Open GIS Consortium Inc.

Date: 2003-01-20

Reference number of this OpenGIS® project document: **OGC 03-021**

Version: 0.1.18

Category: OpenGIS® Discussion Paper

Editor: J. Yutzler

Integrated Client for Multiple OGC-compliant Services

Architecture, Design, and Experience

Copyright notice

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.

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 Papers
Document stage:	Publicly Available
Document language:	English

Contents

i. Preface

The OpenGIS Consortium (OGC) is an international industry consortium of more than 220 companies, government agencies, and universities participating in a consensus process to develop publicly available geo-processing specifications. This Interoperability Program Report (IPR) provides an overview of the requirements, architecture, and design of Integrated Clients developed during the OGC Open Web Services Thread Set 2 (OWS 1.2) program. Additionally, this IPR includes a discussion of the experiences gained during the development of the integrated clients during the effort within the context of the OGC General Services Architecture with respect to consistency and completeness. This discussion is primarily intended to serve as an introduction to those undertaking the development of client services.

Suggested additions, changes, and comments on this draft report are welcome and encouraged. Such suggestions may be submitted by OWS 1.2 portal message, email message, or by making suggested changes in an edited copy of this document.

Changes made to this document can be tracked by Microsoft Word. If you choose to submit suggested changes by editing this document, please make your suggested changes with change tracking on.

ii. Submitting organizations

This Interoperability Program Report is being submitted to the OGC by the following organizations:

Laser-Scan, Inc.
Ionic Software

iii. Submission contact points

CONTACT	COMPANY	ADDRESS	PHONE/FAX	EMAIL
Jeff Yutzler	Laser-Scan Inc.	45635 Willow Pond Plaza, Sterling, VA 20164 USA	(703) 709-9306	jyutzler@lsiva.com
John Vincent	Intergraph Mapping and GIS Solutions	PO Box 6695 Huntsville, AL 35824-6695 USA	(256) 730-7767	jtvincen@intergraph.com
Chris Tucker	Ionic Enterprise, Inc.	PO Box 2635 Alexandria, VA 22301 USA	(703) 535-5973	tucker@ionicenterprise.com
Jerome Sonnet	Ionic Software, S.A.	Rue de Wallonie, 18 B-4460 Grâce-Hollogne Belgium	+32 4 364 0 364	jerome.sonnet@ionicsoft.com
Thomas M. Tuerke	Autodesk, Inc.	111 McInnis Parkway San Rafael, CA 94903 USA	(415) 507-5000	thomas.tuerke@autodesk.com

iv. Revision history

Date	Release	Author	Paragraph modified	Description
5 July 2002	0.0.1	Dibner	-	Initial version.
22 July 2002	0.0.2	Osborne	-	-
23 July 2002	0.0.3	Osborne	Sections 5 and 6	Added additional text to the overview and use case sections.
9 Sep 2002	0.1.4	Osborne	Section 8	Added Chris Tuckers comments. Added use case diagrams.
11 Nov 2002	0.1.5	Yutzler	Sections 5 and 6	Merged sections and

				added additional detail.
13 Nov 2002	0.1.6	Tuerke	Section 9	Added UI section
13 Nov 2002	0.1.7	Tuerke	Section 9	Added UI section
13 Nov 2002	0.1.87	Dibner	Section 8 (new)	Added outline and some text for software component / implementation section.
14 Nov 2002	0.1.98	Yutzler	All	Reviewed for content and style
153 Jan 2003	0.1.109	Yutzler	Sections 5.5 6– 9-9	Sections completely rewritten based on input from Dibner, Tuerke
14 Jan 2003	0.1.11	Yutzler	Annex A	Tie reports added
14 Jan 2003	0.1.12	Yutzler	All	Reviewed for content and style
14 Jan 2003	0.1.13	Yutzler	Annex A	Replaced section for Ionic with new text
14 Jan 2003	0.1.14	Dibner	Sections 10 and 11	Expanded section on test considerations, prepared TIE summary table, wrote concluding summary.
15 Jan 2003	0.1.15	Tuerke	Annex A	Formatting corrections; updated contact info. Fixed incorrect heading, some text.
15 Jan 2003	0.1.16	Dibner	TOC	Fixed glitch in header identification
16 Jan 2003	0.1.17	Dibner	Section 5.3	Fixed error in reference to Figure 7
19 Jan 2003	0.1.18	Dibner	Document	Reformatted – eliminate corruption causing crashes

v. Changes to the OpenGIS® Abstract Specification

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

Foreword

This document (OGC 03-021) is an Interoperability Program Report (IPR) that reflects work carried out during the OGC Web Services Initiative, Thread Set 2.

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. OGC Inc. shall not be held responsible for identifying any or all such patent rights.

This IPR is intended to be informative, and does not seek to modify any existing OGC specifications, nor create any new specifications. There are currently no annexes in the IPR., although some may be added in the future.

Introduction

This Interoperability Program Report (IPR) provides an overview of the general requirements, architecture, and design considerations of 'Integrated Clients' developed for the OGC Open Web Services Thread Set 2 (OWS 1.2) program. In addition, this IPR includes a discussion of the experiences gained during the development of the integrated clients during the effort within the context of the OWS 1.2 architecture with respect to consistency and completeness. This discussion is primarily intended to serve as an introduction to those undertaking the development of client services.

Within the context of this effort an integrated client is defined as a software application that provides common functionality for the discovery, retrieval, and handling of data from sources that fall into the following categories:

- Feature data (GML encoded vector data)
- Image data (raster)
- Sensor Web data (XML)

At the core of the integrated client concept is the requirement to provide a unified environment that allows a user to visualize, analyze, and/or edit data from all three of the above source categories simultaneously. In addition the integrated client may also support a persistent project file, so that a complex set of data layers – comprised of source, service chain, and portrayal information – can be stored and shared between applications and users.

This IPR will include integrated clients which utilize many or all of the following specifications: Web Registry Service (WRS), Web Map Server (WMS), Web Feature Server (WFS), Style Layer Descriptor (SLD), Style Management Service (SMS), Web Coverage Server (WCS), Coverage Portrayal Service (CPS), Image Archive Service (IAS), Sensor Planning Service (SPS), Web Notification Service (WNS), and Sensor Collection Service (SCS).

OGC Interoperability Program Report: The Integrated Client

1 Scope

This IPR describes the requirements, use cases, architectural and design considerations for the development of an integrated, multi-service client; and also discusses the experiences of OWS 1.2 participants in creating such clients. In fulfilment of these goals, the IPR includes:

A) Definitions of the common terms associated with the effort.

B) A discussion of the functional breakdown of the integrated client, and the OGC services that are related to each functional category.

C) A discussion of use cases for the integrated client, and how these use cases might take advantage of blending functionality across the functional categories and various OGC services.

D) A discussion of possible architectures for the integrated client, with a focus on both thick and thin client types.

E) A discussion of design issues and tradeoffs associated with the development of an integrated client, with respect to the similarities and differences between the OGC services.

F) And a discussion of the key accomplishments and lessons learned by the OGC members participating in the development on integrated client for the OWS 1.2 effort.

2 Reference Documents

The following documents contain provisions that serve as points of reference for portions of this report. For dated references, subsequent amendments to, or revisions of, any of these publications do not apply. However, parties to agreements based on the notions in this text are encouraged to investigate the possibility of applying the most recent editions of the documents indicated below.

The OpenGIS Abstract Specification Topic 12: OpenGIS Service Architecture (Version 4.2), Kottman, C. (ed.), OGC AS 12, September 2001.

OWS1 Registry Service, Martell, Richard (ed.), OGC 02-050r5, 19 August 2002.

OGC Contexts Definition and Encoding, Monie, D. & Humblet Jean-Philippe (eds.), OGC 02-066r1, 29 August 2002.

Web Feature Service Implementation Specification, Version 1.0.0, OGC Document #02-058, 19 September 2002.

Web Map Service Implementation Specification, Version 1.1.1, OGC Document #01-068r3, 16 January 2002.

Sensor Collection Service IPR, Version 0.7.0, OGC Document #02-028, 22 October 2002.

UDDI Experiment, OGC Document #02-054r1, 22 August 2002.

3 Terms and definitions

During previous OGC IP efforts, there have been discussions about client issues, but there has not been common concrete agreement on the definition of terms ‘client’, ‘thin client’, ‘thick client’, and ‘integrated client’ among others. For the purposes of this document, the following terms and definitions apply:

Client
A computer program which remotely accesses data or services from one or more servers.

Client-Server
A common form of distributed computing in which functionality is split between server software and client software. A client sends requests to a server, according to some protocol, asking for information to be returned and/or an action be performed, and the server responds.

Integrated Client
A client which unifies common service discovery, feature production, imagery exploitation, portrayal managment, and sensor web exploitation functionalities, and provides an environment for visualizing, analysing and/or editing data from these sources/services.

Interface
Named set of operations that characterize the behavior of an entity [OGC AS 12].

Operation
Specification of a transformation or query that an object may be called to execute [OGC AS 12].

Request
An invocation by a Client of an Operation.

Response
The result of an Operation, returned from a Server to a Client.

Service
Distinct part of the functionality that is provided by an entity through interfaces [OGC AS 12].

Server, Service Instance
A computer program that implements a service.

Thick Client
A computer program that is installed on a target platform, and is executed within a heavyweight operating system on that platform.

Thin Client
A computer program that runs a lightweight operating system and executes applications downloaded over a network.

4 Conventions

The following sections define the conventions used in this document.

4.1 Symbols (and abbreviated terms)

1D	One Dimensional
2D	Two Dimensional
3D	Three Dimensional
4D	Four Dimensional
API	Application Program Interface
COM	Component Object Model
CORBA	Common Object Request Broker Architecture
COTS	Commercial Off The Shelf
CPS	Coverage Portrayal Service
DCE	Distributed Computing Environment
DCP	Distributed Computing Platform
DCOM	Distributed Component Object Model
FPS	Feature Portrayal Service
GML	Geographic Markup Language
IDL	Interface Definition Language
ISO	International Organization for Standardization
OGC	Open GIS Consortium
PKI	Public Key Infrastructure
SRS	Spatial Reference System
SMS	Style Management Service
SLD	Styled Layer Descriptor
UML	Unified Modeling Language
WCS	Web Coverage Service
WFS	Web Feature Service
WMS	Web Map Service
XML	Extensible Markup Language

4.2 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 standard are described in the diagram below.

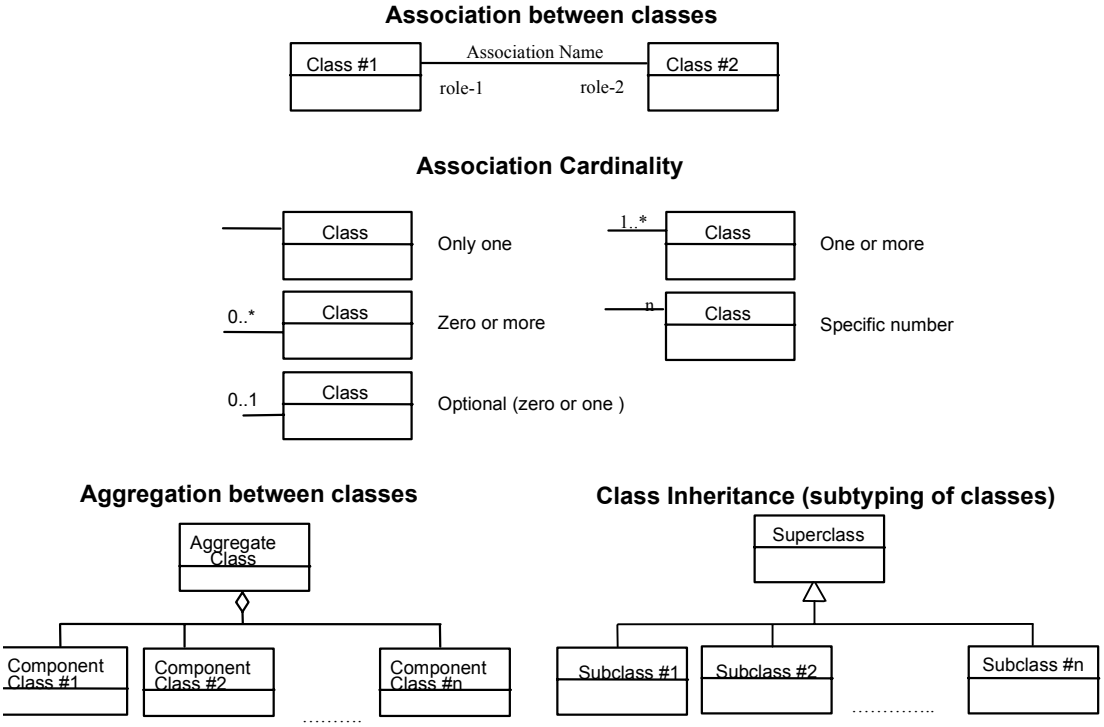

Figure 1 — UML notation

In this standard, 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.

5 Overview

The core purpose of an integrated client is to provide a unified environment that allows a user to visualize, analyze, and/or edit data from feature, imagery, and sensor web data sources simultaneously. Within the context of the OGC, this means that the integrated client allows a user to publish, discover, access, integrate and apply all types of spatial data (e.g., raster, vector, coverages, and sensor observations) from a wide range of vendor "web services" through OGC standard interfaces.

