Open Geospatial Consortium, Inc.

Date: 2006-01-29

Reference number of this OGC project document: **OGC 05-116**

Version: 0.0.3

Category: OGC Interoperability Program Report

Editors: Stan Tillman and Jody Garnett

OWS Integrated Client

Architecture, Design, and Experience

Warning

Document type:	OGC Interoperability Program Report
Document stage:	Draft
Document language:	English

Contents

i. Preface

The Open Geospatial Consortium (OGC) is an international voluntary consensus standards organization of more than 300 companies, government agencies, and universities. This OGC 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 3 (OWS 3) 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-3 portal message, email message, or by making suggested changes in an edited copy of this document.

ii. Submitting organizations

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

Intergraph Corporation
Refractions Research

iii. Submission contact points

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

CONTACT	COMPANY	EMAIL

Stan Tillman	Intergraph Mapping and GIS Solutions	stan.tillman@intergraph.com
Jody Garnett	Refractions Research Inc.	jgarnett@refractions.net

iv. Revision history

Date	Release	Author	Paragraph modified	Description
31 Aug 2005	0.0.1	Tillman	-	Initial version.
27 Oct 2005	0.0.2	Tillman	Document	Merged content from Refractions
04 Nov 2005	0.0.3	Tillman	Document	First Release
29 Jan 2006	0.0.4	Carl Reed	Document	Copyright, general edits

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 05-116) is an Interoperability Program Report (IPR) that reflects work carried out during the OGC Web Services Initiative, Thread Set 3. In the OWS-3 RFP, it was stated that this document may replace the discussion paper published from the OWS-1.2 experiments. However, the editors and contributors of this document felt that much of the information from the original work was still very beneficial to those developing integrated clients. For this reason, much of the information was retained and used in various sections of this document. This paragraph is meant to give credit to those involved in the previous effort.

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. This IPR contains an informative annex.

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 3 (OWS-3) 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 3 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)
- Video data (MPEG4)
- Sensor Web data (XML) – (Intergraph Client Only)

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 four of the above source categories simultaneously.

This IPR will include integrated clients which utilize most or all of the following specifications: Web Map Server (WMS), Web Feature Server (WFS), Feature Portrayal Service (FPS), Web Coverage Server (WCS), Catalog Service – Web (CS/W), Data Aggregation Service (DAS), GeoVideo Service (GVS), Sensor Planning Service (SPS), Sensor Observation Service (SOS), and Context Documents.

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 3 participants in creating such clients. In fulfillment of these goals, the IPR includes:

- Definitions of the common terms associated with the effort.

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

    B) 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.

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

    D) 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.

    E) And a discussion of the key accomplishments and lessons learned by the OGC members participating in the development of an integrated client for the OWS 3 effort.

2 Normative References

The following normative documents contain provisions that, through reference in this text, constitute provisions of this document. For dated references, subsequent amendments to, or revisions of, any of these publications do not apply. However, parties to agreements based on this specification are encouraged to investigate the possibility of applying the most recent editions of the normative documents indicated below. For undated references, the latest edition of the normative document referred to applies.

Integrated Client for Multiple OGC-compliant Services, Version 0.1.18, OGC Document #03-021, 20 January 2003

The OpenGIS Abstract Specification Topic 12: Open GIS Service Architecture, Version 4.3, OGC Document #02-112, 19 September 2001

OGC Catalogue Services – ebRIM. Version 0.9.1, OGC Document #04-017r1, 12 October 2004

OpenGIS® Catalogue Service Implementation Specification (CAT/CS-W). Version 2.0, OGC Document #04-021r2, 2 August 2004

OGC Web Services Context Documents (OWS Context) Interoptability Experiment, Version 0.0.3, Document #OGC-05-062, 11 August 2005

OpenGIS® Web Map Context Implementation Specification (WMC), Version 1.1, OGC Document #05-005, 3 May 2005

Sensor Observation Service, (OGC Document 05-088r1), 31 October 2005

Sensor Planning Service, (OGC Document 05-089), 18 October 2005

Web Coverage Service Implementation Specification, Version 1.0.0 (OGC Document 03-065r6), 16 October 2003

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

Web Feature Service Implementation Specification, Version 1.1.0 (OGC Document 04-094), 03 May 2005

Level 0 Profile of GML3 for WFS (Level 0), 0.0.10, OGC Document #03-003r10, 10 May 2004

Web Map Service Implementation Specification, Version 1.0.0 (OGC Document 00-028), 19, April 2000

Web Map Service Implementation Specification, Version 1.1.0 (OGC Document 01-047r02), 21 June 2001

Web Map Service Implementation Specification, Version 1.1.1 (OGC Document 01-068r03), 18 April 2002

Web Map Service Implementation Specification, Version 1.3.0 (OGC Document 04-024), 02 August 2004

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 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. Often a web client.

