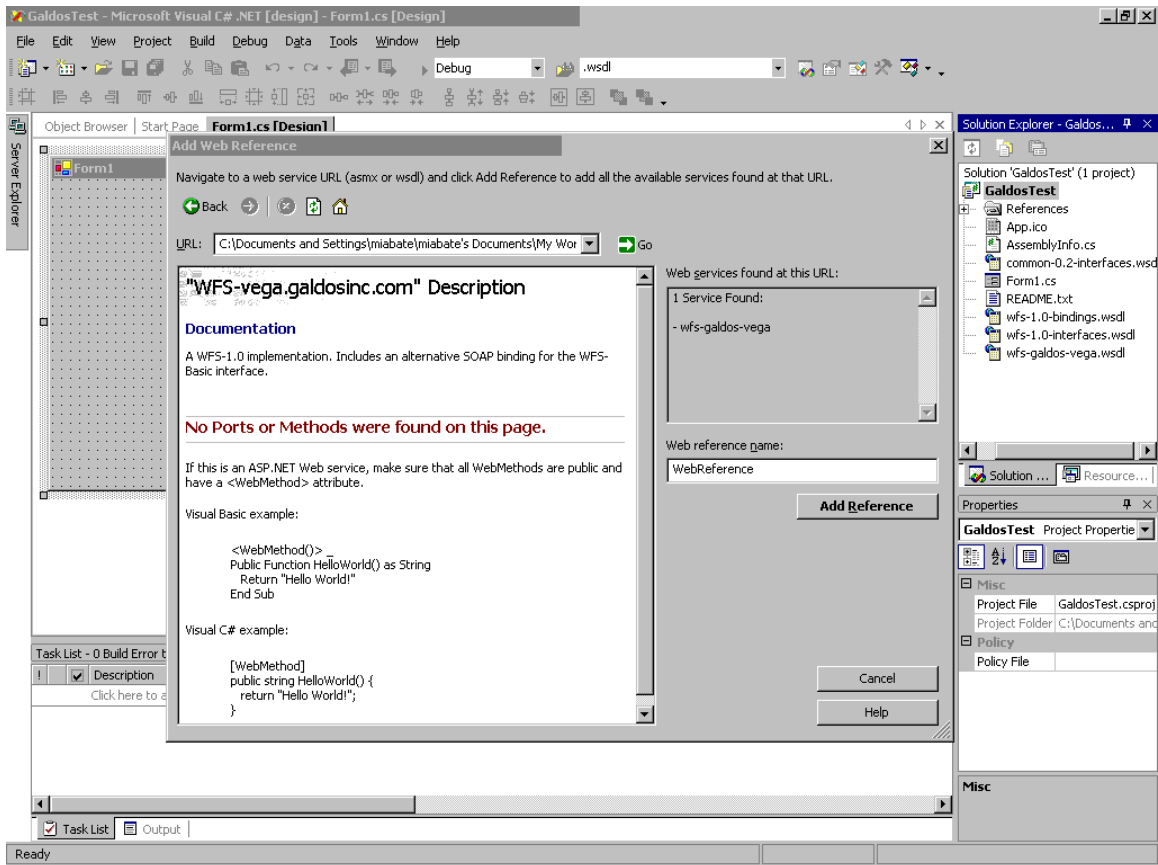
11.5.4 Conclusions

Most of the errors initially reported were due to problems with import statements, once all files were combined only two errors were reported. Stub generation failed but fixing the two errors exposed in validation may allows stubs to be generated.

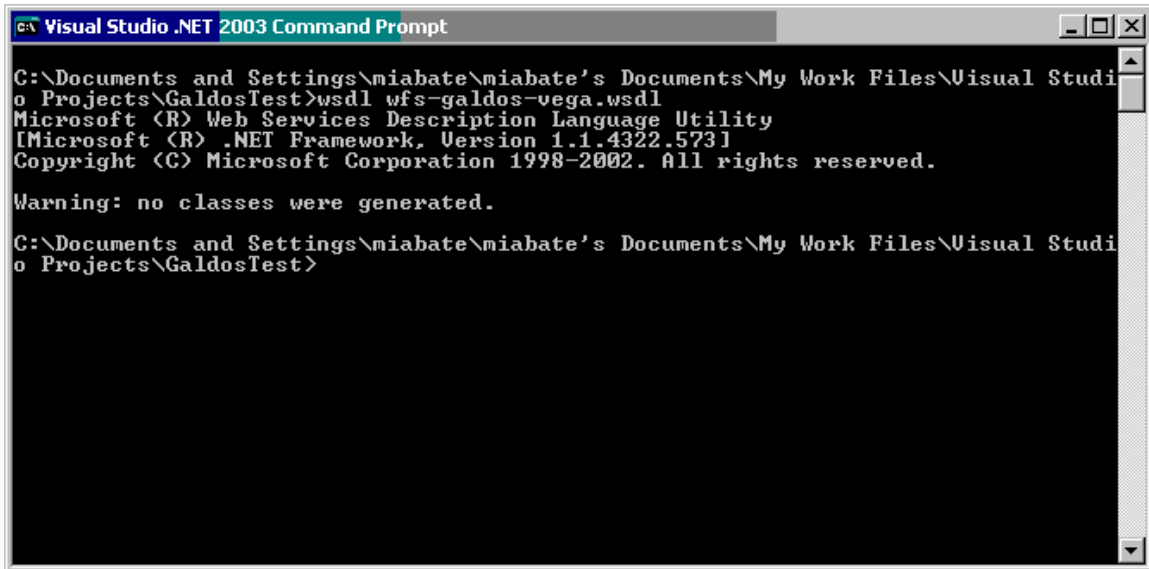
11.6 .Net Use of WFS

11.6.1 WSDL Validation

First the Add Reference command was used on the WSDL. VS .NET was not able to understand the WSDL.



The command line tool wsdl.exe was used on the WSDL as well. It reported back that no classes were generated, probably due to the fact that it could not understand the WSDL.



```
Visual Studio .NET 2003 Command Prompt
C:\Documents and Settings\miabate\miabate's Documents\My Work Files\Visual Studi
o Projects\GaldosTest>wsdl wfs-galdos-vega.wsdl
Microsoft (R) Web Services Description Language Utility
[Microsoft (R) .NET Framework, Version 1.1.4322.5731]
Copyright (C) Microsoft Corporation 1998-2002. All rights reserved.

Warning: no classes were generated.

C:\Documents and Settings\miabate\miabate's Documents\My Work Files\Visual Studi
o Projects\GaldosTest>
```


Next Add Reference was used on the combined WSDL file. This time VS .NET was able to understand the file but reported errors that would not allow it to generate stubs. The errors reported were:

The document at the url file:///C:/Documents and Settings/miabate/miabate's Documents/My Work Files/Visual Studio Projects/GaldosTest/wfs-galdos-vega.wsdl was not recognized as a known document type.

The error message from each known type may help you fix the problem:

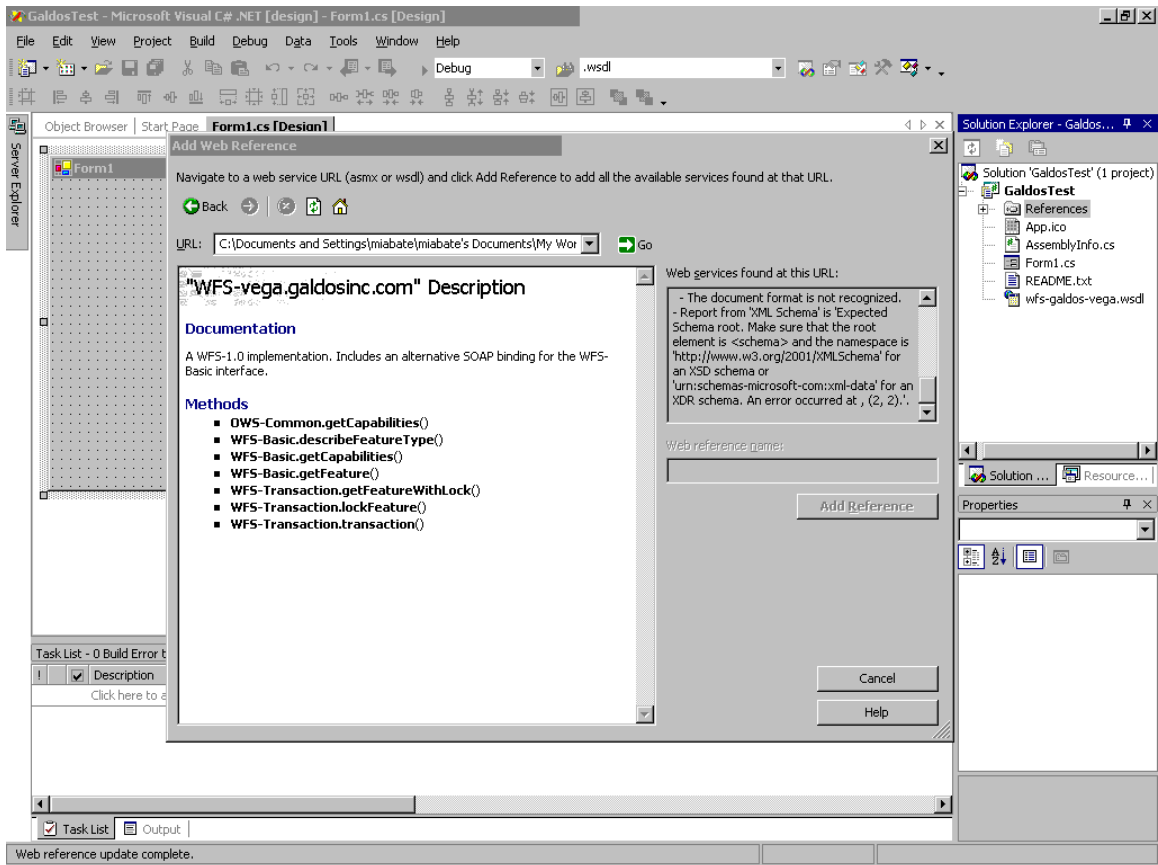
- Report from 'WSDL Document' is 'There is an error in XML document (141, 4).'

- More than one message named 'GetCapabilitiesRequest' was specified. Each message must have a unique name.

- Report from 'DISCO Document' is 'Discovery document at the URL file:///C:/Documents and Settings/miabate/miabate's Documents/My Work Files/Visual Studio Projects/GaldosTest/wfs-galdos-vega.wsdl could not be found.'