The functionality of an integrated client can be divided into the following five categories:

A) Service Discovery & Binding

B) Feature Production

C) Imagery Production/Exploitation

D) Sensor Web Planning/Exploitation

E) Project Persistence and Sharing

Each of these functional categories is described in additional detail in the following sub-sections.

This section also presents use cases for the integrated client. These use cases are not necessarily limited the functionality in which they have been placed. In some circumstances they include functionality from more than one category.

For each integrated client, the implementation must harness specific technologies and adopt particular architectural approaches. Each technology/architecture pairing presents different reliability, availability, serviceability, usability, security, and performance characteristics. And as such, different technology/architecture pairings may be more or less suitable for various purposes across an enterprise. This issue will be discussed in detail in section 7, Architectural Design Considerations.

5.1 Service Discovery & Binding

A service registry is a software component that supports the run-time discovery and evaluation of available service offerings. The Service Discovery & Binding functionality of the integrated client provides, as a minimum, a tool for finding data and services by querying service registries.

There are a number of existing service registries in use, and as the number of available registries grows it will become increasingly difficult for users to find all the possible data of interest and choose the best data for the task at hand. The functionality provided by the integrated client is intended to assist the user in maintaining persistent knowledge of a set of service registries, executing queries against these registries, and creating service chains to provide discovered data to the client in the desired form.

The Service Discovery & Binding functionality can be divided into the following 5 functions:

A) Registering a service to a Service Registry (WRS)

Once GIS data is published in an OGC web service instance (W*S), its presence must be announced so that GIS data analysts can find it. A GIS data provider can do this by using an integrated client to register the service with a WRS (see Figure 2). The client generates an XML document containing the URL and other information pertaining to the service and sends this document to the WRS. When the WRS receives the request, it then queries the W*S service for its capabilities.

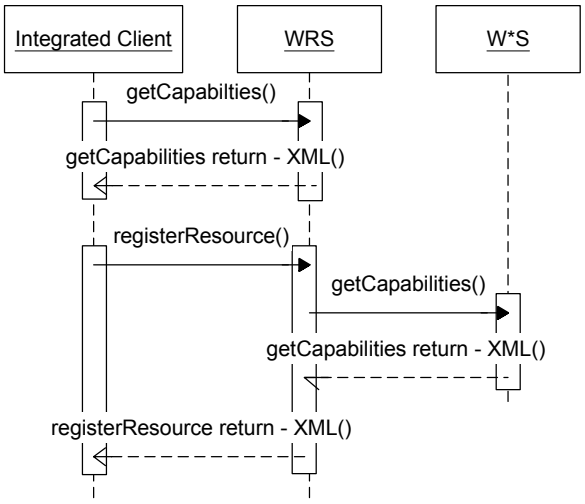

Figure 2: Service Registration and Harvesting

B) Querying a Service Registry for OGC Web Services.

A GIS analyst must be able to locate OGC Web Services. The analyst can use an integrated client to query a WRS for available services (see Figure 3) based on location and other parameters. The WRS returns an XML document containing capability metadata for the available services. The client should present these results in such a way that the analyst could select a specific service and view its capabilities.

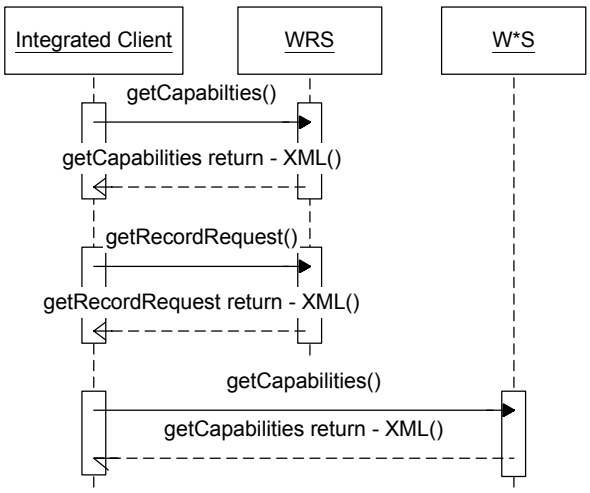

Figure 3: Service Discovery

C) Querying a Service Registry for data layers.

Service registries not only contain information on the services registered, they also contain metadata on the data layers contained by each service. A GIS analyst can query an integrated client to discover not only services but also data layers. The client retrieves metadata for these layers from the service registry so that the analyst can filter the results in order to find available data which meets time-of-collection and data quality requirements.

D) Assembling Service Chains to provide data layers for the client.

Integrated clients, no matter how complex, will never be able to render every possible data source. Therefore, additional services may be required to generate an appropriate data layer. The client can be used to discovery additional data transformation and portrayal services that can be chained together to produce a data layer that can be supported.

In the example in figure 4, a GIS analyst uses an integrated client to query a WRS and discover a Web Feature Server (WFS). In this scenario, the data in the WFS is rendered by a WMS that supports external feature rendering. The resulting service chain can be used by the integrated client to retrieve a data layer. The client then queries the WMS, passing all of the data required to retrieve feature information from the WFS. The WMS retrieves the feature data from the WFS and returns a raster image to the client.

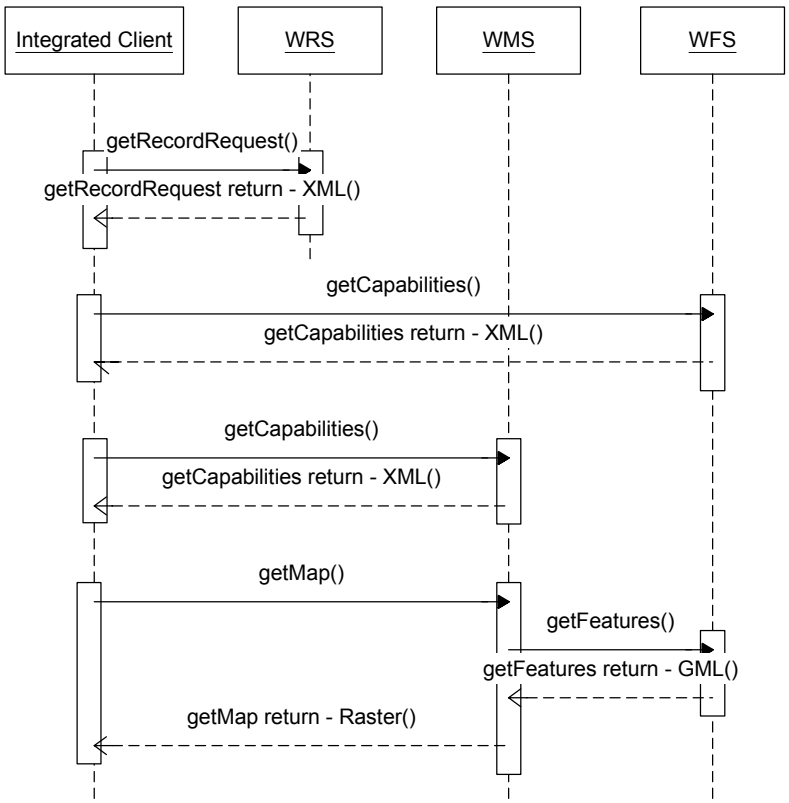

Figure 4: Service Chaining Example

In another example, a GIS analyst invokes a tool that allows the user to specify a filter for how they want to view the imagery. First, the client accesses a Service Registry to find a Web Coverage Service that will operate on one of the output formats produced by the Imagery Archive Service. The analyst identifies the appropriate service. Next the client searches the Service Registry to find a matching Coverage Portrayal Service. It finds a match, but it can not support the results because a coordinate transformation is required. The client then proceeds to search the Service Registry to find a matching Coordinate Transformation Service. Once this is completed, the client has the full sequence of necessary operations and constructs and invokes the following service chain: Web Coverage Service→Coordinate Transformation Service→Coverage Portrayal Service. The client then queries to the Web Coverage Service, requesting multiple bands of the imagery, as a reduced-resolution "browse thumbnail" for the overall area. The coverage data is chained through the chosen Coordinate Transformation Service and Coverage Portrayal Service, which renders the imagery as a simple JPEG image. The analyst then browses the resulting image using the client imagery viewer. Having found a desirable image, the user fetches the full-resolution imagery using the same service chain: Web Coverage Service→Coordinate Transformation Service→Coverage Portrayal Service. The analyst can now browse and combine the different image bands as needed.

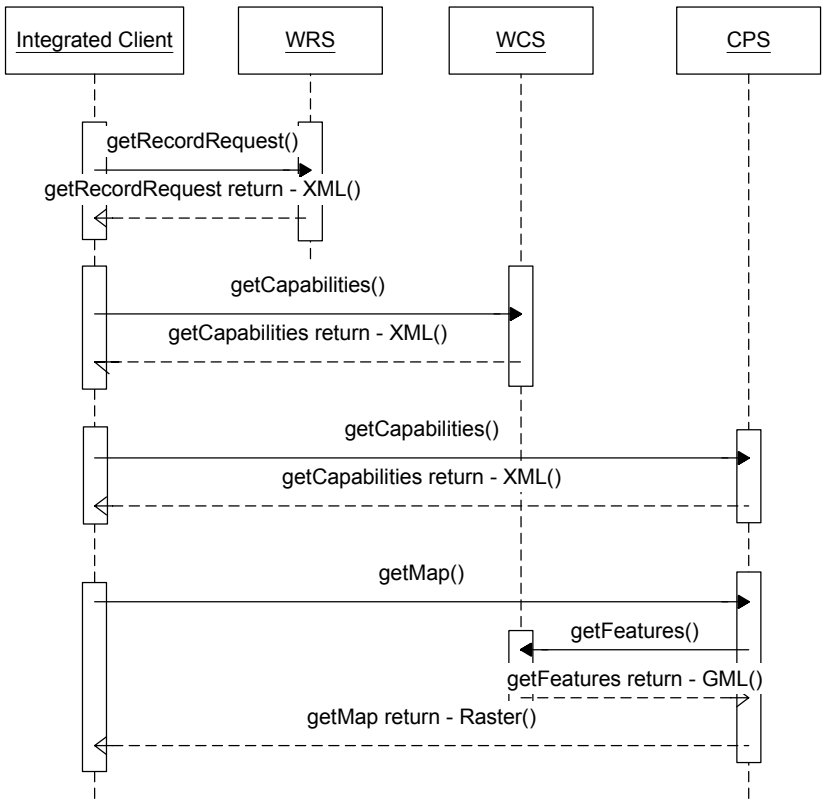

Figure 5: Additional Service Chaining Example

E) Managing a collection of Service Registries.

Instead of querying a specific WRS, an integrated client could potentially query multiple WRS services simultaneously. This has the potential of increasing the breadth of the search, but there are ramifications for performing this operation. Since each query returns an XML document which may be quite large, bandwidth restrictions may make this operation impractical. There is also the potential for retrieving multiple duplicate entries and the complexity of organizing the results from multiple servers.

5.2 Imagery Production/Exploitation

The Imagery Exploitation functionality serves to provide retrieval and viewing of imagery. This includes querying for imagery based on geometry and attributes and creation of service chains to utilize additional services to render the imagery in a specific manner. The user will use this component to find and use imagery data, and then find and use imagery application services to operate on the imagery data. The Imagery Production functionality requires support of some or all of the following OGC interfaces: WMS, WCS, CPS, ICS, and IAS. It can be divided into the following 4 functions:

A) Querying an Imagery Catalog.

The client must have search tools to specify, find, and retrieve data. The client must also provide the user the means to view and interact with the data. The client must have tools

to select and invoke imagery application services, and to invoke service chains (e.g. Image Catalog→Image Archive→Coordinate Transformation Service→Web Coverage Service). The client might access map data to depict their study area, view imagery footprints from an Image Catalog, select imagery coverage, etc. This also involves using Web Map Servers and Web Feature Servers.

B) Retrieving Imagery from an Imagery Archive.

The user wants a recent imagery over the disaster area. The user formulates a request based upon the well-known Imagery Metadata Model employed by the Image Catalog. The user employs the client to access an Image Catalog to find recent satellite, aerial and ground imagery of the area. (As described here, the client knows about the Image Catalog Service, but the client might also discover this service through a service registry that operates as a broker for several Image Catalogs.)

The user finds the Image Metadata they want through the catalog search and now must access the appropriate Imagery Archive Service to fetch the imagery and imagery support data. The client formulates the request to the archive, stipulating where the data are to be delivered for the client to later exploit. This process might take some time, if for example the archive has to fetch the data from tape storage. The Imagery Archive Service completes its assignment by delivering the imagery data to the appropriate Web address. Optionally, the Imagery Archive Service might employ a Notification Service to alert the User about the availability of their requested data. The data is now available for exploiting, although it is still in its tiled archive format. (The archive service likely supports mosaicing, re-tiling, and re-sampling to deliver the imagery in a form that is ready for exploitation.)

C) Assembling a Service Chain to retrieve raster data from a WCS and rendered according to client specified styles and parameters by a CPS.

D) Local manipulation of imagery (translucency, edge detection, etc.)

5.3 Feature Production

The Feature Production functionality serves to provide retrieval and viewing of feature geometry and attributes, supporting complex querying for features based on geometry and attributes, cartographic portrayal of feature data, feature analysis, and feature editing capabilities. The Feature Production functionality requires support of one or more of the following OGC interfaces: WMS, WFS, FPS, SMS, SLD. It can be divided into the following 2 functions:

A) Managing/editing features contained in a WFS-T.