4 Conventions

The following sections define the conventions used in this document.

4.1 Symbols (and abbreviated terms)

The following symbols, acronyms and abbreviations are used in this document:

1D	One Dimensional
2D	Two Dimensional
3D	Three Dimensional
4D	Four Dimensional
API	Application Program Interface
CORBA	Common Object Request Broker Architecture
COTS	Commercial Off The Shelf
CPS	Coverage Portrayal Service
CSW	Catalog Service Web (aka Catalog 2.0)
DAS	Data Aggregation Service
DCE	Distributed Computing Environment
DCP	Distributed Computing Platform
DRM	Digital Rights Management
FPS	Feature Portrayal Service
GML	Geographic Markup Language
I&A	Identification and Authentication
ISO	International Organization for Standardization
OGC	Open Geospatial Consortium
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
WSDL	Web Service Description Language
XML	Extensible Markup Language

4.2 UML Notation

The diagrams that appear in this document are presented using the Unified Modeling Language (UML). Both static structure diagram and activity diagrams will be used as appropriate.

4.2.1 UML Static Structure Diagrams

The UML notations used for static structure diagrams in this document are described in the diagram below.

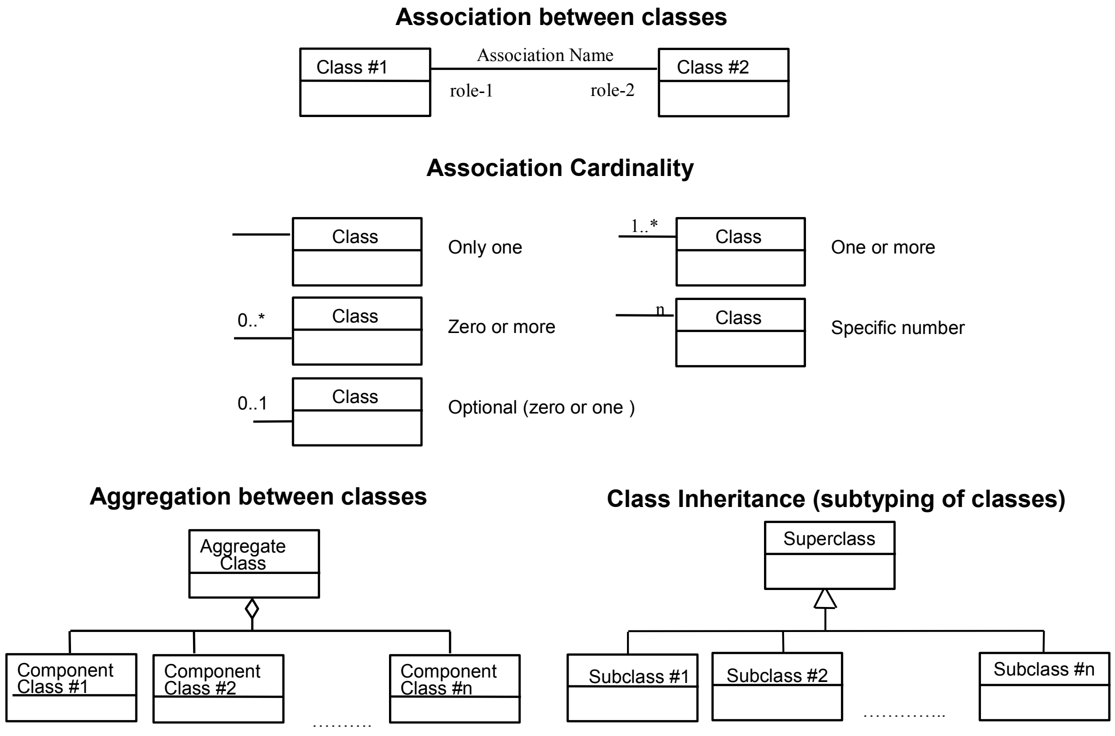


Figure 1 UML Static Structure Notation

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

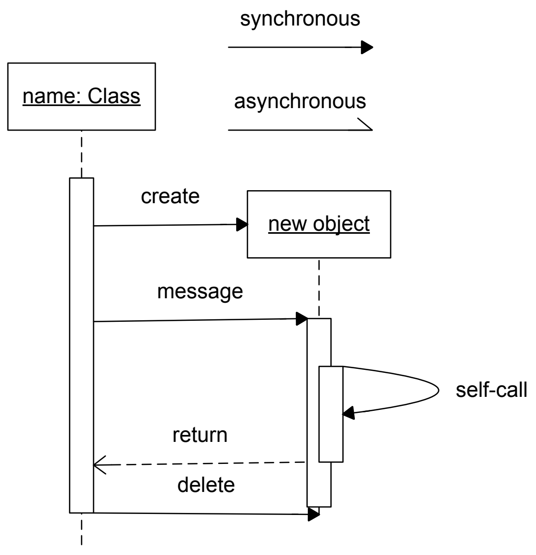
a) <<Interface>> A definition of a set of operations that is supported by objects having this interface. An Interface class cannot contain any attributes.

a) <<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.

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

4.2.2 UML Sequence Diagrams

UML Activity Diagrams are helpful in explaining workflow. Given the nature of this document these diagrams will focus on the use of Sequence diagrams.



*: iteration message ()

[condition] message ()

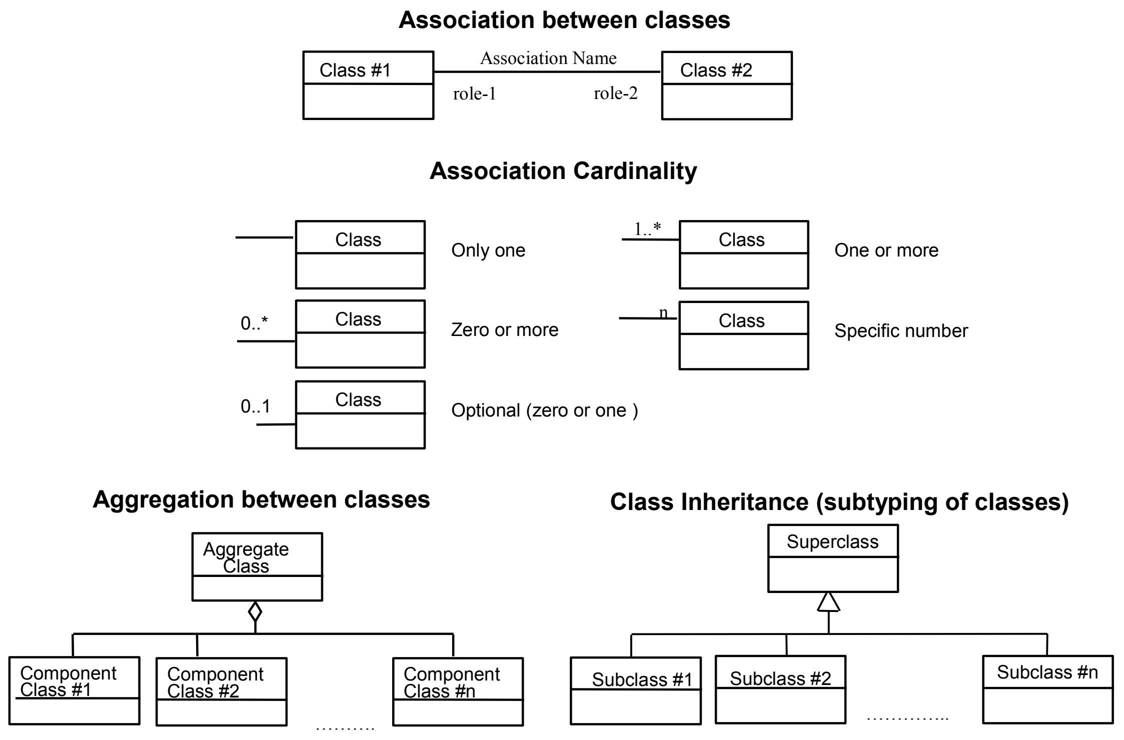


Figure 2 UML Sequence Diagram Notation

We have extended the basic UML notation to give some indication of streaming access. This will be indicated by multiple return arrows for a single message.