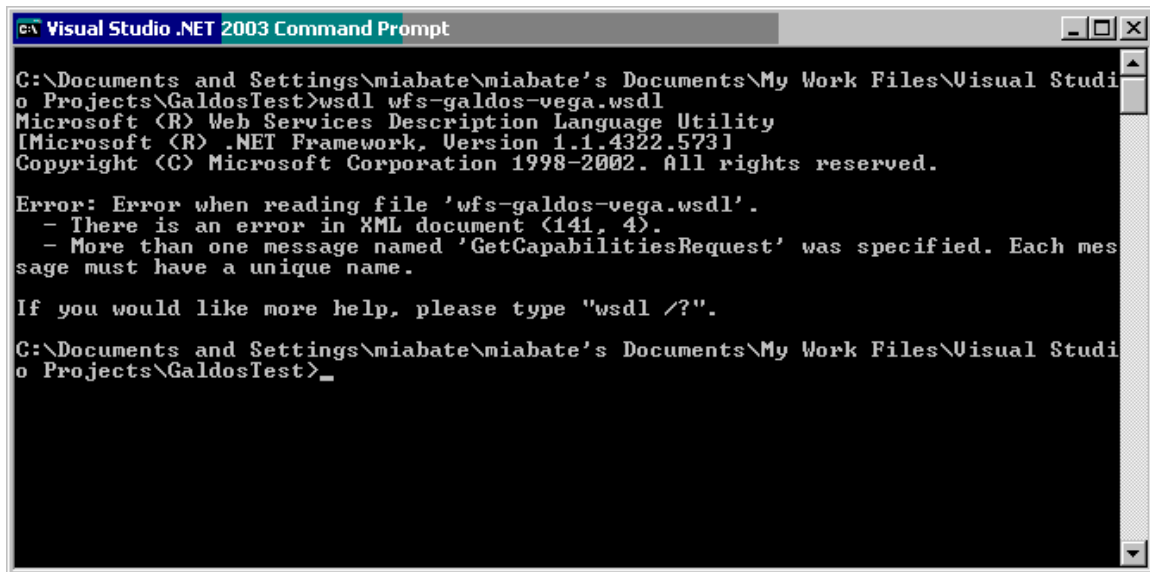
- The document format is not recognized.

- Report from 'XML Schema' is 'Expected Schema root. Make sure that the root element is <schema> and the namespace is 'http://www.w3.org/2001/XMLSchema' for an XSD schema or 'urn:schemas-microsoft-com:xml-data' for an XDR schema. An error occurred at , (2, 2).'



11.6.2 Stub Generation

Stub generation was attempted on the combined WSDL file using wsdl.exe. This attempt resulted in errors that are documented in the Issues section of the README.txt that accompanies the zip file.



```

e:\ Visual Studio .NET 2003 Command Prompt
C:\Documents and Settings\miabate\miabate's Documents\My Work Files\Visual Studi
o Projects\GaldosTest>wsdl wfs-galdos-vega.wsdl
Microsoft (R) Web Services Description Language Utility
[Microsoft (R) .NET Framework, Version 1.1.4322.5731
Copyright (C) Microsoft Corporation 1998-2002. All rights reserved.

Error: Error when reading file 'wfs-galdos-vega.wsdl'.
- There is an error in XML document (141, 4).
- More than one message named 'GetCapabilitiesRequest' was specified. Each mes
sage must have a unique name.

If you would like more help, please type "wsdl /?".

C:\Documents and Settings\miabate\miabate's Documents\My Work Files\Visual Studi
o Projects\GaldosTest>_

```

11.6.3 Using the Stub

Stub testing was not possible.

11.6.4 Conclusions

VS .NET was not able to understand the WSDL in the multiple file format. Once the WSDL files were combined VS .NET was able to understand the WSDL but then found errors that would not allow it to create stubs. If the errors are resolved then .NET stub generation may be possible. The errors that were reported are documented in the README.txt file that accompanies the zip file the WSDL came in.

12 MapInfo - Report on WSDL WMS

12.1 Overview

This report reviews testing of the Ionic OWS2 WMS WSDL contribution. The files from the folder "*Ionic WSDL, 4th it.zip*" posted to the OGC Web Services, Phase 2 section of the <http://portal.opengeospatial.org>

12.2 Steps

4. Validation of WSDL
5. Generation of stubs to call services
6. Use of stubs in a program to test services

12.3 Tools

- Eclipse 3.0.0 (<http://www.eclipse.org/platform>)
- WSDL/SOAP Validation Plug-in for Eclipse (<http://www.eclipse.org/wsvt/>)
- WSDL2Java (<http://ws.apache.org/axis>)
- Visual Studio .NET 2003 Architect Edition
- XMLSPY 5 Professional Edition

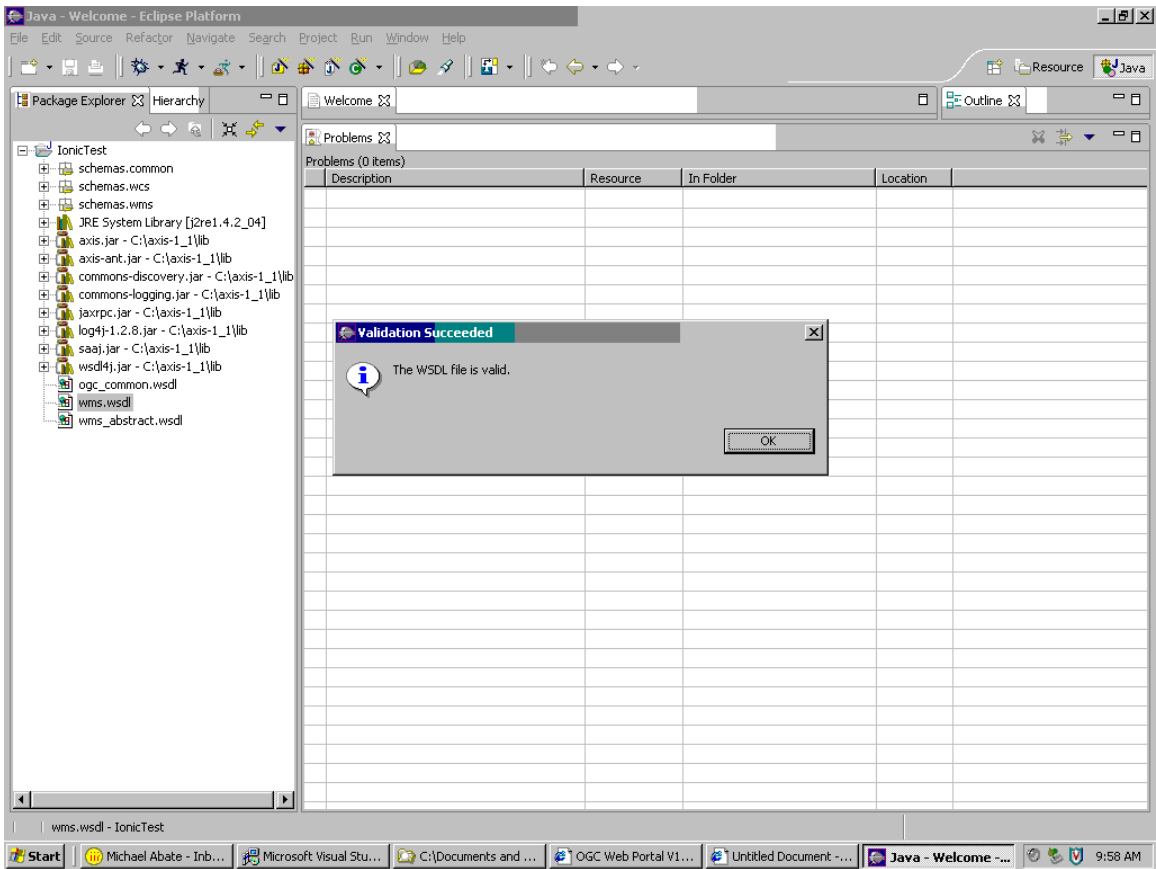
12.4 Notes

- Validation was done using the WSDL validation plug-in for Eclipse
- Generation of Java stubs was attempted using WSDL2Java

12.5 Java Use of WMS WSDL

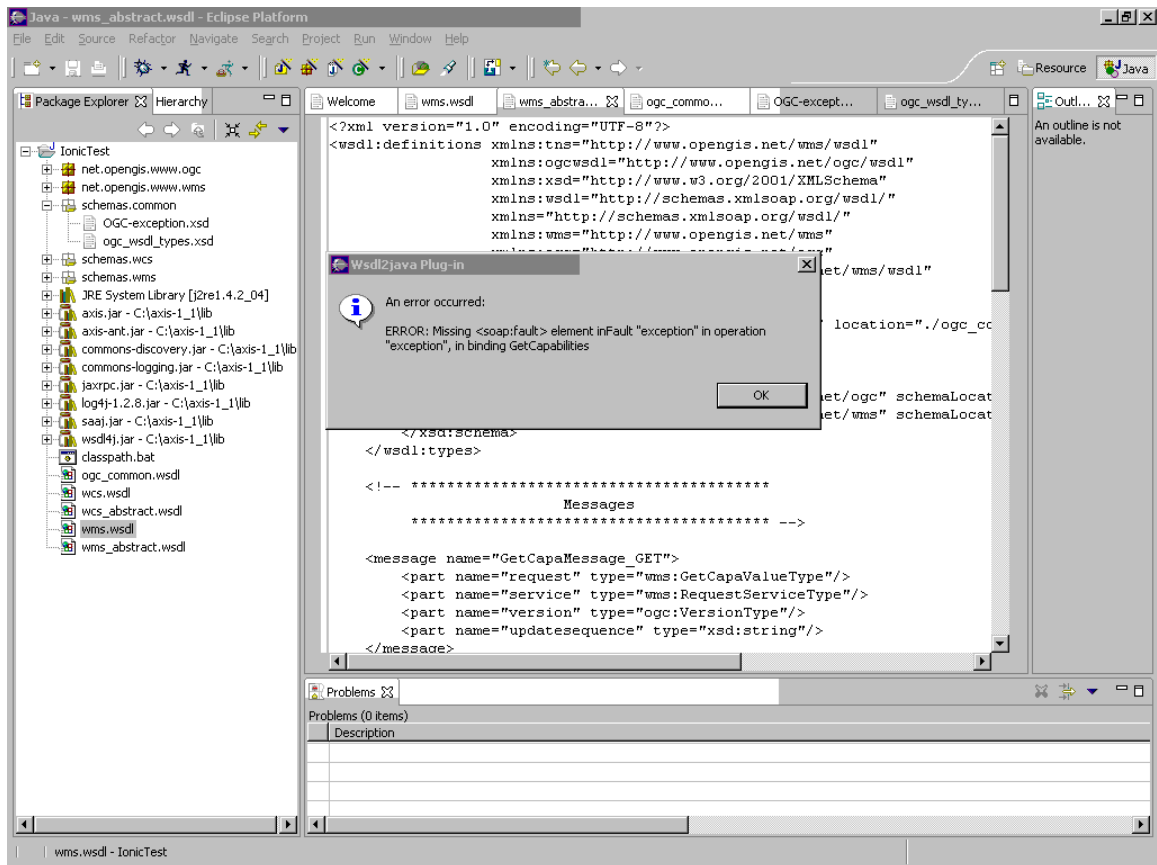
12.5.1 WSDL Validation

Validation of the succeed with the files straight from the zip archive that was downloaded from <http://portal.opengeospatial.org/>