A Transactional Web Feature Server (WFS-T) allows users to retrieve and modify feature data. In the example in figure 6, a GIS data producer employs recent imagery as a source for feature analysis and update. The integrated client employs an Image Catalog Service and Image Archive Service to access the imagery. Next, the user browses and queries Web Service Registries for feature metadata. The user employs this metadata to select the appropriate feature data for use in disaster response. Having discovered the appropriate feature data, the client then employs a Transactional Web Feature Service

(WFS-T) to access the feature data. The client then uses feature extraction tools to update the data.

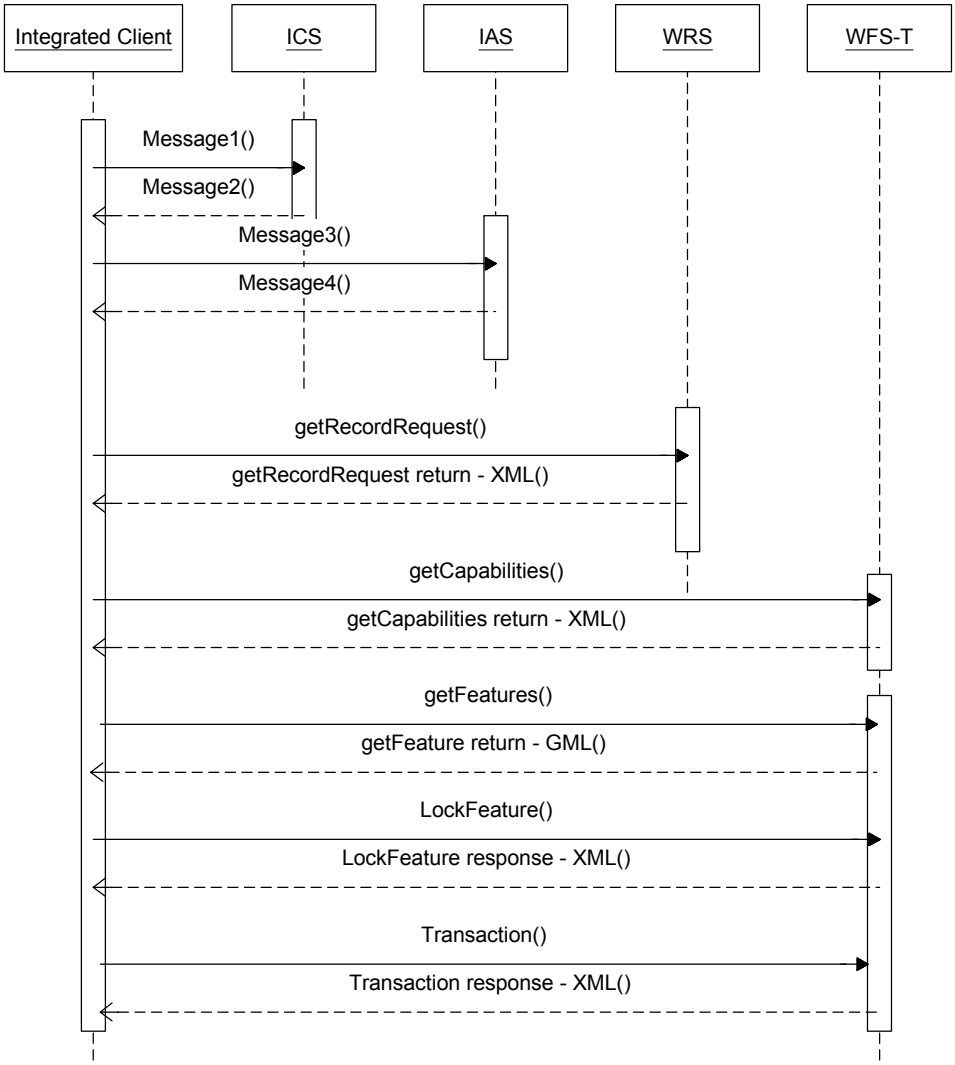

Figure 6: WFS Transaction with Imagery Server Support

B) Assembling a Service Chain.

The client provides the means to view, filter, and interact with feature data rendered according to client defined styles and client specified parameters.

In the example in Figure 7, an SLD-enabled Web Map Service is used to portray cartographic layers that are styled according to user-selected symbolization preferences. The user selects the desired SLD from an SMS; this SLD was previously constructed using a Style Editor component.

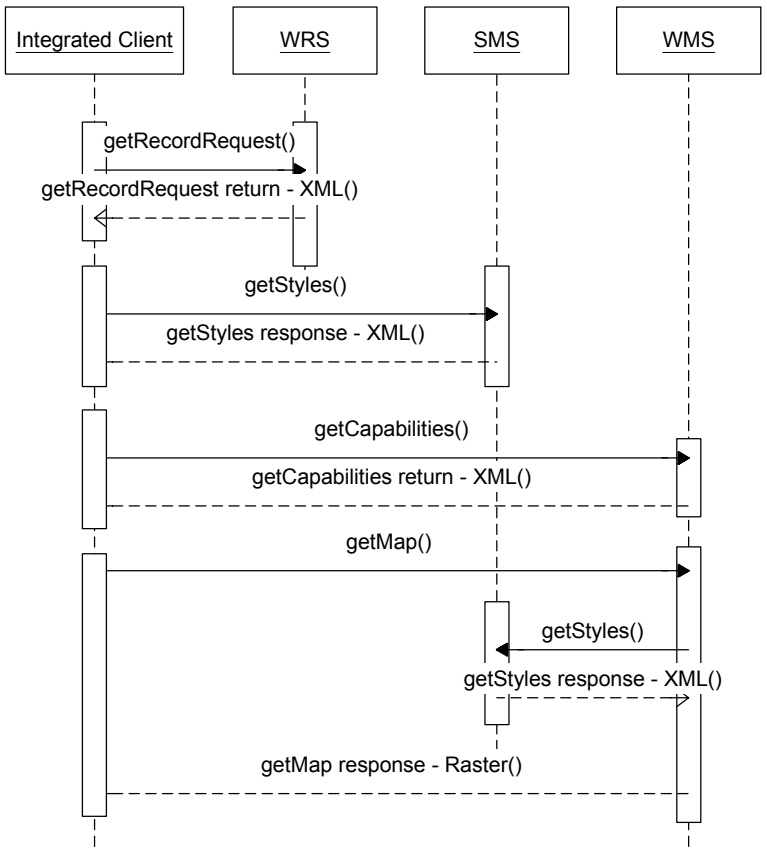

Figure 7: Feature Portrayal using SMS

5.4 Sensor Web Production

A number of remote sensors, but in-situ and mobile, are in use today. The data from these sensors can be analyzed for their spatial and temporal patterns and visualized through maps either statically or via animation. A number of OGC services were created to provide a common framework for working with sensors which are connected to the Internet. The Sensor Web Exploitation functionality requires support of some or all of the following service types: SPS, SCS, WNS. It can be divided into the following 3 functions:

A) Retrieving sensor data from a SCS.

Support for a Sensor Collection Service (SCS) allows users to retrieve data from a remote sensor. Sensors may be queried by location, time, and coordinate system. The SCS responds to a query with an XML document containing the sensor observation data.

In the example in Figure 8, an Emergency Management analyst wants to find the water level *l* at any location *p* within the river system at time *t*, in response to rising water levels during a severe thunderstorm. The analyst builds a query for a OGC Registry Service asking "Is there a service that is able to give me the data I need synchronously?" and receives the URL of an appropriate SCS. The analyst retrieves the capabilities of the

SCS. The capabilities indicate that the service provides data for the entire river system, and the observations can be retrieved by providing the location and time. The analyst provides these parameters and sends them to the SCS using the getObservation() request; the SCS responds with the appropriate observation data.

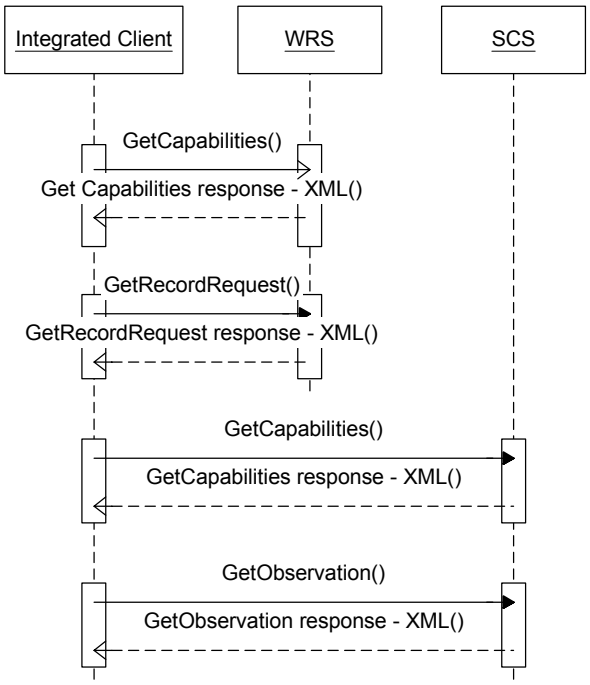

Figure 8: In-Situ Synchronous Water Level Sensor

B) Managing a sensor plan through a Sensor Planning Service.

The Sensor Planning Service (SPS) is used to generate and edit collection plans. Pre-collection prediction capability is used to help develop the plans required for mobile sensors to provide the needed sensor coverage. This service accepts location information identifying the region/target of sensor coverage. The prediction capability considers the physical environment, communications environment, sensor, and platform to determine the relevant area, path, time, duration, and/or similar parameters, and acceptable deviations that the platform must take into account to correctly position the sensor. In a UAV scenario, the pre-collection prediction capability may determine the collection geometry, which may be represented in 2D or 3D to help identify possible flight area/path, speed, and elevation in a way users can insure that the planned sensor flight provides the needed sensor coverage. Displaying similar information for a series of regions/targets can help the user identify a complete flight circuit appropriate for single sensor collection against multiple targets. This service allows a UAV collection plan to be generated and then saved. In addition to the flight plan details, corresponding sensor Collection Requests are also specified. This information is used to fly the UAV and task the air quality sensor to perform collections.

C) Handling sensor plan notifications from a Web Notification Service.

When a request is made through an SPS and it is not immediately known whether the requested action can be performed, a WNS is used to notify the user that the collection has been successful. The user is then free to use utilize the SCS functionality to retrieve the data.

In the example in Figure 9, an Emergency Management analyst wants to know the concentration of a hazardous airborne pollutant at any location within a city and surroundings at a given time *t*, in response to a disaster at a chemical plant. The analyst builds a query for a OGC Registry Service asking: "Is there a service that is able to give me the data I need?" and receives the URL of an SPS. The analyst retrieves the capabilities of the SPS. The analyst sends a collection feasibility request to the SPS for their desired collection; the SPS responds with an affirmative. The analyst submits the collection request to the SPS; the SPS registers the request with a WNS. When the collection has been completed the SPS informs the WNS, and the WNS notifies the client that the collection has been stored in a given SCS. The client retrieves the observations from the SCS.

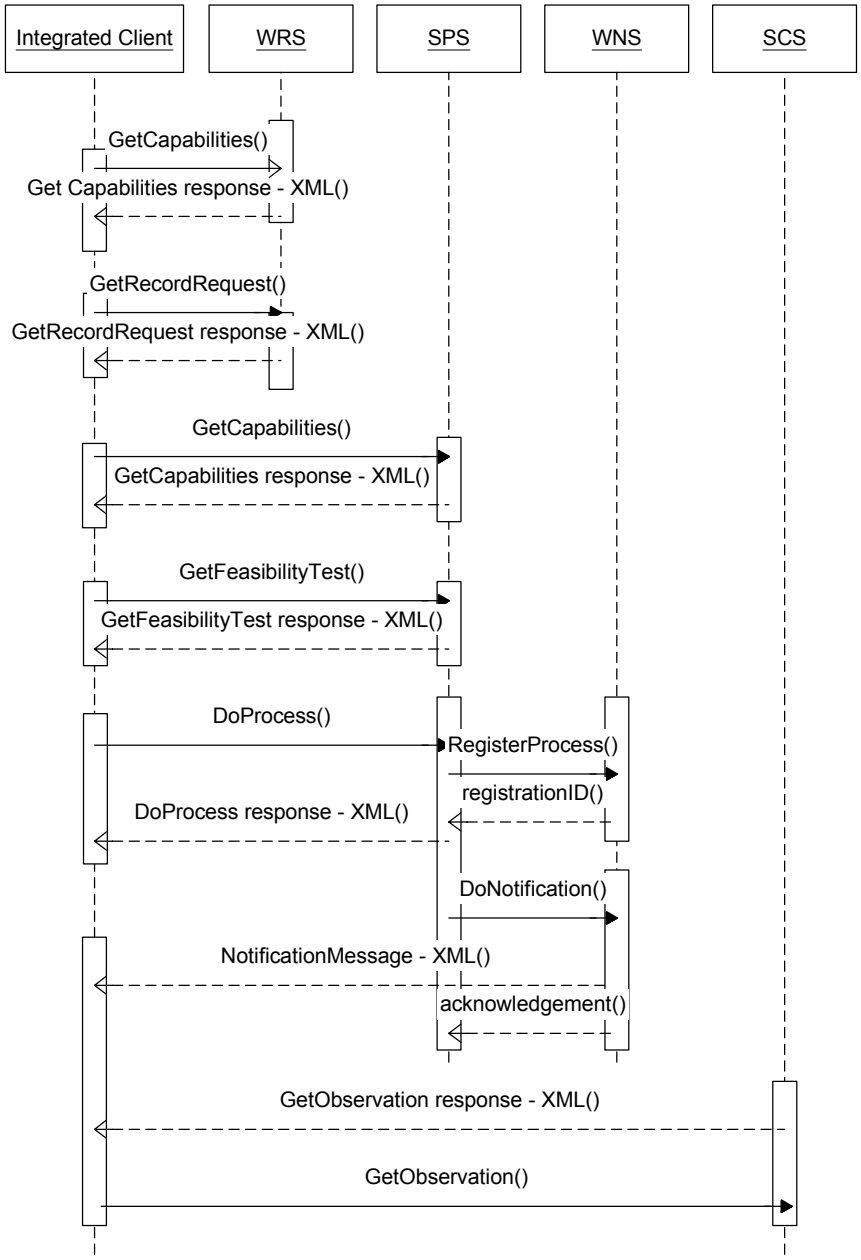

Figure 9: In-Situ Asynchronous Air Quality Sensor

5.5 Project Persistence & Sharing

The Project Persistence & Sharing functionality serves the need for users to maintain a flat file representation of the knowledge aggregated in a client project and enables sharing of this knowledge between different clients. The current most robust OGC specification for supporting this functionality is known as the WMS Context Specification. There are however, other draft OGC specifications, as well as higher aspirations.