4.3 Use of Patterns

In several cases the text and UML diagrams have been instrumented with Pattern notation to convey architectural motivation where appropriate. In this document the following patterns are referenced.

b) Extensible Interface; allows the support of additional interfaces in a dynamic manner.

c) Proxy; provide a surrogate or placeholder for another object to control access to it

d) Bridge; decouple an abstraction from its implementation so that the two can vary independently

e) Strategy; define a family of algorithms, encapsulate each one, and make them interchangeable.

f) Abstract Factory; provide an interface for creating families of related or dependent objects without specifying their concrete classes.

g) Null Object; act a surrogate for the lack of an object of a given type

Familiarity with patterns will assist in a deeper understanding of this document. The text description of architecture and design will be limited to the immediate concerns of an integrated client. The use of patterns allow us to communicate many additional tradeoffs we will not have time to cover in detail.

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, video and sensor web data sources within a single client. 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.

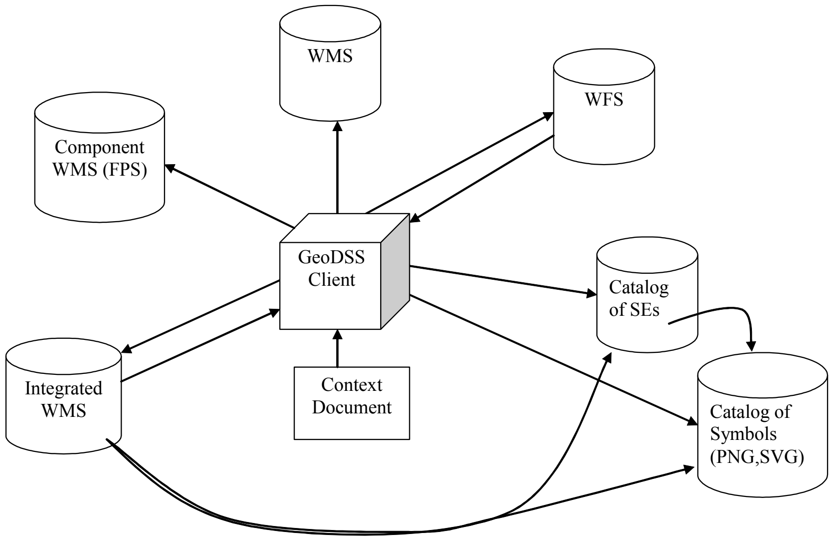


Figure 3 GeoDSS Client

For the purpose of this OWS-3 project we have focused on the creation of a Geographic Decision Support System client.

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.

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

Once geospatial data is published in an OGC web service instance (W*S), its presence must be announced so that geospatial data analysts can find it. A geospatial data provider can do this by using an integrated client to register the service with a catalog. A URL endpoint pertaining to the service is sent the catalog service. When the catalog receives the request, it then queries the W*S service for its capabilities.

B) Querying a Service Registry for OGC Web Services.

A geospatial analyst must be able to locate OGC Web Services. The analyst can use an integrated client to query a Catalogue Service – Web (CS-W) for available services based on location and other parameters. The CS-W 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.

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 geospatial 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 that 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.

E) Managing a collection of Service Registries.

Instead of querying a specific CS-W, an integrated client could potentially query multiple CS-W 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/Expoitation

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, DAS. 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. For example, a geospatial 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.

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.

5.4 Sensor Web Production

A number of remote sensors, both 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 that are connected to the Internet. The Sensor Web Exploitation functionality requires support of some or all of the following service types: SPS, SOS, WNS. It can be divided into the following 3 functions:

A) Retrieving sensor data from a SOS.

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

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 utilize the SOS functionality to retrieve the data.

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.

a) Web Map Service Context Document

The present Context specification is known as a "Web Map Context Document," or simply a "WMS 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.

b) Open Web Service Context Document

The Open Web Service Context Document is a product of a recent Interoptability Experiment (). An "OWS Context" allows for the definition of a Map via Web Coverage Servers, Web Feature Servers and Web Map Servers.

This document is constructed with the assumption that the Open Web Service will need to be contacted for the additional information available in its Capabilities document.

This technology does represent a work in progress, and this document contains several recommendations aimed at refining this work.

5.5.1 Use of Context Documents

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 WMS 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 OWS Context document assumes the OWS Server will need to be contacted.
- 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.

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

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

6.3 Project creation, storage, loading, and sharing

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

6.4 OGC Specifications

- OGC specifications relevant to externally accessed services, or consistent with versions under development during the OWS-3 Testbed.