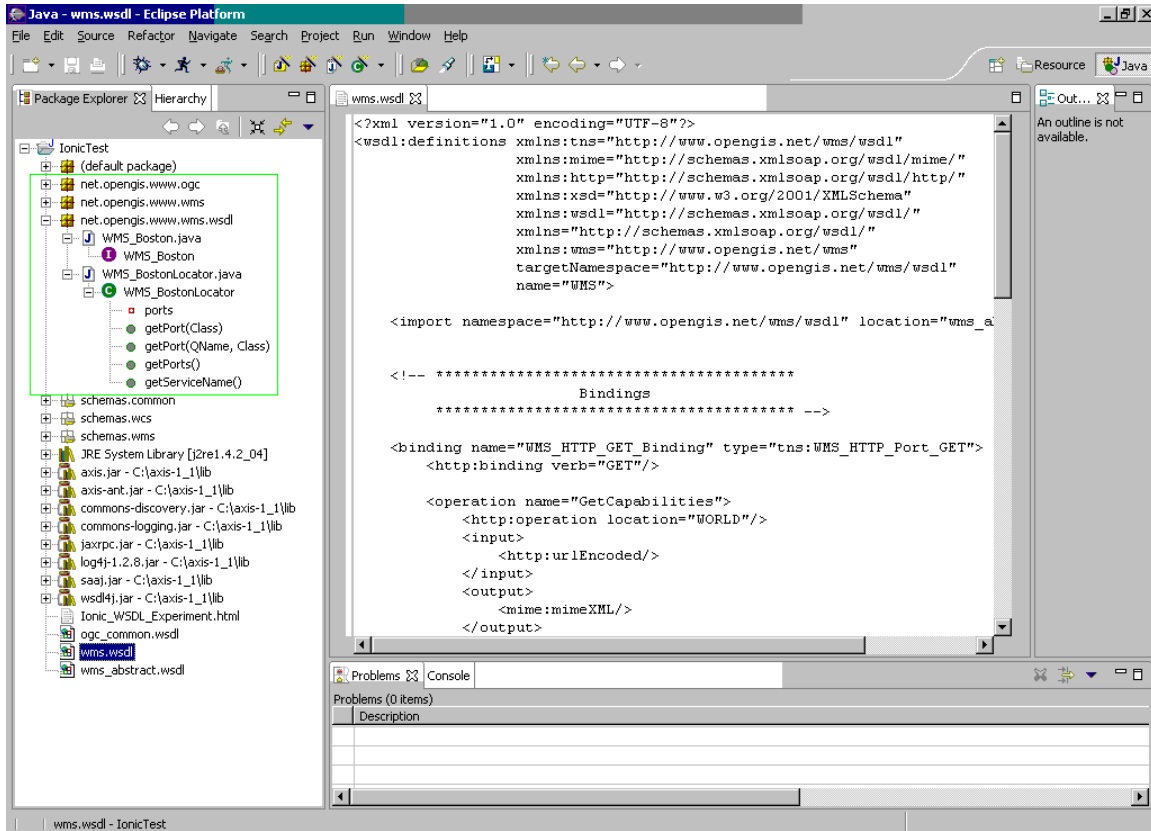


12.5.2 Stub Generation

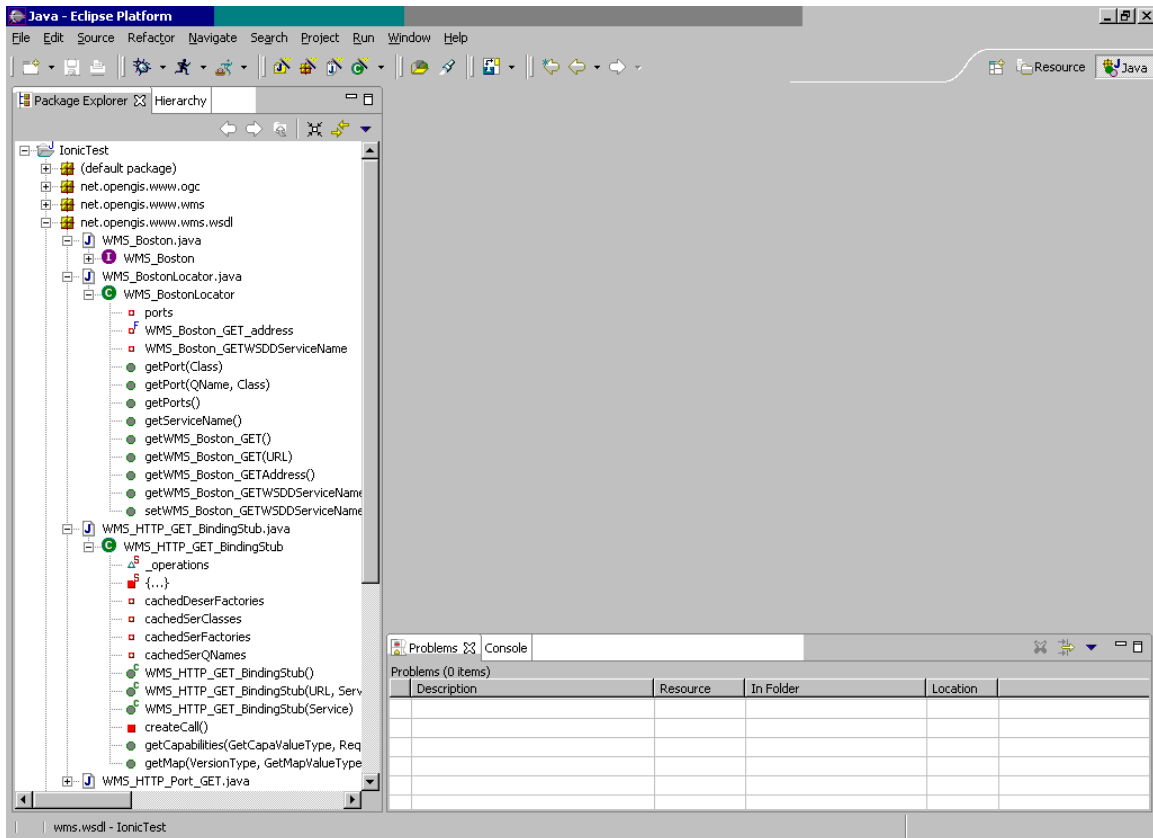
WSDL2Java was not able to generate Java stubs based on the WSDL documents. This error message seems odd since the WSDL documents did validate. Taking into consideration importing problems the *wms.wsdl* and *wms_abstract.wsdl* files were combined but generation met with the same error even though the combined file passed validation as well. (Note: adding *ogc_common.wsdl* to the combined file might have fixed the problem but was not completed due to validation errors that were produced when combining the file was attempted)



The fault element that caused stub generation to fail was removed and stub generation was attempted again. This time the generator reported that stub generation had succeeded. Further investigation revealed that only code for the types that the service uses were generated. Code to communicate with the service had not been generated. Only type classes were generated because the WSDL only contained HTTP services, no SOAP services.

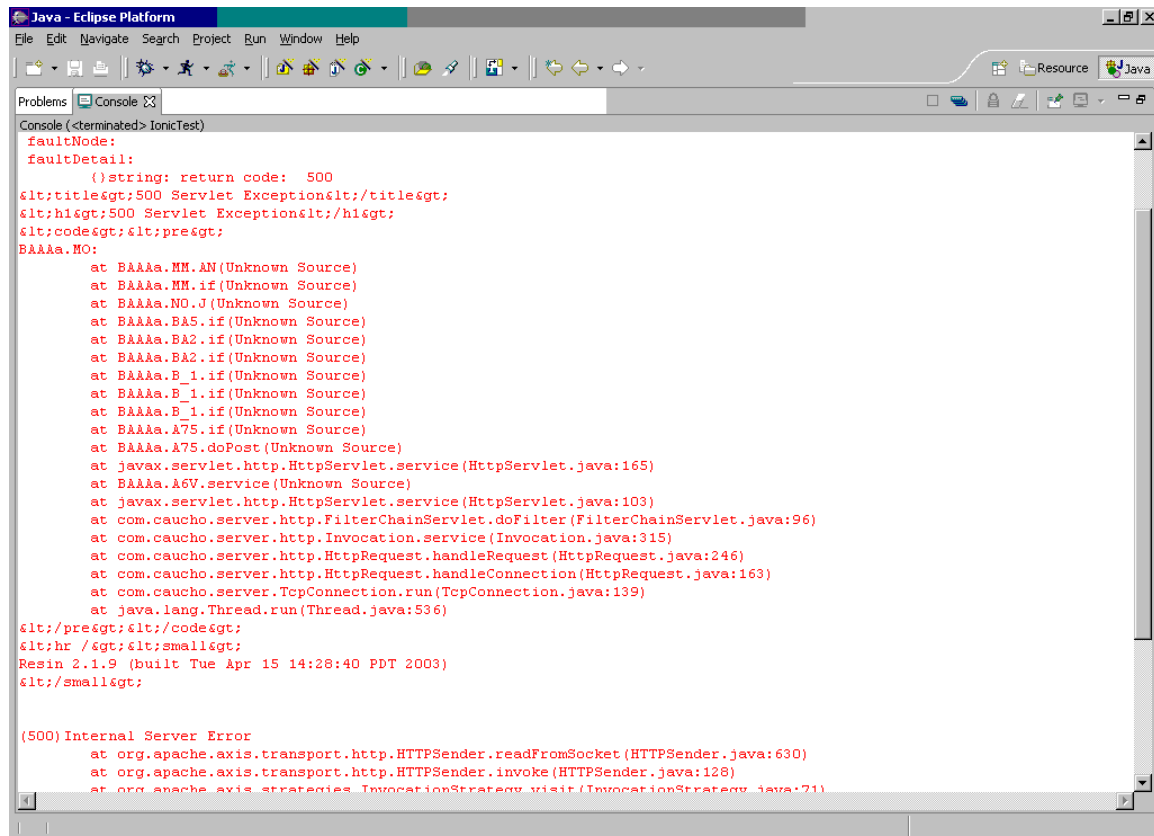


One of the HTTP services was converted to a SOAP service with SOAP bindings and stub generation was attempted. This type generation created code to communicate with the services.