5.5.1 WMS Context

The present Context specification is known as a "Web Map Context Document," or simply a "Context." It states how a specific grouping of one or more maps from one or more map servers can be described in a portable, platform-independent format for storage in a repository or for transmission between clients. There are several possible uses for Context documents:

- The Context document can provide default start-up views for particular classes of user. Such a document would have a long lifetime and public accessibility.
- The Context document can save the state of a viewer client as the user navigates and modifies map layers.
- The Context document can store not only the current settings but also additional information about each layer (e.g., available styles, formats, SRS, etc.) to avoid having to query the map server again once the user has selected a layer.
- The Context document could be saved from one client session and transferred to a different client application to start up with the same context. Contexts could be catalogued and discovered, thus providing a level of granularity broader than individual layers.

A Context is an XML document that includes information about the server(s) providing layer(s) in the overall map, the bounding box and map projection shared by all the maps, sufficient operational metadata for Client software to reproduce the map, and ancillary metadata used to annotate or describe the maps and their provenance for the benefit of human viewers.

5.5.2 Location Organizer Folder (LOF)

The LOF specification is a draft specification which has not seen much movement since the Geospatial Fusion Services (GFS) test bed. In concept, it is a “grab bag” that enables users to create a collection of “geo-links” as features, and link in a wide range of http: linkable resources (e.g., video, web pages, documents, graphics, etc.) into a personalized collection. This XML document can be shared with colleagues in either “thick” or “thin” versions. A thin LOF is just an XML document with links to remote resources. A thick LOF actually harvests the remote resources and stores them locally in the document.

The LOF is hindered by the need for a specific application schema, which has not been well specified, and is not in synch with several evolving OGC specifications. It is likely that the LOF specification will ultimately we overcome by the next generation of Context specifications.

5.5.3 Capturing Additional Project Information

In the future, it would be desirable to have project persistence encodings that support annotations (e.g. XIMA or some other form), WFS queries/filter expressions, and service chaining (for instance, WCS to CPS). It is probable that a WMS/WFS Context will be the next natural step toward more comprehensive project sharing.

6 Requirements

Requirements for an integrated client come from a variety of sources.

6.1 Sponsor Requirements

Sponsors of the development initiative for an integrated client that interacts with a variety of OGC services have a variety of goals in funding this initiative. Included among these are:

- Simplicity of Environment: one consistent user environment in which an end user can interact with and use all OGC services.
- Applicability to their respective domains of interest.
- Demonstration in the context of an easily followed narrative that depicts a realistic scenario involving geographic analysis.

6.2 End User Requirements

End user requirements for geospatial clients are as varied as the users themselves, and we cannot detail them here. However, the following are of particular importance and generality in the OWS1.2 initiative and in the context of bringing new, easy-to-use technology to a community that is already sophisticated in the use of tools for geospatial analysis:

- Simplicity and familiarity of use. Standard paradigms for data discovery, import, collation.
- Familiar features and approaches to specifying map parameters such as bounding region, spatial reference system (SRS), scale, and styling.
- Seamless integration of data processing and data rendering capabilities.

6.3 Project creation, storage, loading, and sharing.

While project persistence is not an explicit goal for this initiative, several of the OWS1.2 integrated clients do or will support it in some capacity. Among the considerations for project persistence in this initiative are:

- Novel opportunities for technical innovation.
- Marketability: satisfaction of end user requirements for functionality and workflow management.
- Interoperability Requirements

6.4 OGC specifications

- Specifications relevant to externally accessed services, or consistent with versions under development during the OWS 1.2 test bed.

7 Architectural and Design Considerations

In practice, an integrated client implementation must harness specific technologies and adopt particular architectural approaches. Each technology/architecture pairing presents different reliability, availability, serviceability, usability, security, and performance characteristics. And as such, different technology/architecture pairings may be more or less suitable for various purposes across an enterprise.

As mentioned previously, there are five categories of functionality (or supported use cases) are called for within a fully integrated multi-service client. These are:

A) Service Discovery & Binding

B) Feature Production

C) Imagery Production/Exploitation

D) Sensor Web Planning/Exploitation

E) Project Persistence and Sharing

Each use case strains different architectures in different ways. And, different kinds of clients (e.g., thick clients, JAVA clients, Browser + Plug-ins, Browser + JAVA Applet, Browser + Active X Controls or JavaScript, and Browser + HTML), due to the capabilities of the implementation technology, are differentially capable of supporting these use cases.

A separate, but related issue is the server-side client generators that might enable thin(ner) clients to cascade various web mapping calls to other servers, or even design and invoke complex service chains. While thick client products can sometimes draw upon OGC conformant server-side components to provide such functionality, these thick clients can also enable value-adding and data manipulation functions that are impossible to replicate in a browser, and difficult to replicate in a browser when augmented by various plugins, applets, or scripting languages.

7.1 Characteristics of Client Technology and Architecture Choices

As mentioned above, each technology/architecture pairing presents different reliability, availability, serviceability, usability, security, and performance characteristics. And as such, different technology/architecture pairings may be more or less suitable for various purposes across an enterprise.

7.1.1 Reliability

The reliability of any network-based architecture can be affected by a number of factors. And, the wider the network and more distributed the server infrastructure, the more these factors are out of your direct control.

- Network bottlenecks and networks under peak usage can lead to poor quality of service for distributed geo-processing, regardless of whether the client is thick, thin, or anything in between.

- Depending on the information security measures taken at various points along the network topography, various forms of active code (e.g., JavaScript, VB script, ActiveX, etc.) will often be disallowed. Therefore, the reliability of clients utilizing active code will not be automatic across any network.

- The server infrastructure for a distributed geoprocessing solution can either be deployed and maintained for reliability, or not. If inadequate attention is paid to the server infrastructure (and when clustered, to relevant clustering technology), then reliability will be compromised.

7.1.2 Availability

Users of thick client devices enjoy a certain level of data persistence that they do not get with a browser based WMS view. When network availability fails, so does access to data. As such, client-side caching can enable the availability of whatever data is locally available at the time of network failure. However, thick client implementations still must operate within an OGC conformant distributed geo-processing infrastructure if others are to have access to value added data (e.g., annotations, attribute updates, or geometry/topology updates).

7.1.3 Serviceability

The servicing of various client technology/architecture configurations ranges widely. Thick client deployments entail configuration management which, depending on the technology, can be centrally managed. Browser deployments also require configuration management, to ensure the proper level of browser feature support. Yet, providing intermittent servicing and updating of client/application functionality can be much more frequent and simple for web-based client generators.

7.1.4 Usability

The usability of a client is primarily based on the quality of interface design. Generally it is irrelevant whether a capability is held locally in a client, or across a network. However, a browser client based upon a WMS "views" concept will pose user latency between views. This is not inherent to http://, but rather to un-augmented browsers engaged in web mapping.

7.1.5 Security

Both thick client applications and browser-based applications have the potential for offering single-sign-on PKI identification and authentication (I&A), which can conform with the Department of Defense's Defense Information Security Agency (DISA) PKI standards. Browser selection, however, may be influenced by the native vulnerabilities in the browser for managing PKI certificates. Also, allowable scripting and plug-ins may be limited in order to achieve a high level of information assurance.

The other side of information security, audit and profiling, can be implemented on each server, including client generators. Thick client applications making calls to a distributed geo-processing infrastructure would face similar logging. However, insofar as thick clients cache larger volumes of data on the client side, the click-stream analysis would be less fine-grained.

7.1.6 Performance

The same network issues introduced and discussed in the 'reliability' section above are just as important to the performance of a distributed geo-processing solution.

Thick clients have the potential to provide certain performance gains, once data is brought across the network. WMS calls bring map layers to a thick client. WFS calls bring feature data to the thick client. And, WCS calls bring subsets of coverages (e.g., imagery, etc.) to the thick client. Once in the thick client, the navigation of the data does not face any network latencies. Navigating through the data (panning, zooming, etc.) is one function that can be very susceptible to performance problems. However, to take advantage of the potential for caching data on the client side, clients must retrieve more data than just the current view. For example, allowing for client side panning would require retrieval of data well outside the extents of the current view. Unless this sort of mechanism is implemented, there is no performance advantage to the thick client for this scenario. This is a trade-off that must be considered by the implementers.

Browser based clients most often adopt the strategy of drawing upon a range of portrayal services to provide data sources through a WMS interface. This strategy provides a view into the distributed geospatial resources, with limited functionality for editing and manipulating this data at the client level. And, every new view requires another call across the network. However, the size of a WMS request and response is very minimal and poses limited impact on a network.

Particular scripting and plug-in augmentations bring the more bandwidth intensive WFS and WCS outputs directly to the browser client. Such clients can offer client-side caching of data that can reduce the number of network calls, but the volume of data originally pushed to the client far exceeds a WMS call. Also, such clients can enable actually editing and manipulation of data at the client level, which might be transactionally inserted back into the originating service.

Clustering allows greater performance than more distributed network/server topographies. While clustering over http:// does enable you to avoid network latencies, the current OGC servlet (http://) interfaces do not maximally enable clustering the way CORBA or JAVA interfaces do.

7.2 Other Architectural and Design Choices

In addition to the question of whether a client is clustered or distributed, an implementer must decide how the various components of the software connect to one another, how control is imposed on the flow of logic through any given process, and how and whether extensions are accommodated. Extension mechanisms are often important in the case of OGC interfaces, which are made available only as plug-ins or software development kits in a number of commercially important cases.

For many GIS systems that predate the growing acceptance of interoperability standards, the core system may be considered a black box that provides hooks for adaptors or translators of file formats not supported in the original design. Call-backs and callouts to extensions that support OGC interfaces are essentially functional interfaces. They may be mediated through jump tables that are populated from configuration files when the application is started. The most monolithic systems can be extended only by their developers, by compiling new interfaces directly into their code.

Some more recently developed systems are be built upon a concept of messaging between their various internal components. A system of this sort resembles a clustered architecture as described above, except that its various components all exist and communicate within the same address space, as a single application. Information passed across the internal interfaces may be documents similar or identical to those mandated by OGC standards. This approach minimizes the cost of converting external documents to internal object format.

8 Client Components and Modules

Like most complex software applications, the extensive functionality of an Integrated Client of OGC Web Services is typically implemented as a suite of software modules that have more or less independent although related functions. Although there is a relationship between functional categories and the software modules that implement them, the relationship is not necessarily one to one.

By the same token, neither functionality nor the underlying software always bear a direct relationship to user interface. In this section, we describe some of the software components that support the fundamental operations of an integrated client. In the next, we discuss user interface.

8.1 Factoring

There are many ways that code can be factored, and it will be impossible to catalog and evaluate them all in this short treatise. Consider, for example, how rendered data are compiled in a map display window. The window may be nothing more than a target region of application memory, into which each active rendering service sums its data in turn when an update event is delivered from system control logic. Alternatively, the display could be an active and self-managing buffering system that polls constituent services. We will attempt to keep our discussion above the level where code factoring makes a difference in the way a functional block is conceived.

8.2 Generic Descriptions of Client Components

The following paragraphs provide a description of modules that may be found in a client that supports a variety of OGC specifications.

8.2.1 Search and Discovery System

The ability to search for data is fundamental to Service Discovery and Binding functionality. User interface of some sort (see Section 9) is essential to this system component; it has no use as a hidden internal engine. Speaking schematically, the search subsystem connects at one end to an OGC search protocol client interface, such as a WRS, Stateless Catalog, or UDDI client. The discovery subsystem includes code capable of parsing and organizing the Capabilities Documents that come through the protocol interface, as well as analyzing and presenting the material from the parsed data.