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) 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, AJAX, Applets/Active X, Browser Plug-ins), 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. Thick client products can sometimes draw upon OGC conformant server-side components to provide such functionality.

Thick clients can also enable value-adding and data manipulation functions that are difficult to replicate in a browser. The recent advances with AJAX have raised user expectations of thin client software. It is indicative of this change that many thin client implementations have been rewritten over the course of the last year.

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. 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 one does 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.

Usability is also dependent on the application's ability to supply sensible defaults. In many practical situations this boils down to the quality of the specification being leveraged, and the availability of appropriate metadata.

7.1.5 Security and Digital Rights Management

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.

This document covers the use of Digital Rights Management (DRM) with respect to those services encountered during the OWS-3 project.

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 most 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 Interaction with Open Web Services

This section concentrates on the use of Open Web Services, in particular constructing your application to work in the face of the many services and standards currently deployed.

7.2.1 Representation of Open Web Services

When locating information, either directly at the bequest of a user or via a catalogue search, you will often find the same information available through several workflows. As an example a Catalogue search will reveal this information through multiple service associations.

In addition you can often safely inspect the following:

- Sending a WFS Capabilities request to a provided WMS end point (most commercial and public domain WMS implementations allow for this)
- Inspection of the WMS DescribeLayer information for an available a WFS end-point
- Knowledge of the popular implementations

Traditionally there are several approaches to working with remote services (or resources in general).

A) Representation by Proxy

The remote service may be represented by use of a Proxy, as shown by CORBRA or recent XML SOA Binding technologies. The use of a Proxy class results in a viable, direct representation of an external service.

A Proxy class is limited to representing a single OWS Service, and is problematic as services availability and version change over time.

B) Representation by Handle

The remote service may also be managed through the concept of a resource handle. By not representing a service explicitly we gain the ability to lazily bind to the service, and may represent a service that is unavailable or not yet created.

An advantage of the resource handle approach is the ability to offer a binding using a workflow appropriate API. This flexibility allows your architecture to adapt to future considerations such as the addition of additional services and workflows based on the same information.

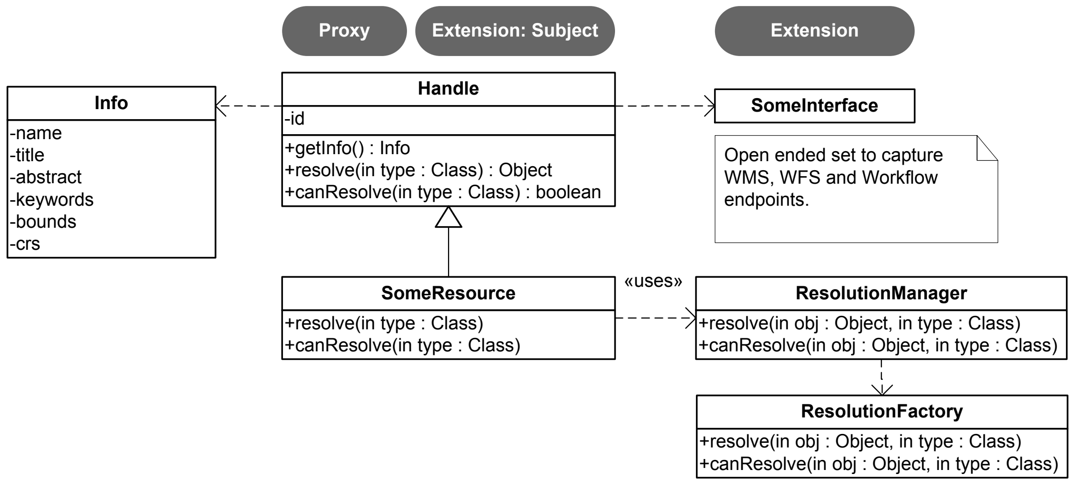

Proxy
Extension: Subject
Extension
Info
-name
-title
-abstract
-keywords
-bounds
-crs
Handle
-id
+getInfo() : Info
+resolve(in type : Class) : Object
+canResolve(in type : Class) : boolean
SomeInterface
Open ended set to capture WMS, WFS and Workflow endpoints.
SomeResource
+resolve(in type : Class)
+canResolve(in type : Class)
«uses»
ResolutionManager
+resolve(in obj : Object, in type : Class)
+canResolve(in obj : Object, in type : Class)
ResolutionFactory
+resolve(in obj : Object, in type : Class)
+canResolve(in obj : Object, in type : Class)

Figure 4 Resource Handle

In the above diagram a Handle offers access to metadata information (represented by Info in the above diagram). The implementation of a handle will vary according to source (the results of a Catalogue search, or the processing of a capabilities document). A ResolutionManager indicates the availability of a plug-in system to define additional resource bindings.