12.5.3 Using the Stubs

The WSDL only specified HTTP services so it was expected that an attempt to communicate with the services using SOAP stubs would result in error.



```

Java - Eclipse Platform
File Edit Navigate Search Project Run Window Help
Problems Console
Console (<terminated> IonicTest)
faultNode:
faultDetail:
  ({}string: return code: 500
<title>500 Servlet Exception</title>
<h1>500 Servlet Exception</h1>
<code><pre>
BAAAA.MO:
  at BAAAA.MM.AN(Unknown Source)
  at BAAAA.MM.if(Unknown Source)
  at BAAAA.NO.J(Unknown Source)
  at BAAAA.BA5.if(Unknown Source)
  at BAAAA.BA2.if(Unknown Source)
  at BAAAA.B_1.if(Unknown Source)
  at BAAAA.B_1.if(Unknown Source)
  at BAAAA.B_1.if(Unknown Source)
  at BAAAA.A75.if(Unknown Source)
  at BAAAA.A75.doPost(Unknown Source)
  at javax.servlet.http.HttpServlet.service(HttpServlet.java:165)
  at BAAAA.A6V.service(Unknown Source)
  at javax.servlet.http.HttpServlet.service(HttpServlet.java:103)
  at com.caucho.server.http.FilterChainServlet.doFilter(FilterChainServlet.java:96)
  at com.caucho.server.http.Invocation.service(Invocation.java:315)
  at com.caucho.server.http.HttpRequest.handleRequest(HttpRequest.java:246)
  at com.caucho.server.http.HttpRequest.handleConnection(HttpRequest.java:163)
  at com.caucho.server.TcpConnection.run(TcpConnection.java:139)
  at java.lang.Thread.run(Thread.java:536)
</pre></code>
<hr /><small>
Resin 2.1.9 (built Tue Apr 15 14:28:40 PDT 2003)
</small>

(500) Internal Server Error
  at org.apache.axis.transport.http.HTTPSender.readFromSocket(HTTPSender.java:630)
  at org.apache.axis.transport.http.HTTPSender.invoke(HTTPSender.java:128)
  at org.apache.axis.strategies.InvocationStrategy.visit(InvocationStrategy.java:71)

```

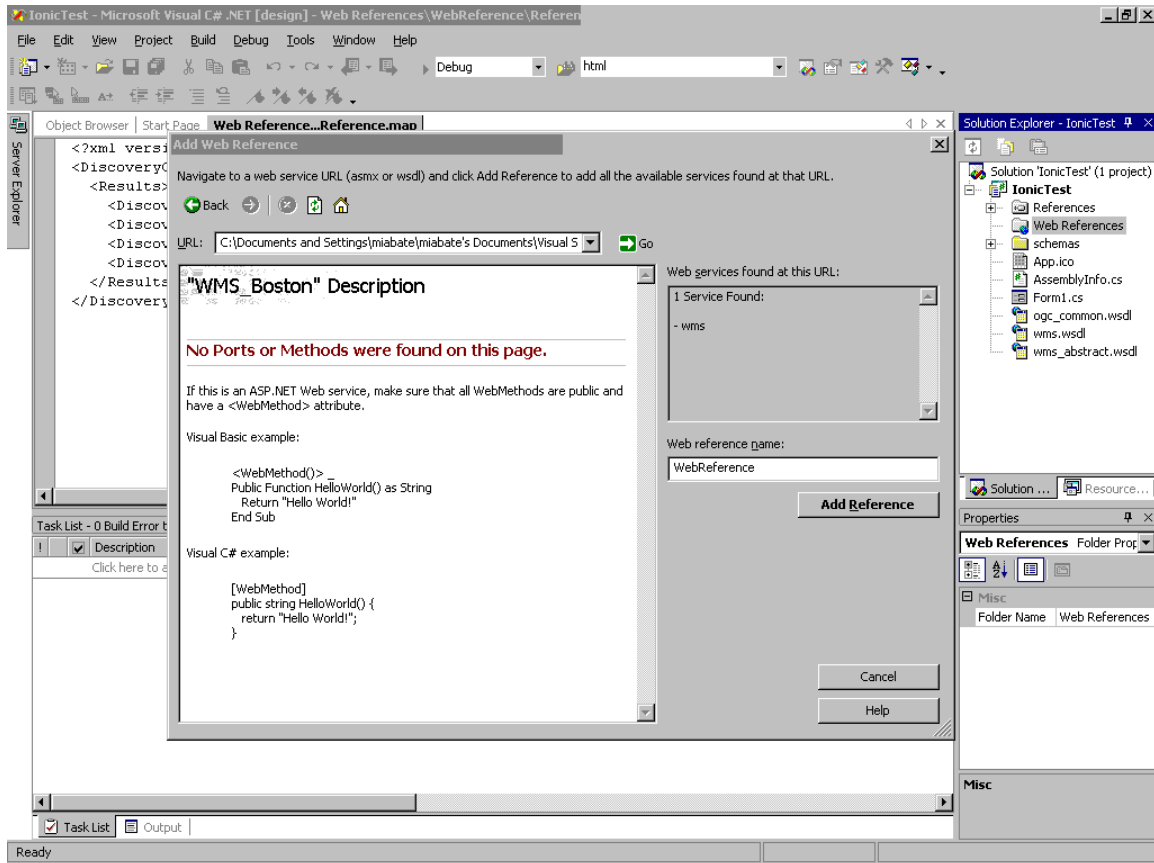
12.5.4 Conclusions

The WSDL documents validated without any need for modification but then WSDL2Java failed to generate Java stubs based on the documents. By removing the element that was causing stub generation to fail it was possible to generate classes for the types that the service used but it was still not possible to generate code that would be used to communicate with the service. After converting a HTTP service to a SOAP service stub generation succeed but the operation was not supported on the server end. The failure of the SOAP stubs was expected and is reasonable since the WSDL specified HTTP services and the address provided for the stub to communicate would be expecting HTTP communication. All that is required for the WSDL is a SOAP service on the server end and a modification to account for the error that was reported in the first stub generation attempt.

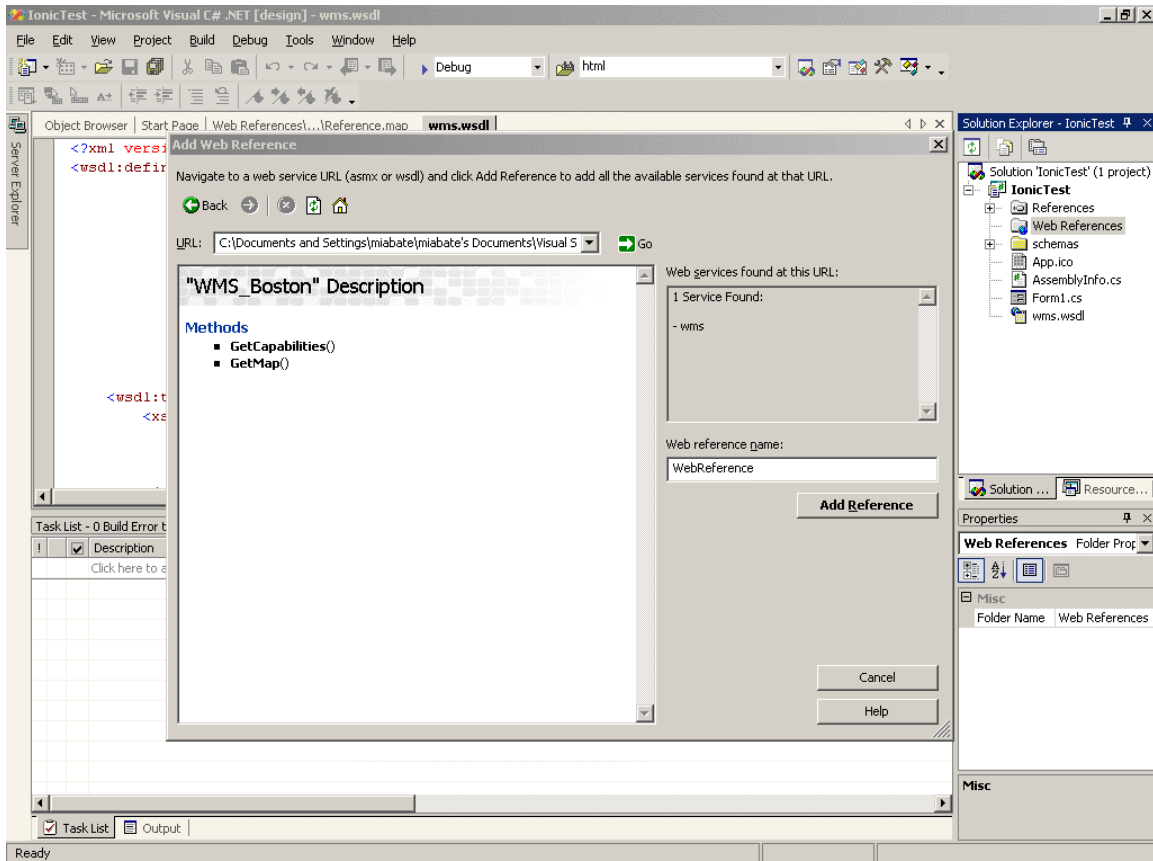
12.6 .Net Use of WMS WSDL

12.6.1 WSDL Validation

First an attempt to add a Web Reference based on the supplied WSDL documents was made. VS .NET was unable to validate the information to determine the methods that should be supplied.

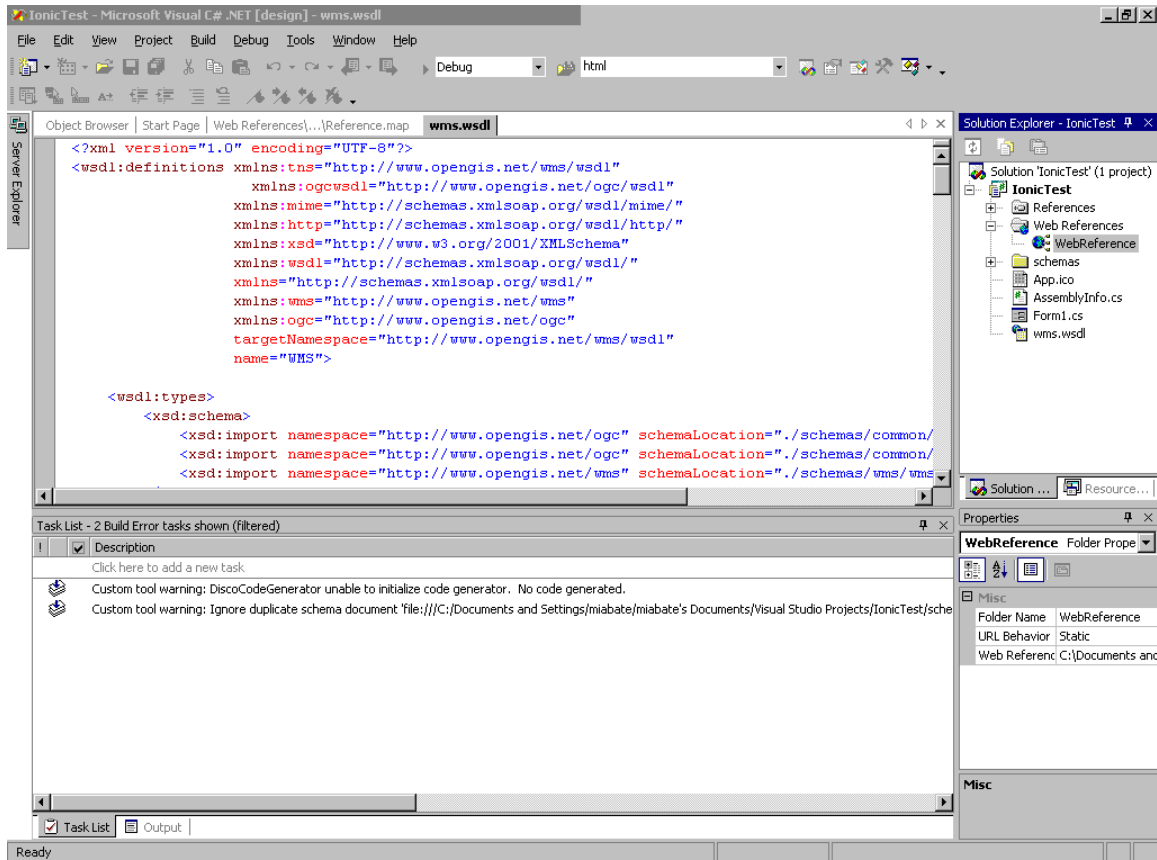


Next the files *ogc_common.wsdl*, *wms.wsdl*, and *wms_abstract.wsdl* were combined to avoid errors due to imports. With the combined file VS .NET was able to determine the methods the service exposed.