8.2.2 Data Selection Component

Once data sources have been presented by the Search and Discovery System, they may be selected by the user for retrieval, and subsequent display or further processing. Details of the logical organization of selection software may be varied and are not discussed here. The selection event itself, resulting from user interaction with a GUI widget such as a pull-down menu or selectable list, is typically mediated by the user interface library that comes with the operating system or windows support subsystem. Ultimately, control passes to code that actually binds to a data service, retrieves the selected layers, and

caches them or passes them to display or processing components within the client application.

There are some variations on this theme. The selection software may be used to choose local as well as remote data sets. Data selection software might also connect to services that appear to be OGC data services, but in fact are opaque interfaces to rendering or other processing services that ultimately connect to data services. The only such service at this time is the Coverage Portrayal Service, which presents a WMS interface to the web.

8.2.3 Display and Navigation System

Clients that provide for visualization at all include software to control the map display, zoom controls, layer ordering and visibility control, legend, etc. This software is inherently dependent on user interface, both for visualization on output, and user control on input. There are actually several components of this. The map display itself may include a secondary buffering system in addition to the primary display buffer, to assist in the support of pan, zoom and other navigation actions.

8.2.4 Coordinate Transformation Engine

Many clients have the ability to perform coordinate transformation. This service is relevant to all three map-creation functionalities: Feature Production, Imagery Production and Exploitation, and Sensor Web Planning and Exploitation. The coordinate transformation engine is typically an essentially isolated entity within the application or an external, dynamically bound library. It does not inherently support or require user interface elements, but it may receive parameters from user interface components used for selecting transformations or target spatial reference systems. It may also report source and destination reference systems to a status display.

8.2.5 Rendering Engine

The rendering engine is the component of the client that converts data to a form in which it can be displayed graphically. The data come from remote services, including OGC data services, and possibly local data stores. Styling information may be created or manipulated locally (see below), or may be received from another source, possibly as a standard styling document such as a Style Layer Descriptor.

Note that images retrieved from WMS servers are already styled, although it would be possible for a client application to restyle them via one-to-one color remapping or some more sophisticated algorithm. However, raw data retrieved from WFS cannot be displayed unless they are rendered, with some sort of styling. Data from WCS might or might not be displayable upon receipt, depending upon format and the client's capabilities.

8.2.6 Style Editing and Management Interface

A client may include a number of tools for determining the styling of data it receives. As indicated above, this is essential for WFS. It is also highly desirable for WCS coverage data, and useful as well for WMS information, especially if it is received from an SLD-enabled WMS server.

Clients that interact with SLD-enabled WMS may contain entire subsystems for editing SLD documents. These may resemble or even be fully integrated with the style editing features that control the appearance of local data.

8.2.7 Geospatial Analysis Logic

A client that supports several OGC interfaces may be much more than a simple mapping tool. Several included in this initiative are full-fledged GIS systems. As such, they support a great variety of analytic capabilities, including unions and intersections of areas, distance and containment relationships, statistical and report generation facilities, means to combine two or more layers for analysis or enhanced display, etc.

8.2.8 Modelling Tools or Interfaces

GIS systems and simpler, more specialized clients alike may include tools for predictive or analytical modelling. They may also support interfaces to external tools that perform modelling or analysis.

8.2.9 Image Processing Engine

Graphical displays inherently require image processing, which is often provided by the operating system's user interface libraries. Tasks required of the system or application image processing code include: summing information from multiple layers into a map display buffer. and resampling and subsetting, required by map display navigation; more sophisticated resampling features for clients that support coordinate transformation for imagery.

Clients may also provide an entirely different class of image processing functionality whose purpose is to enhance the displayed imagery or perform some sort of analysis. Enhancement tools provided by some of the OWS1.2 include histogram equalization, image differentiation or integration, brightness and contrast management, and thresholding. Analytical tools are highly varied, and may be quite specialized. They include software that implements clustering and classification algorithms, noise reduction, guided or fully automated feature extraction, image stitching, orthorectification, and any number of other tasks. Such tools are typically controlled by a set of user interface elements.

8.3 Implementations Developed or Presented in OWS 1.2

The following table shows how many clients out of 8 in the project have planned support for each of several existing or emerging OGC standards:

Current or Experimental OGC Standard	Number of Integrated Clients Supporting (out of 8)
WMS	8
WFS	7

WCS	4 (+2 via CPS)
SLD	2
CPS (in addition to independent WCS support)	1
SMS	1
SCS	3
SPS	3
WRS discover	6
WRS register	1
UDDI discover	2
Sharable project persistence (Context)	1

9 User Interface

9.1 Introduction

User Interface and human factors engineering constitute a huge field in the domain of software as well as hardware development. It involves extensive test, experimentation and domain expertise. Successful human-computer interfaces seldom result from haphazard design and hasty implementation. However, they are often significantly aided by an appropriate dose of inspiration, especially in these early days of the art. Human interface is still a rapidly evolving field, and still ripe for innovation.

UI is also one very important way that vendors and other developers distinguish and brand their products. In some measure, it is dictated by the operating system: different platforms offer different means of interacting with the computer. Often these are merely stylistic variations (bevelled "3D" buttons versus rounded-rectangle areas versus "jewel-like" buttons), but in other cases represent distinctly different levels of support (Windows' three styles of combo-box vs. HTML's <select>).

For these and other reasons, this document does not seek to mandate a standard for Integrated Client UI. However, it does make observations, and records cases where certain UI elements do seem in some sense to be standard in existing implementations. It also discusses the benefits and issues of standardization in OGC Integrated Clients.

These suggestions are platform-neutral, and are intended to identify major components without codifying them in detail.

9.2 Considerations Regarding a Standard User Interface

In this section we examine the practicality of standardizing UI elements for applications that use OGC standards.

9.2.1 Benefits

The primary reward to users of interface standardization would be the consistency of experience among applications from different vendors. It allows users to understand immediately how to operate a standard component, even in unfamiliar environments, and to transition smoothly between products while performing daily tasks. It reduces the need for specialized training.

Standardization can benefit developers as well. By embodying best practices known to the industry representatives that form the standards body, it eliminates the need for redesign and shortens time to market. It also provides a market ready-trained in the use of the standard components.

9.2.2 Obstacles

Despite the benefits, and even assuming that some standard components would be valuable despite concerns about maturity of the art and vendor acceptance, there are obstacles to the adoption of a prescribed User Interface standard. Among these is the great diversity of environments currently in use, with their attendant capabilities.

At one end of the spectrum are the ultra-light clients, hosted exclusively within HTML (and perhaps DHTML) browsers. While visually rich, these tools fall short in interaction; solutions that must support a wide cross-section of available browsers are often reduced to marginal lowest-common-denominator capabilities. Even browser-specific solutions find the present state of the art very restrictive and require means of augmenting the User Interface, either by extensive scripting, or the use of embedded technologies like Java. Enhanced interaction description languages, such as XUL, are too immature at present to offer any near-term improvements to these problems. At the other end of the spectrum are heavy-weight clients, often very specific to an operating system, and usually supported by proprietary subsystems that provide very powerful capabilities, but typically only to the specific vendor's suite of products.

The diversity of operating environments (specific operating systems, such as Windows, Macintosh, etc., and platforms such as desktop, notebook, palm-top, etc.) each have nuances that distinguish them from the rest. Producing a standard that spans all these environments is not a trivial undertaking, as XVT and Java developers can attest. The novice user sees the "foreign" look and feel, and is often uncomfortable with the interaction. (Novice users on the Macintosh, for example, used to ask why such applications aren't like "real" Mac applications...)

Applications themselves often introduce complexities that make standardization more difficult. The OGC provides a set of technologies that permit multiple vendors to interact in a standard fashion. Yet, each vendor brings with it customers that are already indoctrinated with terminology and interaction metaphors specific to that vendor, or to the industry or vertical market served by that vendor.

Finally, the level of user sophistication may necessitate simplifying or entirely removing certain aspects of User Interface in some cases. Even a geographically sophisticated layperson is unlikely to know more than standard Latitude/Longitude and may care little for various means of projection. The less sophisticated layperson may not even want to know about that, and alternate means of specifying bounds may be called for. (A case in point: how often has the typical MapQuest® user been presented with a bounding box, let alone a means of specifying a Spatial Reference System?)

None of these barriers is definitive, and none precludes specifying a standard for one or more environments, or possibly a single standard that spans many environments. Some market segments might have goals of their own with respect to UI, and might welcome an interim standard based on the benefits discussed in the previous section.

9.3 UI Description Languages

While somewhat tangential to the matter of OpenGIS User Interfaces, it becomes apparent that there is a need to uniformly describe the interface, both in the design sense, and in a machine-actionable way. In the course of this initiative, we reviewed a few new languages that are intended to fill this need. The following summary identifies the most salient points of our exploration of these technologies:

- Benefits
    - Platform-independent, automatically interpreted GUI object rendering

    - Different platform-appropriate interpretations on different OS, different hardware (e.g., desktop vs. hand-held).
    - Possible emerging standards? (No clear single leading candidate at this point.)
- Example languages, limitations, and liabilities
    - Current state of UI description languages may be insufficient to support any but the simplest applications; too coarse to capture subtleties of layout.
    - XUL: Supports HTML only, and in practice often requires platform-specific code embedded in the document.
    - UIML: More general, and getting stronger, but still supports no encoding of functionality, in particular of data validation or behavior.
    - XIML: Includes behavior. No assessment is available at this time.
- UML itself may be adequate to address most issues, especially regarding behavior of these objects with respect to the rest of the application.
- The topic needs more exploration.
    - The notion of using UIML was presented and discussed at the September, 2002 TC Meeting (by Ron Lake, Galdos, Inc.). We can anticipate further research and discussion.
    - UI Description languages (UIDL's) have gained momentum recently, but are still immature. Premature adoption of one UIDL over another may prove counterproductive down the road, possibly requiring some early-adoptees to abandon thwarted standards. Nevertheless, we do believe it would be appropriate to establish a set of evaluation criteria (platform-neutrality, ubiquity, simplicity, richness, etc.) to apply to candidate languages, in order to make an informed recommendation as soon as it is practical to do so.

9.4 Additional UI Topics

There are a number of topics relevant to user interface that we explored, but were not able to fully elucidate in this initiative. We note them here to identify them as points of interest to readers of this document, and to serve as a possible guide to future work in this arena.

- Generic use cases that require some involvement by the user, and thus some form of UI:
    - Search
    - Layer selection and ordering
    - Zooming and other navigation

    - Specifying style and other presentation
    - Tasking active sensors
    - Specifying transformations, or transformation pipelines
    - Bounding box definition: both numerical and graphical
- Opportunities arising from other applications use cases:
    - Catalog a list of UI functional requirements (a minimum feature-set) for each web mapping use case, as detailed in the Overview (Section 5) above, and also as identified in other specific use cases that may come to light.
    - Catalog/describe a range of UI styles at least for the most basic web mapping use-cases, as a sort of style-guide.
- Window allocation and screen real estate: how much space should be allocated to map views as opposed to controls, and under what circumstances? Large, viewable maps are often desirable, but so are accessible and complete control clusters. Is it desirable to have one window that contains display and a full suite of controls, or to support multiple windows? What are the trade-offs for different groups of geospatial systems users?
- Modal vs. modeless behaviors: Interface modes are an important consideration in all UI designs. The most familiar example with geospatial tools is probably navigation. What are the relative benefits of being in "zoom in," "zoom out," or "pan" mode as opposed to having all these operations available all the time? Specialized panels that lock out the rest of the display while the user sets preferences, styling parameters, or other features of an application are another case in point.
- Differences or similarities in UI for accessing different services (WMS, WFS, WCS, SWE data services, various styling services). The suite of clients offered for OWS1.2 provides a substantial arena for this comparison. In overview, it appears that the interfaces for the web mapping services were quite similar, with some variation to support rendering of pure-data services (WFS and WCS). Sensor interfaces were different. However, these comparisons were not quantified. A thorough study would require an evaluator to become familiar with the operation of all the clients.
- Generic user vs. expert modes
- Specific UI elements: There are many ways to do navigation, discovery and metadata presentation. Is there a common "best-practice" current approach? What new, innovative interfaces may be forthcoming in the near future? What requirements might drive developing them? One form or another of each of the following components appeared in several of the Integrated Clients developed during the OWS1.2 Initiative:
    - Map Window

- Navigation tools: zoom, pan, or jump
- Discovery tools and layer selection
- Metadata and auxiliary data display, and filter editors or controls
- Styling editors and controls, including layer ordering tools
- Sensor tasking or query tools and sensor data visualization
- Front ends for modeling or analysis engines, including image or geospatial processors. These appeared in relatively simple WCS clients as well as in general-purpose GIS applications.
- Notification panels or annunciators

9.5 Summary and recommendations

- Because the science of user interface is moving rapidly, and because vendors use user interface to provide their own added value and provide brand identification, it is not appropriate at this time to mandate a standard OGC user interface for web mapping applications.
- There may be opportunities and benefits to specifying a set of standard components, and even the arrangement of these components, for certain market segments. Benefits could include capture of best practices from a broad-based group of implementers and users, a reduced cost for user interface design, and a resultant shorter time to market.
- Any such guidelines should be
  - based upon extensive usability testing
  - extensible, to accommodate innovation and new requirements
  - platform neutral, with respect to vendors, hardware platform, and underlying transport protocol.
- Guidelines could define minimal levels of support, and offer tiers of improved behavior for more sophisticated users, or richer technology platforms.