7.2.2 Use of Multiple Services

The ability to handle several workflows at once against the same logical data is an important aspect of a complete OWS integrated client.

A) Access to Multiple Services

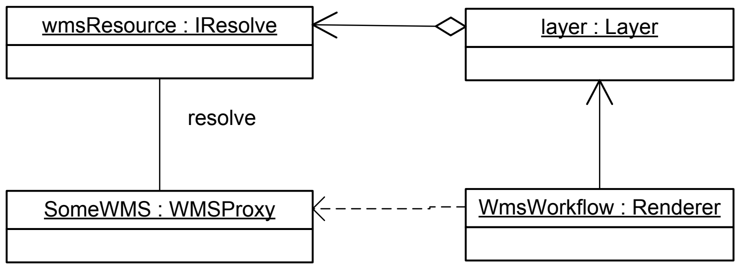


Figure 5 Resource Handle Use (WMS)

In the above diagram illustrates the use of a resource handle with a single WMS workflow. The WMS workflow has resolved the resource handle into the specific WMSProxy required (presumably for a GetMap request).

An editing workflow may also be created, making use of a Web Feature Server providing access to the same information.

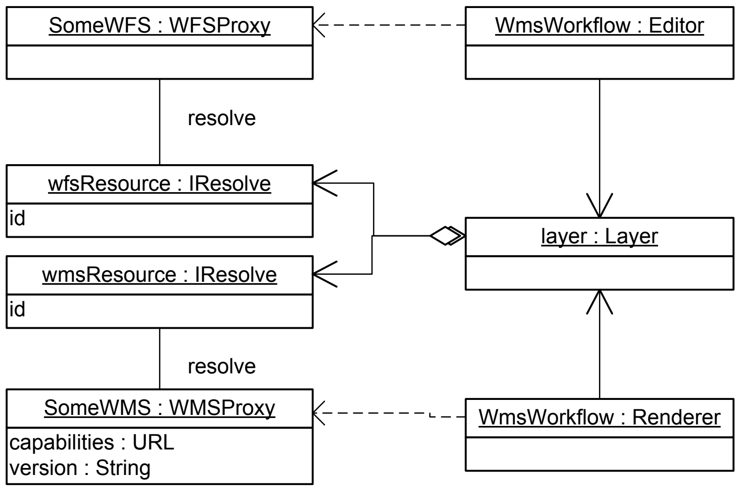


Figure 6 Resource Handle Use with Multiple Workflows

We have added a WFS workflow (an Editor), which has located a wfsResource and resolved it to the appropriate proxy.

7.2.3 Use of Workflows

Use of an integrated client will often involve the use of a variety of workflows. There are several approaches that may be considered to support this functionality.

A) Direct Binding Workflow

The use of direct binding, in the classic client/server sense should not be underestimated. Open Web Services are designed with a larger publish/find/bind system in mind, but are usable in isolation. Section 10 onward will provided suitable examples of direct access in this manner.

B) Service Chaining Workflow

Open web services may be combined together into a service chain. From the perspective of a client developer most of these chains are “opaque” and appear for all the world as a normal Open Web Service. The degree to which this opacity is maintained is a cause for concern when considering the consequences of Digital Rights Management, and cascading web map servers.

A service chain constructed by an integrated client is considered “transparent”. For a worked example of transparent chaining please review the use of Feature Portrayal Service discussion in Section 14 of this document.

C) Adaptive Workflow

The use of metadata allows for an adaptive approach to workflow. The use of metadata combined with simple heuristics can allow an application to respond to users needs appropriately using available services.

To explore these ideas consider the following example constructed around user supplied metadata (style information) and rendering.