12.6.2 Stub Generation

Using the combined file that passed validation in the previous step an attempt to import the web service was made. VS .NET was not able to generate .NET stubs to work with the service. There were no “errors” but two “warnings” generated. The first warning (**Custom tool warning: DiscoCodeGenerator unable to initialize code generator. No code generated.**) seems to indicate that an error occurred somewhere and due to the error code could not be generated



Stub generation was also attempted using the command line tool wsdl.exe. Using wsdl.exe on the multiple wsdl files that were posted resulted in errors.

```
e:\ Visual Studio .NET 2003 Command Prompt
30/04/2004 12:00 PM      8,504 IonicTest.csproj
03/05/2004 08:55 AM      1,803 IonicTest.csproj.user
30/04/2004 11:43 AM          903 IonicTest.sln
29/04/2004 01:02 PM      7,809 Ionic_WSDL_Experiment.html
03/05/2004 09:47 AM      <DIR>      obj
28/04/2004 03:29 PM          770 ogc_common.wsdl
03/05/2004 09:48 AM      <DIR>      schemas
03/05/2004 09:47 AM      <DIR>      test
03/05/2004 09:47 AM      <DIR>      Web References
29/04/2004 12:15 PM      2,566 wms.wsdl
29/04/2004 12:17 PM      3,498 wms_abstract.wsdl
      11 File(s)      32,573 bytes
      7 Dir(s)      12,428,492,800 bytes free

C:\IonicTest>wsdl wms.wsdl
Microsoft (R) Web Services Description Language Utility
[Microsoft (R) .NET Framework, Version 1.1.4322.573]
Copyright (C) Microsoft Corporation 1998-2002. All rights reserved.

Error: Element portType named WMS_HTTP_Port_GET from namespace http://www.opengis.net/wms/wsdl is missing.

If you would like more help, please type "wsdl /?".

C:\IonicTest>
```

The wsdl files were then combined and wsdl.exe was used again, this time only warnings were generated.

```
e:\ Visual Studio .NET 2003 Command Prompt

C:\IonicTest>wsdl wms.wsdl
Microsoft (R) Web Services Description Language Utility
[Microsoft (R) .NET Framework, Version 1.1.4322.573]
Copyright (C) Microsoft Corporation 1998-2002. All rights reserved.

Warning: one or more operations were skipped.
Warnings were encountered. Review generated source comments for more details.

Writing file 'C:\IonicTest\WMS_Boston.cs'.

C:\IonicTest>_
```

The warnings generated from wsdl (shown below) indicate that there is a problem with the input types of the web service methods.

```

Microsoft Development Environment [design] - WMS_Boston.cs*
File Edit View Debug Tools Window Help
wms.wsdl | ogc_common.wsdl | wms_abstract.wsdl | WMS_Boston.cs*
WMS_Boston
WMS_Boston()

//-----
// <autogenerated>
// This code was generated by a tool.
// Runtime Version: 1.1.4322.573
//
// Changes to this file may cause incorrect behavior and will be lost if
// the code is regenerated.
// </autogenerated>
//-----

//
// This source code was auto-generated by wsdl, Version=1.1.4322.573.
//
using System.Diagnostics;
using System.Xml.Serialization;
using System;
using System.Web.Services.Protocols;
using System.ComponentModel;
using System.Web.Services;

/// <remarks/>
// CODEGEN: The operation 'GetCapabilities' from namespace 'http://www.opengis.net/wms/wsd1' was ignored.
// No input MIME formats were recognized.
// CODEGEN: The operation 'GetMap' from namespace 'http://www.opengis.net/wms/wsd1' was ignored.
// No input MIME formats were recognized.
[System.Diagnostics.DebuggerStepThroughAttribute()]
[System.ComponentModel.DesignerCategoryAttribute("code")]
public class WMS_Boston : System.Web.Services.Protocols.HttpPostClientProtocol {

    /// <remarks/>
    public WMS_Boston() {
        this.Url = "http://webservices.ionicsoft.com/worldData/map/";
    }
}

```

It was possible to generate C# classes for the types that the web service uses by using the command line tool xsd.exe

12.6.3 Using the Stub

Stub testing was not possible.

12.6.4 Conclusions

VS .NET was able to understand the WSDL documents when they were combined into one file. When trying to generate the .NET stubs VS encountered an error that caused code generation to fail. By using the command line tool wsdl.exe and the wsdl combined in to one file it was possible to see the errors that caused stub generation to fail. Though stub generation failed it was possible to generate classes for the types that the service uses.

13 Galdos Simple WSDL/SOAP Experiment for WFS

13.1 Experiment Description

This short report summarizes a simple experiment to test the use of WSDL and SOAP for the OGC Web Feature Service.

The WSDL document used is contained in Appendix A of this report.

The WSDL document was generated using the XMLSpy 2004 Enterprise Edition which provides functionality for editing and analyzing WSDL documents. This was reported on in an earlier document posted to the project (OWS 2.0) portal at OGC.

This report uses the previous WSDL document to automatically generate and send SOAP messages to a Galdos WFS supporting SOAP.

The experiment was conducted as follows:

4. The WSDL document was edited to use the service location <http://dali.galdosinc.com:8081/wfs-soap/services/WFSSOAPService>. This is the location of a Galdos test WFS used for CITE.
5. The WSDL document for WFS (testWSDL.wsdl) was read into XMLSpy 2004 Enterprise Edition and the WSDL validated by XMLSpy. Note that XMLSpy is rather forgiving of WSDL errors relative to other tools that we have tried. This will be reported on in another report.
6. XMLSpy 2004 provides the ability to generate and send SOAP messages based on processing the WSDL document.

The objective of the experiment was then to create and send SOAP messages from the XMLSpy SOAP tool to the Galdos WFS listed in the WSDL document.

This experiment was quite successful. The following operations were tested successfully as document in this report:

1. GetCapabilities
2. DescribeFeature
3. GetFeature
4. GetFeatureWithLock

The architecture of the experiment is shown in Figure 1.

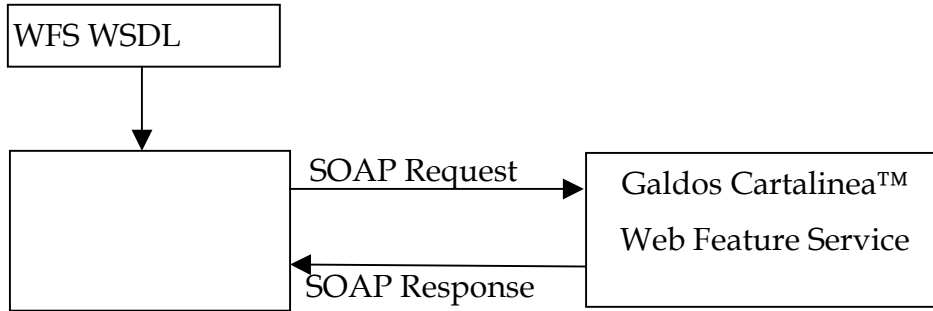


Figure 1. Experiment Architecture

13.2 Experiment Results

This section summarizes the results of the experiment in a series of annotated screen shots.