10 Test Considerations and Results

10.1 Issues and Opportunities

10.1.1 Combinatorics of testing many services.

One requirement of developing an integrated client application that supports multiple OGC services is that the application be tested against all services, and as many instances of each one as possible. This produces a much more substantial burden than testing one interface alone. Mechanics of the test effort itself take time and resources, but the greater effort involves communication with a greater number of organizations, and collating, reporting, and especially resolving failures, partial successes, and ambiguities.

10.1.2 Staged testing; leveraging one service to test others

- Early in the OWS1.2 initiative, we had hoped to gain some leverage against the combinatoric problem by using intermediate services in a chain to test several services that lay beyond it, allowing the exercise of two or more services at one time. This opportunity did not present itself, as the only such service ultimately considered for this work was a CPS instance that was not tested by any integrated clients.

10.2 External vs. Internal Interoperability

Talking to each external service independently is not all there is to exercising an integrated client. We anticipated that some issues might arise in testing whether the internal representations and the presentation of the data are consistent among the services accessed by the clients.

In fact, no prominent issues of this sort did arise. The primary issue was one that has been encountered in many other initiatives, and by clients that access only a single service type: inconsistent spatial reference systems among the remote data sets. Integrated clients with substantial local functionality can provide a solution to this issue, however, by performing coordinate transformations on the data they retrieve.

10.3 TIEs and Results

A number of Technology Integration Experiments (TIEs) have been performed between the different integrated client implementations and the different OGC services. The results of these TIEs appear in Annex A.

These results are also summarized in the following tables. Successes and failures, with some qualifying information, are recorded in the relevant table locations. A blank entry means that no test was reported.

Please note that this summary does not capture all the nuances of the reports in Annex A. A success may mean that the client and server worked flawlessly, but only in a constrained set of experiments. TIEs noted here as failures may have been very close to successful interactions, and may have resulted from inconsistent interpretations of existing or experimental specifications rather than flaws in client or server software. It is also likely that some informal experimentation was not recorded as a TIE at all.

For a full appreciation of the Integrated Client tests and results, please refer to the Annex.

10.3.1 WRS Registry TIEs

WRS Servers:	CubeWerx	Galdos	Ionic	Polexis	Syncline
Clients:					
Laser-Scan, Inc.	Success, but with non-compliant query	No services registered	Failed	Failed: null response	Failed
Ionic SA	Partial success?	Partial success?			Partial success?
Autodesk	Success with workaround	Success	Failed: XML issue	Success, but registry empty	Failed
Intergraph					
GMU	Success				
UAH					
Polexis	Exception response	Success: requires workaround	Failed		Failed

10.3.2 UDDI Registry TIEs

UDDI Servers:	NASA	Syncline
Clients:		
Laser-Scan, Inc.		Failed
Ionic SA		Success
Autodesk		
Intergraph		
GMU		
UAH		
Polexis		

10.3.3 WMS TIEs

Servers:	CAST	CubeWerx	Intergraph	Ionic	JPL Landsat Mosaic	NASA GLOBE data
Clients:						
Laser-Scan, Inc.	Success when server is available	Success with most layers	Success	Success with some layers		
Ionic SA	Success	Success	Success			Success
Autodesk	Success	Success	Success	Success	Success	
Intergraph						
GMU	Success	Success	Success			
UAH	Success	Success	Success		Success	
Polexis		Success, including SLD				

10.3.4 WFS TIEs

Servers:	CubeWerx	Galdos	Intergraph	Ionic
Clients:				
Laser-Scan, Inc.			Success	
Ionic SA	Success, once schema issues resolved	Success, once schema issues resolved	Success, once schema issues resolved	
Autodesk	Limited Success	Success	Success	Limited Success
Intergraph	Success – some pending fixes		Success	Success in parse and retrieve; issues with WFS-T
GMU	Success		Success	Success, including WFS-T
UAH			Success	Success
Polexis				

10.3.5 WCS and IAS TIEs

Servers:	CubeWerx	Intergraph	GMU	PCI	UAH	Intergraph IAS
Clients:						
Laser-Scan, Inc.	Version mismatch	Success with v. 0.4	Version mismatch	Version mismatch		
Ionic SA						
Autodesk						
Intergraph			Success	Success	Success	Success
GMU						
UAH					Success	
Polexis						

10.3.6 SCS TIEs

SCS Server:	Polexis
Clients:	
Laser-Scan, Inc.	Success (intermittent connection issue)
Ionic SA	Success
Autodesk	
Intergraph	
GMU	
UAH	
Polexis	

11 Summary

In this document, we have addressed a variety of topics concerned with clients that integrate the ability to access several OGC-compliant services within a single application, and merge the acquired data into a single map or information display. We have touched on various aspects of client architecture, including the impacts of component distribution across a network, and choice of implementation technology. We have identified a variety of generic and specialized use cases. We have described the functional components of an integrated client, and provided an overview of some user interface features and considerations.

Perhaps most important, this project has resulted in the creation or extension of seven multi-service, integrated OGC client implementations, which have been tested as reported in this document, and deployed in an extensive live demonstration. Further exercise, testing, refinement, and extension of these and other clients is the best way to gain deeper insight into the relative merits of different approaches to client creation.

Annex A – Technology Integration Experiments

1 Laser-Scan, Inc.

1.1 Galdos WRS

1.1.1 Operations Exercised

- WRS GetCapabilities
- WRS GetRecord on WMS and WFS

1.1.2 Test Procedures

1. Open the WorldView client
2. Select the Query Manager tab
3. Select New Query
4. Select the server being queried from the Registry Server drop down (or create a new one if it does not exist)
5. Select Refresh and note the results
6. Enter a title to query against and select Refresh again. Note the results.
7. Enter a string into the abstract keyword field and select Refresh again. Note the results.

1.1.3 Test Results

There are no WMS or WFS servers in this particular WRS. We always get back an empty result set. This prohibits us from advancing past step 5 in the test.

1.1.4 Next Steps (Actions Required)

We will continue to try these tests periodically until we find some data and can continue.

1.2 Syncline WRS

1.2.1 Operations Exercised

- WRS GetCapabilities
- WRS GetRecord on WMS and WFS

1.2.2 Test Procedures

1. Open the WorldView client

2. Select the Query Manager tab

3. Select New Query

4. Select the server being queried from the Registry Server drop down (or create a new one if it does not exist)

5. Select Refresh and note the results

6. Enter a title to query against and select Refresh again. Note the results.

7. Enter a string into the abstract keyword field and select Refresh again. Note the results.

1.2.3 Test Results

Bad Registry message. The server responded with connection refused.

1.2.4 Next Steps (Actions Required)

We will continue to try these tests periodically until they work.

1.3 Ionic WRS

1.3.1 Operations Exercised

- WRS GetCapabilities
- WRS GetRecord on WMS and WFS

1.3.2 Test Procedures

1. Open the WorldView client

2. Select the Query Manager tab

3. Select New Query

4. Select the server being queried from the Registry Server drop down (or create a new one if it does not exist)

5. Select Refresh and note the results

6. Enter a title to query against and select Refresh again. Note the results.

7. Enter a string into the abstract keyword field and select Refresh again. Note the results.

1.3.3 Test Results

Bad Registry message. The server responded with connection refused.

1.3.4 Next Steps (Actions Required)

We will continue to try these tests periodically until we see better results.

1.4 Polexis WRS

1.4.1 Operations Exercised

- WRS GetCapabilities
- WRS GetRecord on WMS and WFS

1.4.2 Test Procedures

1. Open the WorldView client

2. Select the Query Manager tab

3. Select New Query

4. Select the server being queried from the Registry Server drop down (or create a new one if it does not exist)

5. Select Refresh and note the results

6. Enter a title to query against and select Refresh again. Note the results.

7. Enter a string into the abstract keyword field and select Refresh again. Note the results.

1.4.3 Test Results

Bad Registry. The server responded with null.

1.4.4 Next Steps (Actions Required)

We will continue to try these tests periodically until we find some data and can continue.

1.5 Cubewerx WRS

1.5.1 1.5.1 Operations Exercised

- WRS GetCapabilities
- WRS GetRecord on WMS and WFS

1.5.2 Test Procedures

1. Open the WorldView client

2. Select the Query Manager tab

3. Select New Query

4. Select the server being queried from the Registry Server drop down (or create a new one if it does not exist)

5. Select Refresh and note the results

6. Enter a title to query against and select Refresh again. Note the results.

7. Enter a string into the abstract keyword field and select Refresh again. Note the results.

1.5.3 Test Results

- We can successfully query for servers layers in the CubeWerx WMS

- Only Cubewerx data appears in the registry, and only the CubeWerx WMS

- We can only support a version which is not OGC compliant. We can not use the same query as Galdos or Polexis

- Searching only searches on the name of the layer inside the database. The concept of querying on the title or keywords is misleading.

1.5.4 Next Steps (Actions Required)

We are waiting for an OGC compliant WRS to test against

1.6 Syncline UDDI

1.6.1 Operations Exercised

- UDDI GetCapabilities

- UDDI GetRecord on WMS and WFS

1.6.2 Test Procedures

1. Open the WorldView client

2. Select the Query Manager tab

3. Select New Query

4. Select the server being queried from the Registry Server drop down (or create a new one if it does not exist)

5. Select Refresh and note the results

6. Enter a title to query against and select Refresh again. Note the results.

7. Enter a string into the abstract keyword field and select Refresh again. Note the results.

1.6.3 Test Results

Bad Registry message. Java.net.Connection Exception:Operation timed out.

1.6.4 Next Steps (Actions Required)

We will continue to try these tests periodically until we find some data and can continue.

1.7 CubeWerx WMS

1.7.1 Operations Exercised

- WMS getCapabilities
- WMS getMap

1.7.2 Test Procedures

1. Select the Source tab of the client and select New

2. Enter the URL of the WMS based on the TWIKI. Select Refresh. View the results in the bottom text box. Select OK.

3. Select the Layer tab of the client and select New

4. Select the appropriate source in the drop down. Select the requested layers in the right tree that you want to view and select the arrow to move them into the left list. Select OK.

1.7.3 Test Results

We can connect to the server, but we get a Cubeserv Error when trying to display the Foundation data. Other layers work properly.

1.8 Ionic WMS

1.8.1 Operations Exercised

- WMS getCapabilities
- WMS getMap

1.8.2 Test Procedures

1. Select the Source tab of the client and select New

2. Enter the URL of the WMS based on the TWIKI. Select Refresh. View the results in the bottom text box. Select OK.

3. Select the Layer tab of the client and select New

4. Select the appropriate source in the drop down. Select the requested layers in the right tree that you want to view and select the arrow to move them into the left list. Select OK.

1.8.3 Test Results

We can retrieve data from some of the available layers.

1.9 Intergraph WMS

1.9.1 Operations Exercised

- WMS getCapabilities
- WMS getMap

1.9.2 Test Procedures

1. Select the Source tab of the client and select New

2. Enter the URL of the WMS based on the TWIKI. Select Refresh. View the results in the bottom text box. Select OK.

3. Select the Layer tab of the client and select New

4. Select the appropriate source in the drop down. Select the requested layers in the right tree that you want to view and select the arrow to move them into the left list. Select OK.

1.9.3 Test Results

We can connect to the server, and the selected layers work.

1.10 Ionic WMS

1.10.1 1.10.1 Operations Exercised

- WMS getCapabilities
- WMS getMap

1.10.2 Known problems or limitations

The CAST server is unreliable. It is sometimes very slow and sometimes it does not work at all.

1.10.3 Test Procedures

1. Select the Source tab of the client and select New

2. Enter the URL of the WMS based on the TWIKI. Select Refresh. View the results in the bottom text box. Select OK.

3. Select the Layer tab of the client and select New

4. Select the appropriate source in the drop down. Select the requested layers in the right tree that you want to view and select the arrow to move them into the left list. Select OK.

1.10.4 Test Results

We can retrieve data from any of the available layers when the server is up.

1.11 Intergraph WFS

1.11.1 Operations Exercised

- WFS GetCapabilities
- WFS GetFeatureData

1.11.2 Known problems or limitations

We can not handle any coordinate system other than EPSG:4326. This limits us to the Beta WFS provided by Intergraph.

1.11.3 Test Procedures

1. Create new source
    1. Enter URL for Intergraph WFS
    2. Select OK
2. Create new layer
    1. Select Intergraph WFS as the source.
    2. Select a layer to display
    3. Select OK
3. Data will appear on screen.

4. After ensuring that objects from that layer are selectable, select a feature object. Data for that feature will appear in the feature attributes frame.

1.11.4 Test Results

- Successful WFS GetCapabilities
- Successful WFS GetFeatureData

1.11.5 Next Steps (Actions Required)

- Support other coordinate systems
- Support modifications to features

1.12 ubeWerx WCS

1.12.1 Operations Exercised

- WMS getCapabilities
- WMS getMap

1.12.2 Test Procedures

1. Select the Source tab of the client and select New
2. Enter the URL of the WMS based on the TWIKI. Select Refresh. View the results in the bottom text box. Select OK.
3. Select the Layer tab of the client and select New
4. Select the appropriate source in the drop down. Select the requested layers in the right tree that you want to view and select the arrow to move them into the left list. Select OK.

1.12.3 Test Results

Can not connect to server. Server does not implement version 0.4 or 1.0.3.

1.13 PCI WCS

1.13.1 Operations Exercised

- WMS getCapabilities
- WMS getMap

1.13.2 Test Procedures

1. Select the Source tab of the client and select New

2. Enter the URL of the WMS based on the TWIKI. Select Refresh. View the results in the bottom text box. Select OK.

3. Select the Layer tab of the client and select New

4. Select the appropriate source in the drop down. Select the requested layers in the right tree that you want to view and select the arrow to move them into the left list. Select OK.

1.13.3 Test Results

Can not connect to server. Server does not implement version 0.4 or 1.0.3.

1.14 Intergraph WCS

1.14.1 Operations Exercised

- WMS getCapabilities
- WMS getMap

1.14.2 Test Procedures

1. Select the Source tab of the client and select New

2. Enter the URL of the WMS based on the TWIKI. Select Refresh. View the results in the bottom text box. Select OK.

3. Select the Layer tab of the client and select New

4. Select the appropriate source in the drop down. Select the requested layers in the right tree that you want to view and select the arrow to move them into the left list. Select OK.

1.14.3 Test Results

We can retrieve Elevation Coverage for version 0.4, but the server does not implement version 1.0.3.

1.15 GMU WCS

1.15.1 Operations Exercised

- WMS getCapabilities
- WMS getMap

1.15.2 Test Procedures

1. Select the Source tab of the client and select New

2. Enter the URL of the WMS based on the TWIKI. Select Refresh. View the results in the bottom text box. Select OK.

3. Select the Layer tab of the client and select New

4. Select the appropriate source in the drop down. Select the requested layers in the right tree that you want to view and select the arrow to move them into the left list. Select OK.

1.15.3 Test Results

Can not connect to server. The server does not implement version 0.4 or 1.0.3.

1.16 Polexis SCS

1.16.1 Operations Exercised

- SCS GetCapabilities
- SCS GetObservation

1.16.2 Known problems or limitations

The sensor has been turned off. We can only get readings if we request data from a few weeks ago.

1.16.3 Test Procedures

1. Create new layer
2. Select Polexis SCS as the source.
3. Select OK
4. Data will appear on screen.
5. After ensuring that objects from that layer are selectable, select a sensor object. Data for that sensor will appear in the feature attributes frame.

1.16.4 Test Results

- Successful SCS GetCapabilities
- Successful SCS GetObservation
- Intermittent failure to connect

1.16.5 Next Steps (Actions Required)

None.

2 Intergraph

2.1 CubeWerx WFS

Intergraph's multi-source client has successfully parsed the CubeWerx WFS capabilities version 0.0.14 and loaded features from their dataset. In addition, WFS insert, delete and update transactions have been successfully tested. There is an outstanding issue with feature names which use non-alphanumeric characters, such as the parenthesis characters "(" and we are modifying our client to handle these better.

2.2 Ionic WFS

We have successfully parsed capabilities and loaded features from Ionic's WFS services. Transaction interoperability was not achieved due to the service requiring namespace support in the transaction XML and our client's inability to provide it.

2.3 Intergraph WFS

Complete functionality achieved.

2.4 GMU WCS

Intergraph's multi-source client has successfully parsed capabilities versions 0.5 and 0.6 and loaded coverages from their server in the NITF format.

2.5 UAH WCS

Intergraph's multi-source client has successfully parsed capabilities version 0.6 and loaded GOES satellite coverages in GeoTiff format.

2.6 PCI Geomatics WCS

Intergraph's multi-source client has successfully parsed capabilities version 0.5 and 0.6 and loaded coverages in GeoTiff and NITF format.

2.7 Intergraph IAS

Intergraph's multi-source client has successfully parsed WFS capabilities version 0.0.14 and loaded footprints, metadata, and images from its Image Archive Service.

3 Autodesk

Unless otherwise specified, operations Exercised: GetCapabilities, GetFeature/GetMap/GetRecord, as appropriate.

3.1 Galdos WFS

3.1.1 Final Status

Successful request of GML features parsed and rendered.

3.2 Galdos WRS

Galdos WRS requires version=0.7.1 parameter in order for the request=GetCapabilities to work.

- Client-side workaround utilized.

3.2.1 Final Status:

Successful request of registry entries; dynamically built queries returned things such as "all WFS services," etc.

3.3 Syncline WRS

17-Oct-02:

http://ogc-tie.syncline.com:8080/mapaccess/main.jsp?request=GetCapabilities

Produces Error 500.

Partial Stack dump:

```
Root cause:

java.lang.NoClassDefFoundError:
com/syncline/ows/server/common/MapAccessException at
java.lang.Class.getDeclaredConstructors0(Native Method)
```

3.3.1 Final Status:

Unable to complete testing cycle; server unavailable

3.4 Ionic WRS

14-Oct-02: Encountered invalid XML from

http://demo.ionicsoft.com/Registry/wrs/WRS?request=GetCapabilities

Verified in IE:

> XML page cannot be displayed
> Cannot view XML input using XSL style sheet. Please correct the error and then click the
> Refresh button, or try again later.
> ----------
>
> Reference to undeclared namespace prefix: 'xlink'. Error processing
> resource 'http://demo.ionicsoft.com/Registry/wrs/WRS?request=GetCapabilities'. Line 748,
> Position 100

>
>
xlink:href="http://schemas.cubewerx.com/schemas/ogcrim/0.7.0/ebRIM.xsd#Association Type1">

09-Jan-03: Server produces 404 - seems to be removed from service.

> 404 Not Found
> /Registry/wrs/WRS was not found on this server.
>
>
> ---
> Resin 2.1.6 (built Fri Nov 8 08:18:18 PST 2002)

3.4.1 Final Status:

Unable to complete testing cycle; XML produced by server isn't validating; server now unavailable

3.5 Cubewerx WMS

Capability-Document URL: http://demo.cubewerx.com/ows12/cubeserv/cubeserv.cgi

Initial tests show that we're able to connect without problems. Test Development to plan test suite.

3.5.1 Final Status:

Successful requests of maps. Many combinations of layers and options tried with success.

3.6 Intergraph WMS

Using "unadvertised" http://maps.intergraph.com/wms/london/GetCapabilities.asp -- difficulty digesting DOCTYPE with relative path dtd.

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<!DOCTYPE WMT_MS_Capabilities SYSTEM
"capabilities_1_0_0.dtd"> <WMT_MS_Capabilities
version="1.0.0" updateSequence="0">
<Service> ...
```

Now identifying relative paths.

3.6.1 Final Status:

Successful request of maps

3.7 Intergraph WFS

22-Oct-02 Identified that WFS GetFeaturehttp://member.opengis.org/portal/twiki_ows12/bin/edit/Main/GetFeature?topicparent=Main.AutoInt results links to incomplete (or irrelevant) XSD; the only intersection between feature types identified by the WFS and the elements defined in the XSD is States; consequently, feature types aren't recognized as features using rigorous algorithm, relaxed algorithm required to parse and identify objects.