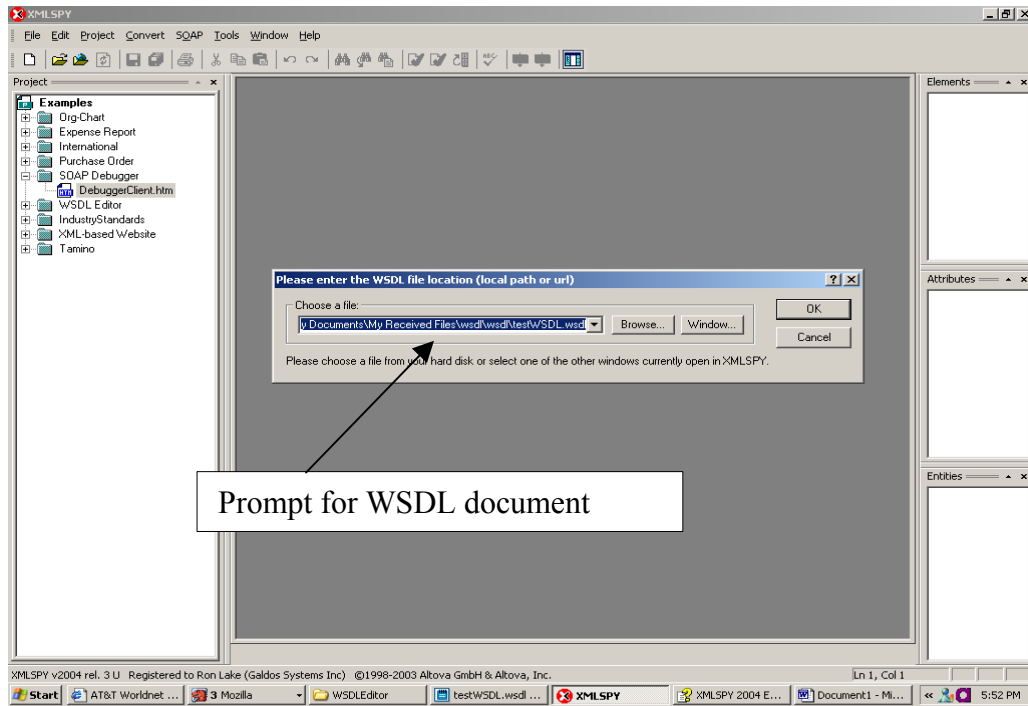


Figure 1. Selecting the WSDL Document for WFS

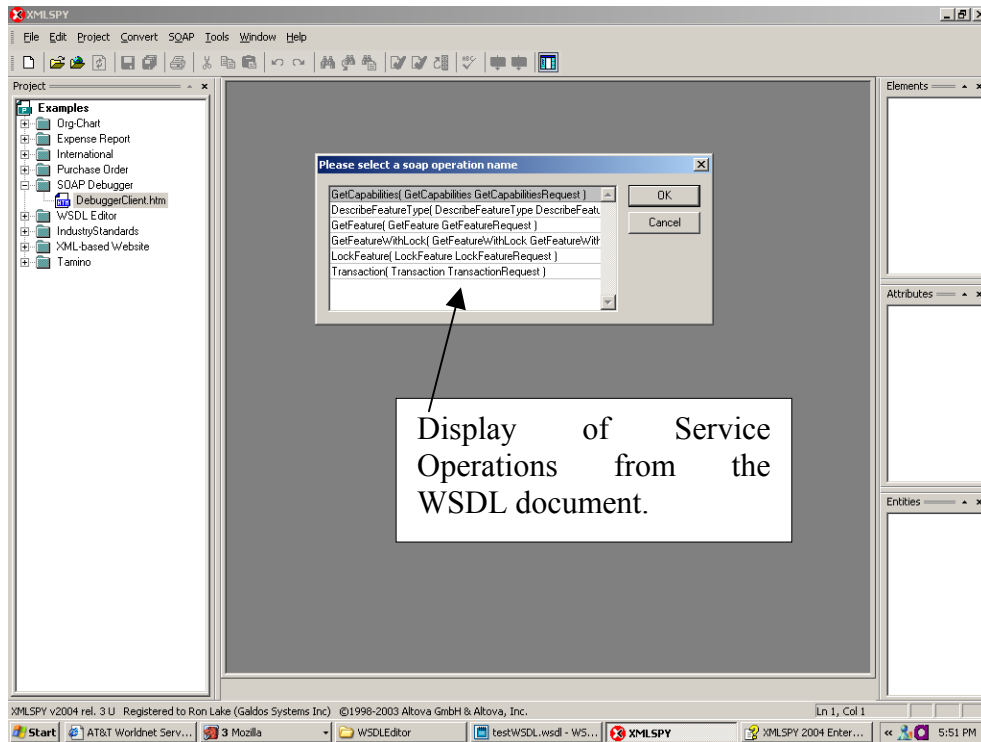


Figure 2. Selecting an operation based on the WSDL Document

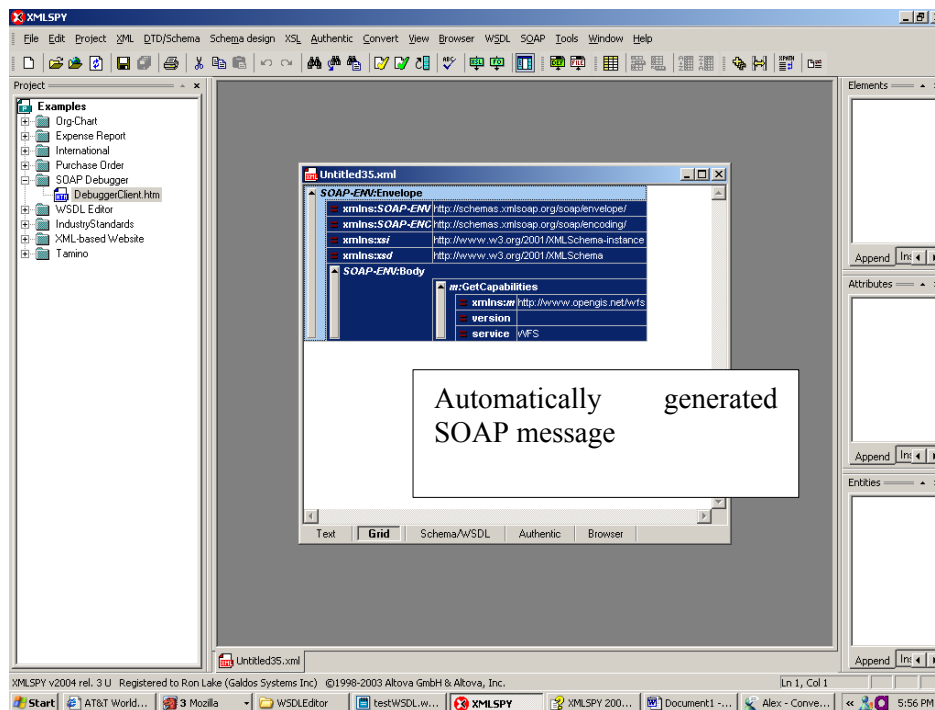


Figure 3. The Generated SOAP message for Get Capabilities

Now we have XMLSpy send the message to the service referenced by WSDL.

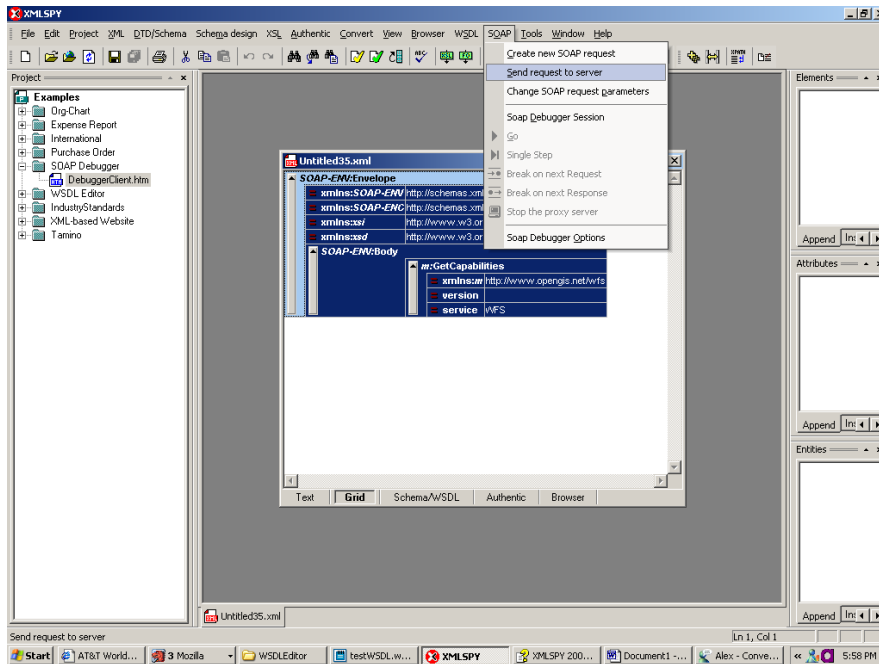


Figure 4. Sending the SOAP message via XMLSpy

Note that in this case we do not need to do anything to the message at all.

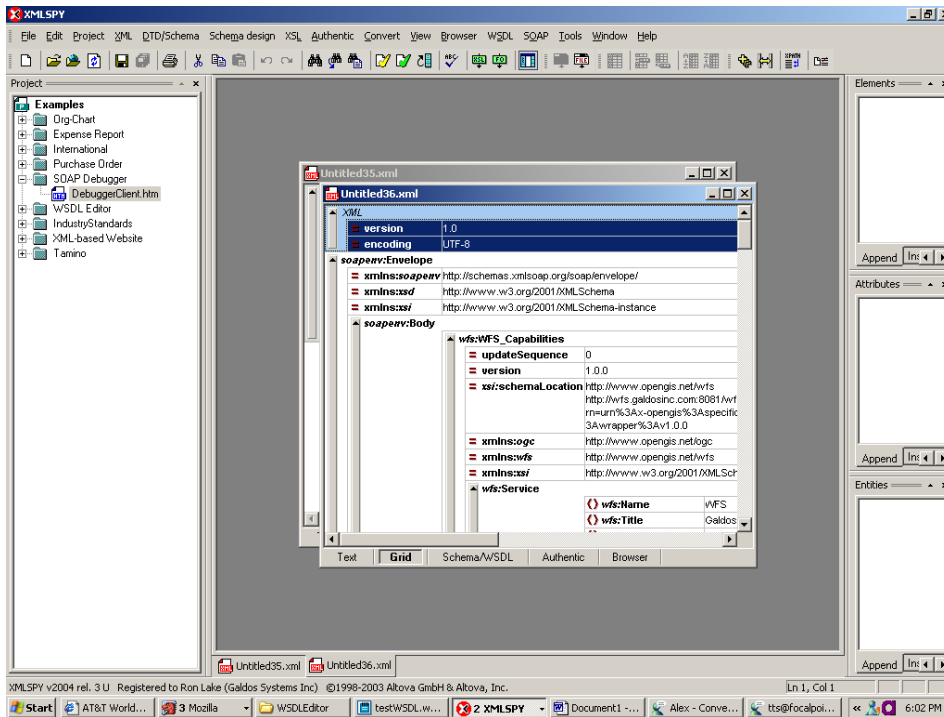


Figure 5. The Response Capabilities Document

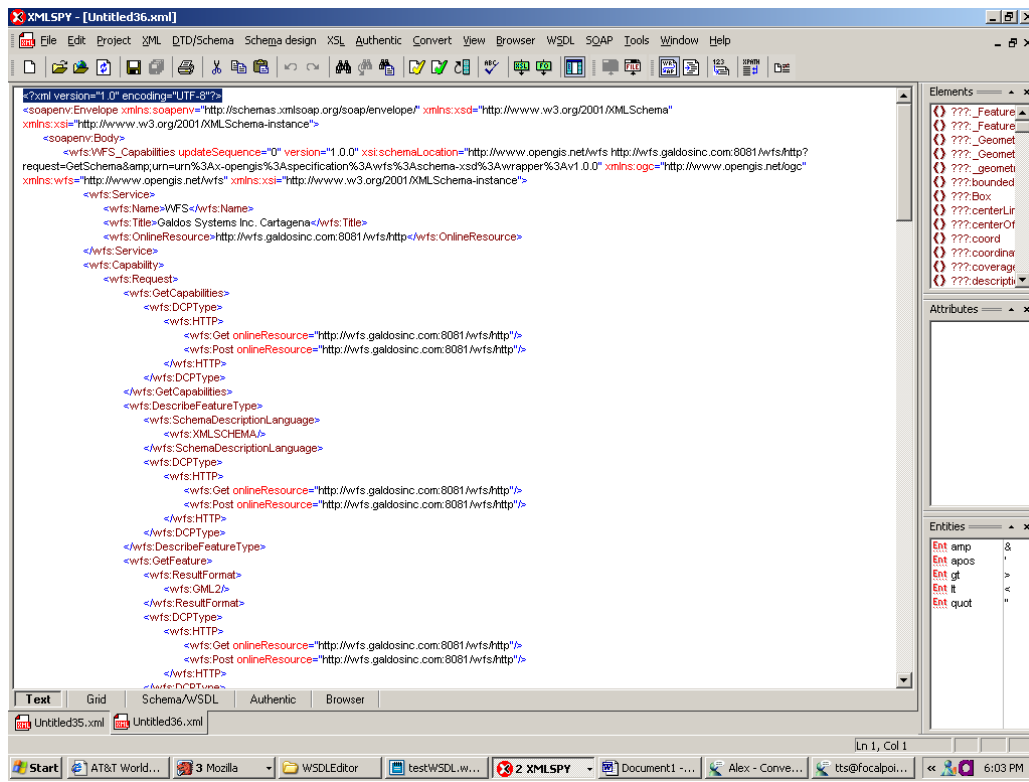


Figure 6. The Capabilities Document – Oops no SOAP in here !!

Note that we have requested a capabilities document via SOAP that does not reference a SOAP “DCP” at all!!

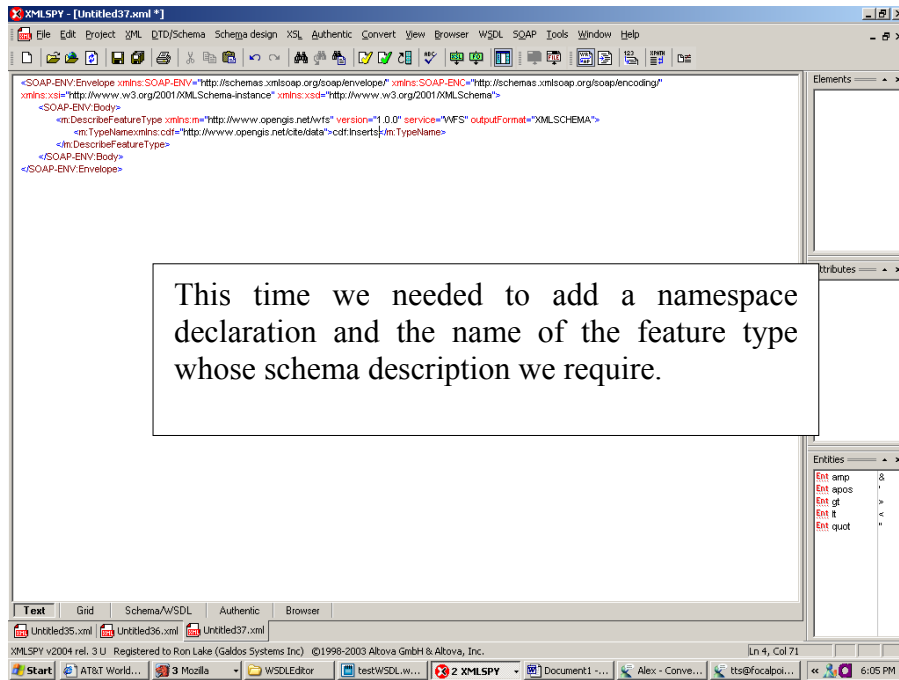


Figure 7. The Describe Feature Request generated from WSDL

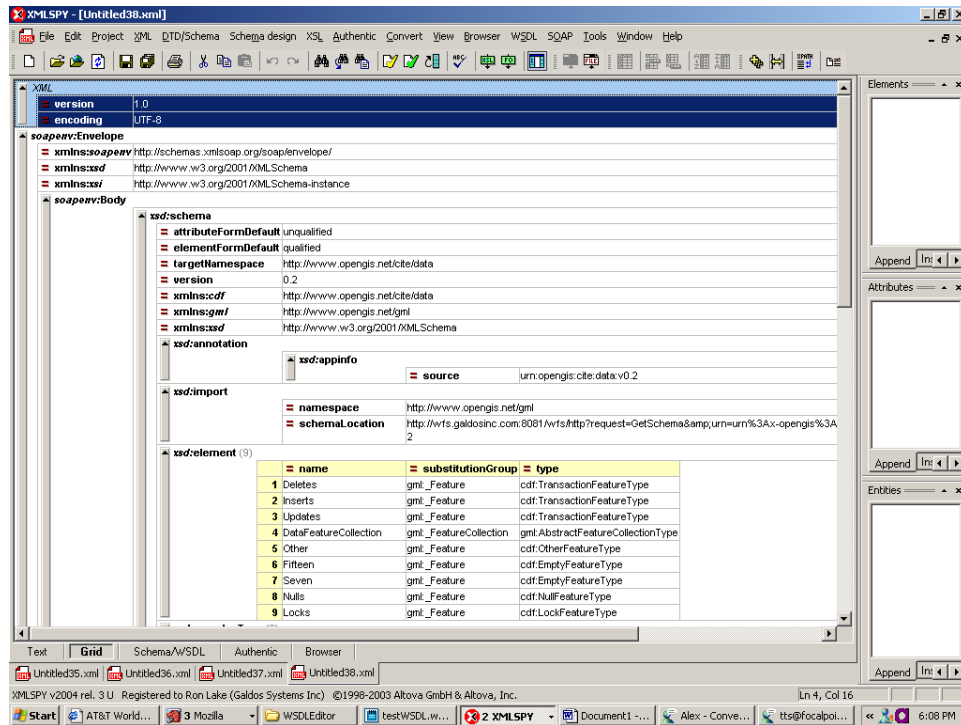


Figure 8. The Describe Feature Response

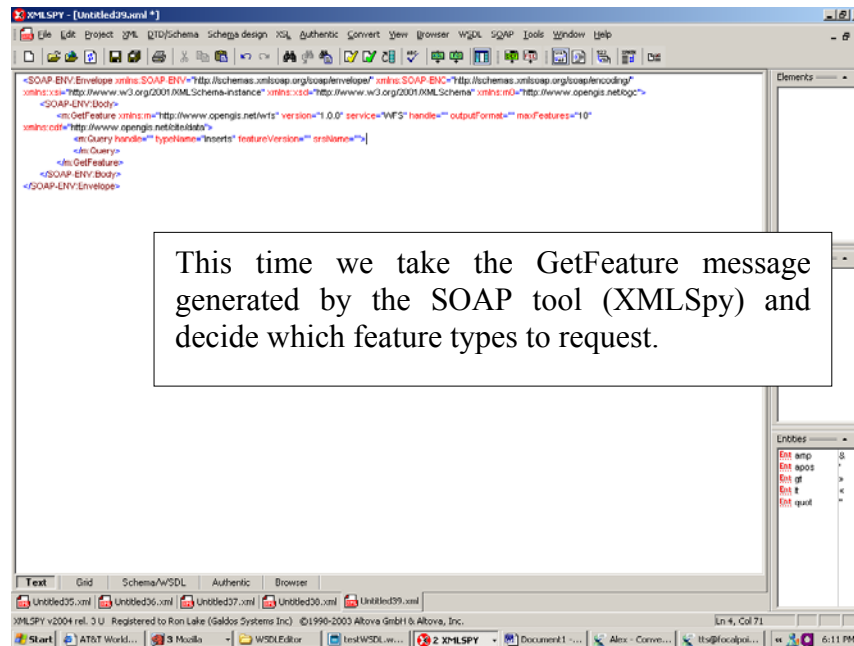


Figure 9. GetFeature Request with Filter Part Removed – “Inserts” Feature Type

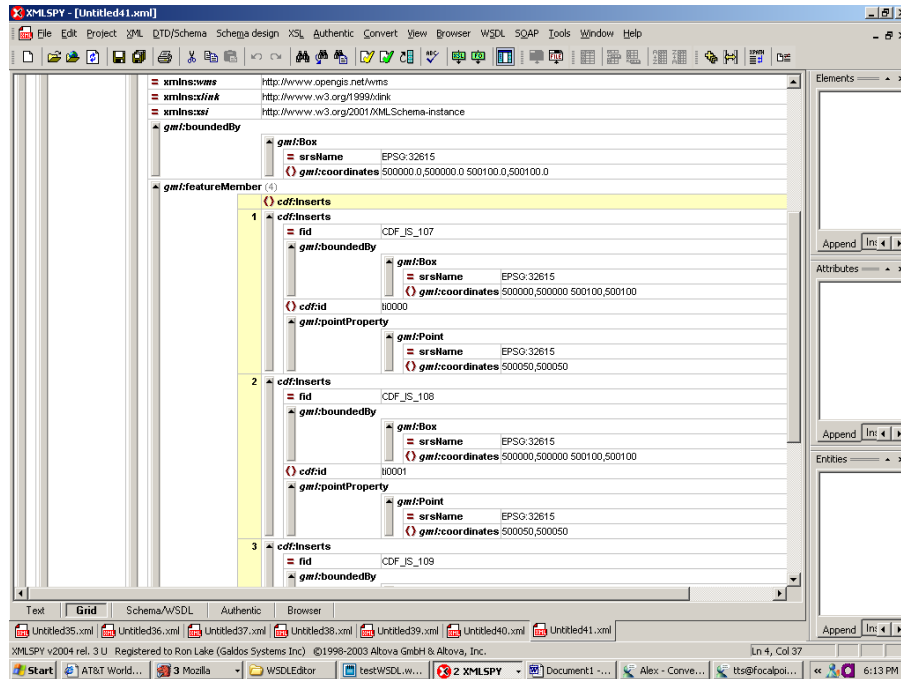


Figure 10. GetFeature Response

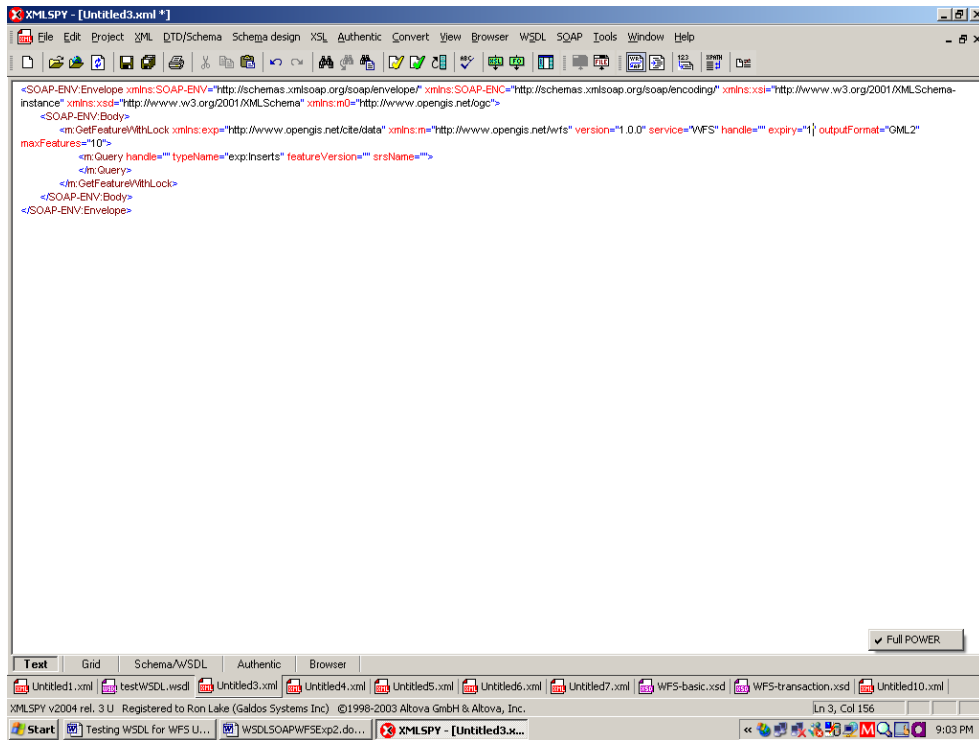


Figure 11. GetFeatureWithLock Request

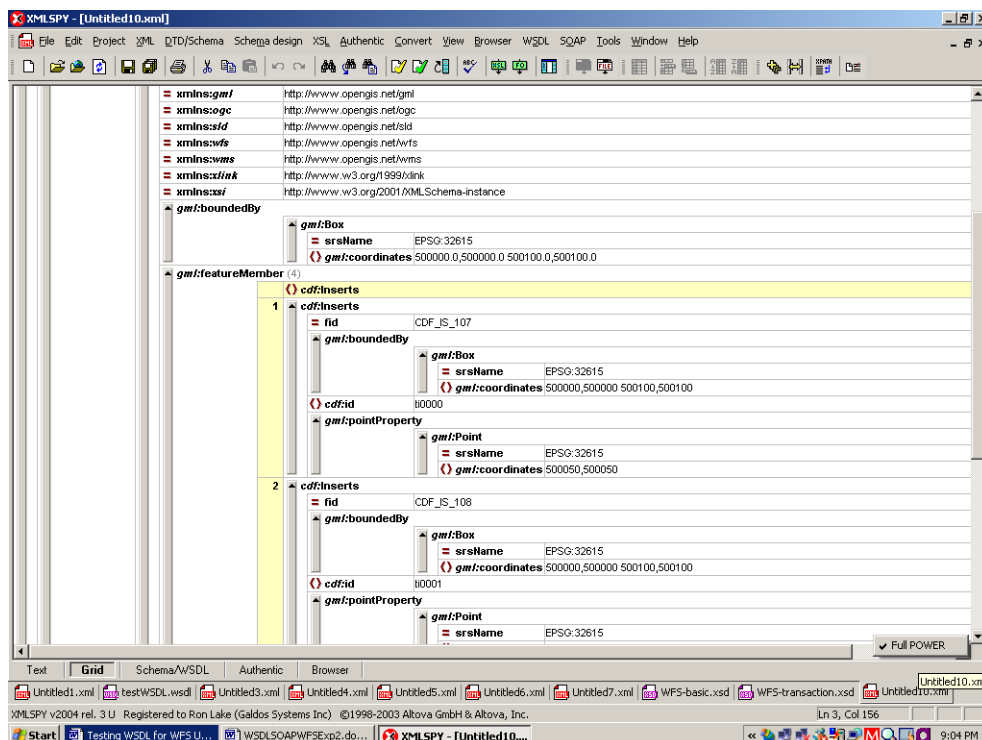


Figure 12. GetFeatureWithLock Response

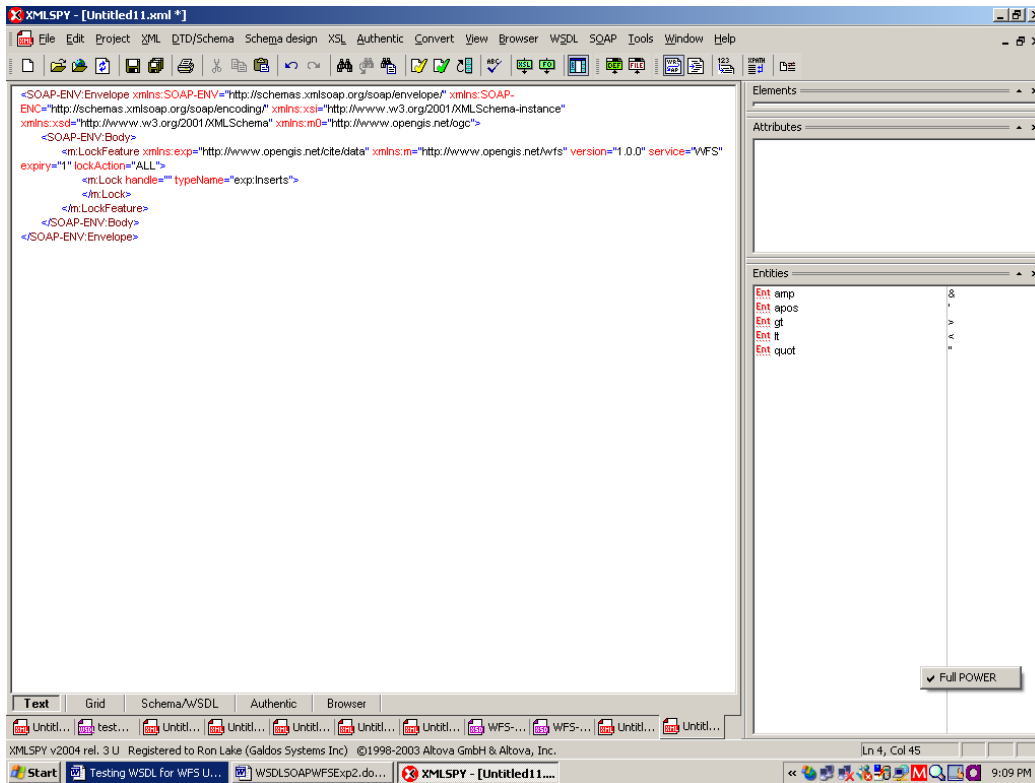


Figure 13. LockFeature Request

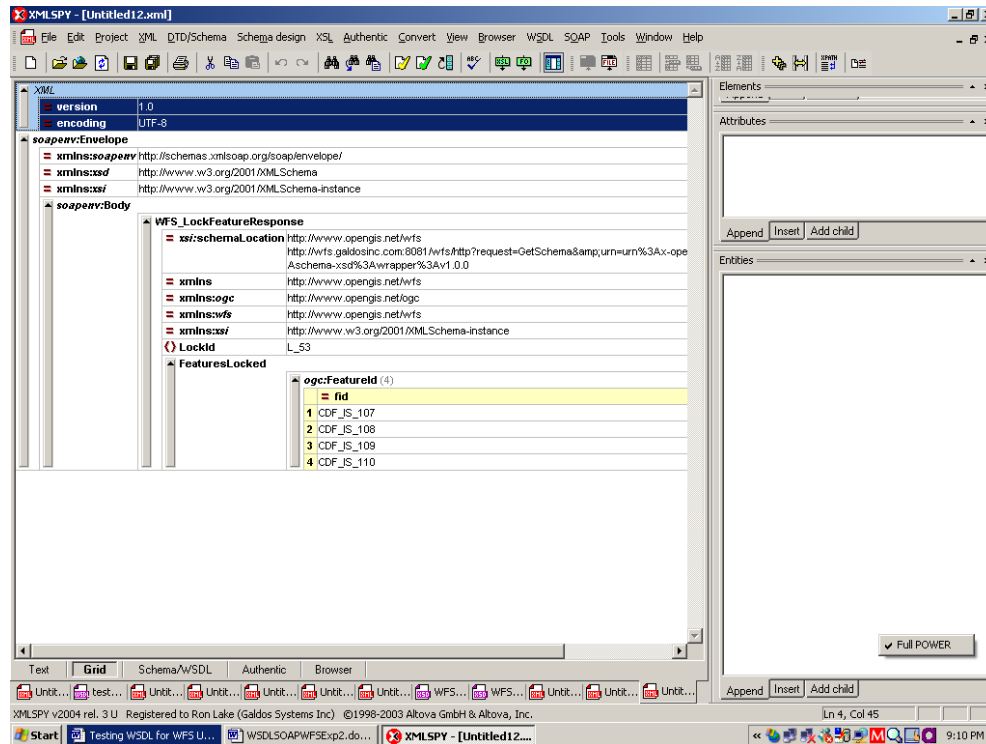


Figure 14. LockFeature Response

14 Ionic Experiment Summary

14.1 Overview

The experiments described below cover the validation and use of the WSDL documents for WCS, WMS, WFS to generate stubs and register with UDDI services.

14.2 Current implementations of the specifications

Below is a description of the tools used, and how they seem to follow the specifications.

14.2.1 Tools

The tools used for these experiments are :

- Soapclient (<http://www.soapclient.com>)
- Mindreef's Soapscope (<http://www.mindreef.com/>)
- Systinet Eclipse
plugin
(http://www.systinet.com/products/wasp_developer/overview)

- JDeveloper (<http://otn.oracle.com/products/jdev/>)
- XMLSpy (<http://www.xmlspy.com/>)

14.2.2 Tools compliance

14.2.2.1 KVP bindings

SoapClient is the only straightforward tool available, but far too permissive. It doesn't map the response to the WSDL response message; simply returns the raw response.

Systinet plugin is supposed to handle KVP bindings, but is very picky (check its description)

14.2.2.2 XML/POST bindings

No tools could be successfully tested with XML/POST binding.

14.2.2.3 XML/SOAP bindings

SoapClient supports SOAP bindings, but chokes on complex WSDL imports and schemas (esp. WCS and WFS).

XMLSpy is fairly trustworthy, although too permissive on some points. It doesn't seem to handle binary responses.

SoapScope is a good reference.

14.2.2.4 UDDI

JDeveloper supports UDDI browsing and querying, but fails to parse WSDL files with imported XSD schemas

Systinet plugin supports UDDI browsing and querying and successfully retrieves WSDL files to build stubs from them.

14.2.3 Reference implementations

There is a need for reference implementations of WSDL, SOAP and UDDI on which to rely, since the specifications are sometimes unclear (esp. WSDL).

None of the tools used in these experiments seem to completely support what they should. Some other tools should be tested, such as JAXRPC, Axis, Wasp.

14.3 Test Results

	KVP	XML
--	-----	-----

WCS - GetCapabilities - DescribeCoverage - GetCoverage	(tested using SoapClient) - OK - OK - NOK, check issue ## (CR on optional params)	SoapClient fails to read the schemas tested using SoapScope, XMLSpy - OK - OK - response comes back OK, BUT don't know how to map binary part in WSDL message (check issue #01)
WMS - GetCapabilities - GetMap	(tested using SoapClient) - response comes back OK, but we lack a WMS schema to define the WSDL message (check issue ##) - OK	SoapClient is able to read the schemas, but no POST implementation of a WMS is available to our knowledge.