```
<?xml version="1.0" encoding="UTF-8"?>
<wfs:featureCollection
   xmlns="http://ogc.intergraph.com/wfs"
   xmlns:gml="http://www.opengis.net/gml"
   xmlns:wfs="http://ogc.intergraph.com/wfs"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://OGC/temp GMLData.xsd">
```

qv http://ogc.intergraph.com/alpha_wfs/GMLData.xsd

17-Oct-02 Successful TIE with Intergraph at the advertised location http://ogc.intergraph.com/alpha_wfs/request.asp -- Previously encountered invalid XML (namespace not defined prior to use.)

3.7.1 Final Status:

Successfully requested and received GML, however the document did not contain a completely descriptive schema, so parsing/feature recognition was impaired.

3.8 CAST WMS

Capability-Document URL: http://kirk.cast.uark.edu:9080/ows/wms

Initial tests show that we're able to connect without problems. Test Development to plan test suite.

3.8.1 Final Status:

Successful request of maps.

3.9 Polexis WRS

Issue with inconsistencies in implementation, however client-side workaround was implemented that successfully circumvented the problem. As such, behaved very similar to Galdos' WRS.

3.9.1 Final Status:

Successful request of registry; dynamically built queries were accepted and returned valid results, however the registry had not been populated with entries that matched the criteria so empty result sets were returned.

3.10 Cubewerx WRS

3.10.1 Final Status:

Successful unconstrained request of registry entries; structured queries akin to those sent to Galdos and Polexis generated exceptions. Further investigation into query syntax to formulate an interoperable query is indicated.

4 UAH

4.1 UAH STT client to WMS servers

4.1.1 Description

The Space Time Toolkit (STT) client is accessing images from several clients using the OGC WMS interface

4.1.2 Operations Exercised

- WMS getCapabilities
- WMS getMap

4.1.3 TIE Partners

- Clark DLGs - *http://kirk.cast.uark.edu:9080/ows/wms* (Transportation and hydro)
- CubeWerx - *http://demo.cubewerx.com/ows12/cubeserv/cubeserv.cgi*http://demo.cubewerx.com/ows12/cubeserv/cubeserv.cgi%3C/EM%3E%3CEM%3E (Builtups, topo, etc.)
- CubeWerx - *http://demo.cubewerx.com/ows1/cubeserv/cubeserv.cgi* (NY Orthos, hydro, others)
- Intergraph - *http://ogc.intergraph.com/alpha/request.asp* (LaPlatahttp://member.opengis.org/portal/twiki_ows12/bin/edit/Main/LaPlata?topicparent=Main.UahWms DOQQs)
- JPL - *http://wms.jpl.nasa.gov/wms.cgi* (Landsat Mosaic)

4.1.4 TIE Components

Space Time Toolkit client (*http://vast.uah.edu/SpaceTimeToolkit*)

4.1.5 How to use

Install STT from link below. Open LaPlatahttp://member.opengis.org/portal/twiki_ows12/bin/edit/Main/LaPlata?topicparent=Main.UahWms Project, enable appropriate data item by selecting in data tree, and clicking on enable in data customizer panel (or double click on selection in data tree)

4.1.6 Physical configuration (data, software, hardware environment)

Java JDK 1.3 Run Time Environment

4.1.7 Test Procedures

URLs and layer names entered into STT Project resource files. User enables these data items through STT interface

4.1.8 Test Description.

URLs and layer names entered into STT Project resource files. User enables these data items through STT interface

4.1.9 Test Results

All listed interactions returned fine.

4.2 UAH STT client to WFS servers

4.2.1 Description

The Space Time Toolkit (STT) client is accessing images from several clients using the OGC WFS interface

4.2.2 Operations Exercised

- WFS getFeatureDescription
- WFS getFeature

4.2.3 TIE Partners

- Ionic WFS server (Maryland Counties)
- Intergraph WFS Server (LaPlata Snapshots)
- Intergraph WFS Server (La Plata DOQQs)

4.2.4 TIE Components

Space Time Toolkit client (*http://vast.uah.edu/SpaceTimeToolkit*)

4.2.5 How to use

Install STT from link below. Open LaPlata Project, enable appropriate data item by selecting in data tree, and clicking on enable in data customizer panel (or double click on selection in data tree)

4.2.6 Physical configuration (data, software, hardware environment)

Java JDK 1.3 Run Time Environment

4.2.7 Test Procedures

URLs and layer names entered into STT Project resource files. User enables these data items through STT interface

4.2.8 Test Description.

URLs and layer names entered into STT Project resource files. User enables these data items through STT interface

4.2.9 Test Results

All listed interactions returned fine.

4.3 UAH STT client to WCS servers

4.3.1 Description

The Space Time Toolkit (STT) client is accessing images from several clients using the OGC WCS interface

4.3.2 Operations Exercised

- WCS getCapabilities
- WCS getCoverage

4.3.3 TIE Partners

- UAH - *http://stromboli.nsstc.uah.edu:8080/sttserv/servlet/DopplerServlet* (WSR88 Doppler Radar - UAH JSO)
- UAH - *http://stromboli.nsstc.uah.edu:8080/sttserv/servlet/GoesServlet*http://stromboli.nsstc.uah.edu:8080/sttserv/servlet/GoesServlet%3C/EM%3E%3CEM%3E (GOES Weather satellite - GeoTIFFhttp://member.opengis.org/portal/twiki_ows12/bin/edit/Main/GeoTIFF?topicparent=Main.UahWcs)
- UAH - http://stromboli.nsstc.uah.edu:8080/sttserv/servlet/NldnServlet (National Lightning Detection Network - UAH JSO)

4.3.4 TIE Components

Space Time Toolkit client (*http://vast.uah.edu/SpaceTimeToolkit*)

4.3.5 How to use

Install STT from link below. Open LaPlata Project, enable appropriate data item by selecting in data tree, and clicking on enable in data customizer panel (or double click on selection in data tree)

4.3.6 Physical configuration (data, software, hardware environment)

Java JDK 1.3 Run Time Environment

4.3.7 Test Procedures

URLs and layer names entered into STT Project resource files. User enables these data items through STT interface

4.3.8 Test Description.

URLs and layer names entered into STT Project resource files. User enables these data items through STT interface

4.3.9 Test Results

All listed interactions returned fine.

5 GMU

5.1 CubeWerx WRS

5.1.1 Operations Exercised

- WRS GetRecord, to find dataset for a area within the specified bounding box

5.1.2 Test Procedures

1. Select CubeWerx WRS server, URL is http://demo.cubewerx.com/ows12/wrs0/cwwrs.cgi
2. Specify the bounding box and coordinate's name
3. Find data within the bounding box
4. Add some result into project as a layer

5.1.3 Test Results

finds out some data on CubeWerx WMS successfully

5.2 CubeWerx WMS

5.2.1 Operations Exercised

- GetCapabilities
- GetFeatureInfo
- GetMap

5.2.2 Test Procedures

1. Select CubeWerx WMS server, URL is http://demo.cubewerx.com/ows12/wfs/cwwfs.cgi
2. Get WMS capabilities
3. Select some maps in the capabilities

4. Get maps
5. Display map, select features in the map to get feature information

5.2.3 Test Results

- Get capabilities on CubeWerx WMS successfully
- Get map on CubeWerx WMS successfully
- Get feature info on CubeWerx WMS successfully

5.3 Intergraph WMS

5.3.1 5.3.1 Operations Exercised

- GetCapabilities
- GetMap

5.3.2 Test Procedures

1. Select Intergraph WMS server, URL is http://ogc.intergraph.com/alpha/request.asp
2. Get WMS capabilities
3. Select some maps in the capabilities
4. Get maps
5. Display map

5.3.3 Test Results

- Get Capabilities on Intergraph WMS successfully
- Get map on Intergraph WMS successfully

5.4 CAST WMS

5.4.1 Operations Exercised

- GetCapabilities
- GetMap

5.4.2 Test Procedures

1. Select CAST WMS server, URL is http://kirk.cast.uark.edu:9080/ows/wms
2. Get WMS capabilities
3. Select some maps in the capabilities
4. Get maps
5. Display map

5.4.3 Test Results

- Get capabilities on CAST WMS successfully
- Get map on CAST WMS successfully

5.5 CubeWerx WFS

5.5.1 Operations Exercised

- GetCapabilities
- DescribeFeatureType
- GetFeature

5.5.2 Test Procedures

1. Select CubeWerx WFS server, URL is http://demo.cubewerx.com/ows12/wfs/cwwfs.cgi
2. Get WFS capabilities
3. Select a feature to get its schema by DescribeFeatureType request
4. Get feature data according to its schema
5. Display feature, select features in the map to get attribute information

5.5.3 Test Results

- Get capabilities on CubeWerx WFS successfully
- Get DescribeFeatureInfo on CubeWerx WFS successfully
- Get feature on CubeWerx WFS successfully

5.6 Ionic WFS

5.6.1 5.6.1 Operations Exercised

- GetCapabilities
- DescribeFeatureType
- GetFeature
- Transaction

5.6.2 Test Procedures

1. Select Ionic WFS server, URL is http://demo.ionicsoft.com/owsData/wfs/CHARLESCOUNTY and http://demo.ionicsoft.com/owsData/wfs/DAMAGEDAREA
2. Get WFS capabilities
3. Select a feature to get its schema by DescribeFeatureType request
4. Get feature data according to its schema
5. Display feature, select features in the map to get attribute information
6. Digitize damaged area and produce a feature layer encoding in GML
7. Submit it to Ionic transaction WFS

5.6.3 Test Results

- Get capabilities on Ionic successfully
- Get DescribeFeatureInfo on Ionic WFS successfully
- Get feature on Ionic WFS successfully
- Insert feature into Ionic WFS by transaction

5.7 Intergraph WFS

5.7.1 Operations Exercised

- GetCapabilities
- DescribeFeatureType
- GetFeature

5.7.2 Test Procedures

1. Select Intergraph WFS server, URL is
http://ogc.intergraph.com/OWS_Demo/request.asp
2. Get WFS capabilities
3. Select a feature to get its schema by DescribeFeatureType request
4. Get feature data according to its schema
5. Display feature, select features in the map to get attribute information

5.7.3 Test Results

- Get capabilities on Intergraph successfully
- Get DescribeFeatureInfo on Intergraph WFS successfully
- Get feature on Intergraph WFS successfully

6 Polexis

6.1 Galdos WRS

POSTing the following query:
<?xml version='1.0' encoding='UTF-8'?>
<wrs:GetRecord xmlns="urn:oasis:names:tc:ebxml-regrep:rim:xsd:2.1"
xmlns:wrs="http://www.opengis.net/wrs" xmlns:ogc="http://www.opengis.net/ogc"
outputFormat="XML">
  <wrs:Query typeName="Service">
    <ogc:PropertyName>/Service</ogc:PropertyName>
  </wrs:Query>
</wrs:GetRecord>
to http://dali.galdosinc.com:80//registry/wrs
requires that the HTTP Content-type header be set to "text/xml". If that is set, the response contains information about 2 services.

6.2 Syncline WRS

POSTing this request:
<?xml version='1.0' encoding='UTF-8'?>
<wrs:GetRecord xmlns="urn:oasis:names:tc:ebxml-regrep:rim:xsd:2.1"
xmlns:wrs="http://www.opengis.net/wrs" xmlns:ogc="http://www.opengis.net/ogc"
outputFormat="XML">
  <wrs:Query typeName="Service">
    <ogc:PropertyName>/Service</ogc:PropertyName>

</wrs:Query>
</wrs:GetRecord>
to http://ogc-tie.syncline.com:8080//mapaccess/main.jsp
requires the client to accept two cookies. Then nothing further happens.

6.3 Ionic WRS

The following request:
<?xml version='1.0' encoding='UTF-8'?><wrs:GetRecord
xmlns="urn:oasis:names:tc:ebxml-regrep:rim:xsd:2.1"
xmlns:wrs="http://www.opengis.net/wrs" xmlns:ogc="http://www.opengis.net/ogc"
outputFormat="XML">
<wrs:Query typeName="Service">
<ogc:PropertyName>/Service</ogc:PropertyName>
</wrs:Query>
</wrs:GetRecord>

returns 500 Internal Server Error

6.4 PCI WRS

No URL is listed for the PCI registry.

Polexis used the client to upload the following file:

C:\Program Files\Polexis\Vigilys 1.2\viking\root\registeredimages\demoArea.gif

with these parameters:

image/gif urn:opengis:services:image-archive:roles:data 0.0.1 content1036708346493

and received this response:

<?xml version="1.0" encoding="UTF-8"?> <ia:TransactionResponse
xmlns:ia="http://www.opengis.net/iarchive"> <ia:InsertResult
handle="content1036708346493"> <ia:InsertedContent msg-content-
ref="content1036708346493" oid="urn:uuid:09471b2c-ad38-4d0b-b3f0-b408b99aea8b"
/> <ia:Status>SUCCESS</ia:Status> </ia:InsertResult>
<ia:Status>SUCCESS</ia:Status> <ia:Message>All operations succeeded (Count
=1)</ia:Message> </ia:TransactionResponse>

Then we used the URL of the inserted image in a HTTP GET request as follows:

http://gws.pcigeomatics.com/ia?request=GetObjectByID&objectid=urn:uuid:09471b2c-ad38-4d0b-b3f0-b408b99aea8b

The result was the original image.

The mime type of the response from the GET request was wrong. This was reported to PCI, who report that that is work in progress.

6.5 CubeWerx WRS

This request:

```
<?xml version='1.0' encoding='UTF-8'?><wrs:GetRecord
xmlns="urn:oasis:names:tc:ebxml-regrep:rim:xsd:2.1"
xmlns:wrs="http://www.opengis.net/wrs" xmlns:ogc="http://www.opengis.net/ogc"
outputFormat="XML">
  <wrs:Query typeName="Service">
    <ogc:PropertyName>/Service</ogc:PropertyName>
  </wrs:Query>
</wrs:GetRecord>
```

returns this response:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<ServiceExceptionReport version="1.2.0">
   <ServiceException>
CWWRS-61394: No error/message description available (raised in function
        wrsParse() of file "wrsParse.c" line 274)
CWWRS-61104: No error/message description available (raised in function
        wrsExecGetRecord() of file "wrsGetRecord.c" line 1033)
CWWRS-61438: No error/message description available (raised in function
        wrsExecQuery() of file "wrsGetRecord.c" line 967)
CWWRS-17000: ORA-00904: invalid column name (raised in function cwdbsExecute()
        of file "cwdbs.pc" line 470)
</ServiceException>
</ServiceExceptionReport>
```

6.6 CubeWerx WMS

The Polexis client has successfully tied with the Cubewerx SLD enabled WMS and used a style from the Polexis SMS to restyle a layer from the WMS.

7 Ionic

7.1.1 Operations Exercised:

- WFS
  - GetCapabilities
  - DescribeFeatureType
  - GetFeature
- WMS
  - GetCapabilities

- GetMap
- GetFeatureInfo

- SCS
  - GetObservation

7.1.2 TIE Partners:

WFS	*Intergraph*	http://ogc.intergraph.com/Alpha_wfs/Request.asp
	Cubewerx	http://demo.cubewerx.com/ows12/wfs/cwwfs.cgi
	Galdos	http://wfs.galdosinc.com:8980/wfs/http
WMS	*NASA*	http://globe.digitalearth.gov/viz-bin/wmt.cgi
	CAST	http://ogc.cast.uark.edu:8080/ows12/wms
	Cubewerx	http://demo.cubewerx.com/ows12/cubeserv/cubeserv.cgi
	Intergraph	http://ogc.intergraph.com/alpha/request.asp
SCS	*Polexis*	http://ogc.polexis.com/airquality/scs
UDDI	*Nasa*	http://sindbad.gsfc.nasa.gov:8080/uddi/inquiry

7.1.3 Client Architecture:

The client architecture is based on the following:

- Classical Web Browser as Netscape, Internet Explorer,…
- The Internet through standard HTML
- J2EE Application Server with a Web Application developed using Ionic components.
- OGC Protocols based on HTTP/XML
- OGC Web Services provided by Ionic Server Components or other participant services

7.1.4 How to use the client

The client is used through a standard web browser; just connect it to http://demo.ionicsoft.com/owsDemo. The client has multiple tabs where you can:

- Search a place: perform a Gazetteer search using GNS multimillion points dataset.
- Add service: add any WMS service from either a URL or an entry in Ionic Web Registry Service.
- Handle layers: perform common navigation tasks as pan, zoom, layer up/down, layer visibility,…
- Contexts: handle contexts, you can save/load it, send it as a mail attachment…
- Report: this is a real time generated report based on WFS features intersected by the damaged area feature displayed in the default context as a red area.
- Client config: some configuration parameter for the client.

7.1.5 Physical configuration of data, software and hardware

The end user access client is a standard web browser (IE, Netscape,…).

The middle tier application is a standard J2EE Web Application and may be deployed in any J2EE compatible environment, such as Tomcat 4.0.x, BEA WebLogic 6.1 or Resin application server.

7.1.6 Known problems or un-implemented features

The Sensor Collection Service access has been added to the client component. Unfortunately, we have not yet updated the user interface to allow access to this kind of service.

7.1.7 Online services

The interoperability is brought to the application by the use of Ionic client components. These components have the ability to connect to any compliant WFS and WMS supporting one of the following specification versions:

- WMS 1.0, 1.1.0, 1.1.1
- WFS 0.0.13, 0.0.14, 1.0.0

For the purpose of this demo we have also added the ability to connect to Sensor Collection Services, Web Service Registries and UDDI Registries.

7.1.8 Results, observations, and lessons learned

For WMS, the result is really encouraging. Almost any available WMS has been accessed without any problem. The only remaining problem is regarding to the publication by the services of the spatial reference system in which underlying datasets are available. See next section for more details.

For WFS, more work has been done to interoperate. The main issue was the compliance of the feature type descriptions published by the services. They were

either wrong or not complete. Once schema issues were resolved, getting and parsing, on the fly discovered features, goes straightforward.

For SCS, the most important fact is that it uses GML as its result encoding. So accessing SCS was just a matter of sending a valid request. The result has been easily integrated to the other components thanks to the use of GML.

For WRS, the interface to access and query registries was successfully used, but the attempt to get useful information back from other vendor registries failed. Actually, differences in the use of the OGC registry information model lead to inconsistency of the use of WRS. This issue has been partially addressed in the latest version of the WRS IPR but was not tested during demo process.

For UDDI, interaction with NASA UDDI registry was successful. The RPC-like interface leads to a very short development effort. The drawback is that the UDDI registry model is too weak to achieve use cases like the ones achieve thanks to the WRS.

7.1.9 Recommend changes

OGC Service:

One important issue we have encounter is the integration of data coming from servers that are not using the same spatial reference system. We had to perform on the fly coordinate transformation of the datasets. Unfortunately, there is no way to know the initial SRS in which the data are stored in the service. This may lead to multiple transformations and therefore to accuracy and performance problem.

WRS:

The filter encoding used in the WRS does not handle all the fine grained queries required by the OGC Registry Information Model. Especially, the complex properties are not handled in an efficient way.

WFS:

The XML schema definition is a very difficult process and many other participant implementations do not provide a valid schema.

Some WFS also define their feature type using draft GML 3 schemas. We came to the conclusion that we do need a way to specify which schema version the client want to use.

SCS:

The only real problem we encounter is regarding the EPSG 4326 definition. The order in which coordinates are encoded seems not clear enough (x,y => lat,long or long,lat ?).