15 Intergraph - WSDL/BPEL TIEs

The behavior of a web service is defined by a WSDL document. And the structure of each of the messages exchanged by a web service operation are generally defined within one or more XML schemas.

With the growing acceptance of web services, a number of tools have emerged which aid in the construction of web services and web service clients. Those tools offer facilities for constructing new WSDL documents; a typical tool also provides the ability to consume an existing WSDL document and its related schemas for the purpose of creating a web service client. And of course there are many tools for constructing and validating XML schemas.

The W3C has published and continues to maintain specifications (recommendations) that define XML, XML schemas, WSDL and related technologies. The tool vendors create their tools based on the W3C specifications. Of course, there are differences in appearance and method of operation among the tools on the market. However, one would expect these tools to arrive at the same interpretation of a particular WSDL file and its related schemas. In practice, however, we found that **not** to be the case. Although there certainly were occasions where the tools would agree on an issue with with a WSDL file or schema, there were also situations where the tools would not agree.

There are different possibilities as to why these tools may sometimes not agree on the status of a particular WSDL or schema file:

OGC 04-060r1

- The tool itself is flawed. XML, XML schemas and WSDL are relatively new to the world of computing and the tools have not yet reached a level of maturity.
- The W3C specification is "vague" with respect to a particular guideline and the tool vendors arrived at different interpretations.
- The tool stops short of implementing a particular aspect of the specification, failing to detect a problem.

Given imperfect tools, an implementer is faced with two choices:

- Convincing the vendor to fix the tool in a timely fashion, or
- Altering the WSDL and/or schemas to accommodate the tool.

For the OWS2 effort, we took both approaches. The Common Architecture group provided assistance to the IH4DS group which constructed BPEL Service Chains using the Oracle BPEL Designer and BPEL Process Manager. Oracle's BPEL tools are still in beta, and exhibited some shortcomings. Some of the problems that were identified Oracle was able to fix. And in some cases the BPEL tools indicated problems with our WSDL and/or schemas and we took actions fix them.

[As of this writing, we are still tracking down some schema related issues. As a workaround, we have developed simplified WSDL files that have no dependence on external schema files. All WSDL messages are defined with elements which are either XML simple types or a special 'anyType' type which can accommodate any XML element structure.]

During the course of the OWS2 effort, we explored a number of WSDL and schema validation test tools:

WSDL:

- XMLSpy Enterprise Edition (Altova)
- SoapScope (Mindreef)
- Oracle BPEL Designer

XML Schema:

- XMLSpy Enterprise Edition
- Schema Quality Checker (SQC) (IBM)
- XML Schema Validator (XSV) (University of Edinburgh)
- MSXML SP2 (Microsoft)