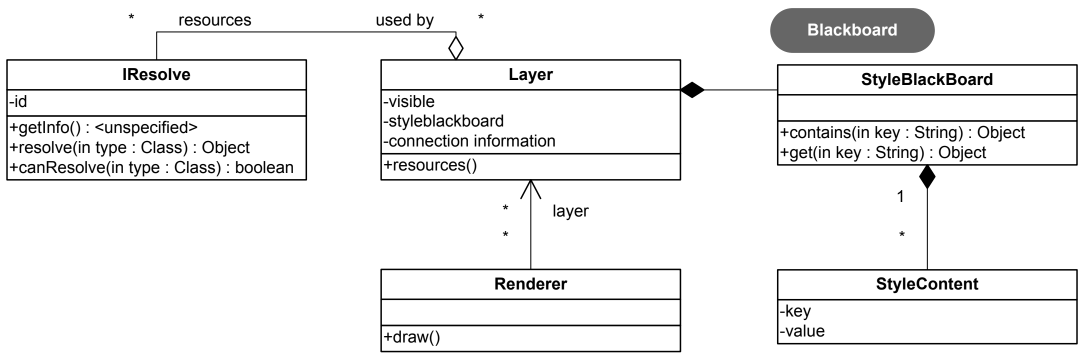


Figure 7 Metadata-Controlled Adaptive Workflow

Given the dynamic nature of a user controlled application we will be using a blackboard to record this information. Selection of an appropriate workflow based on metadata may be considered a matter for a voting algorithm.

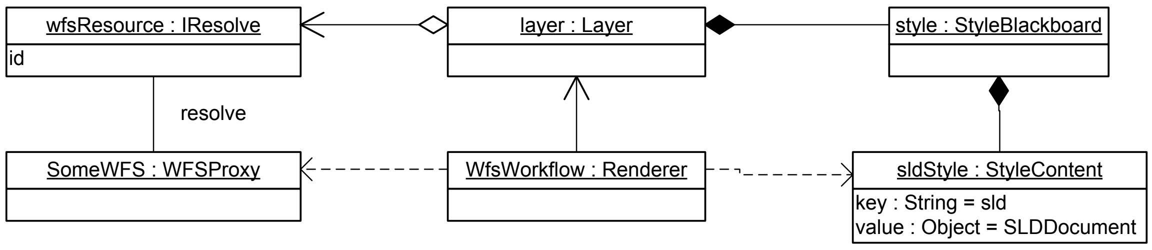


Figure 8 Adaptive Workflow Example WFS

In this initial example a default SLD style has been generated and placed on the blackboard, and a voting process has determined that a WfsWorkflow represents an appropriate method of visualizing the current layer. The WfsWorkflow resolves the wfsResource and makes a GetFeature request. As the feature information is retrieved the sldStyle is used to render the contents to the screen.

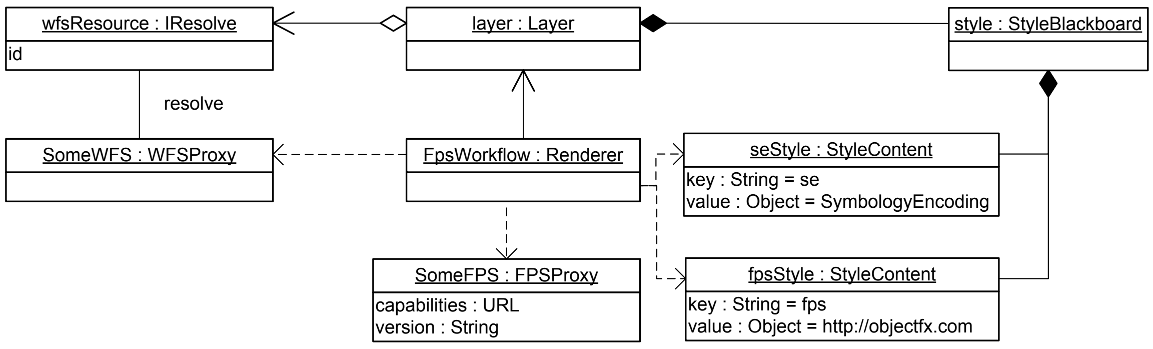


Figure 9 Adaptive Workflow Example FPS

After a catalog search the blackboard has been updated with a Symbology Encoding document (presumably matched to the FeatureType beind displayed) and a fpsStyle indicating a capable FPS. The voting process engages a FpsWorkflow which assembled this information into a GetPortrayal request for display.

The use of a blackboard for inter module communication in this manner stands in contrast to layer based architectures traditionally seen in network applications.

7.2.4 Supporting Multiple Versions

There are currently deployed a wide range of WFS and WMS services. An integrated client needs to be capable of supporting a version range with respect to each specification and should be designed to additional versions as they become available.

Many popular OWS-Service implementations vary slightly for the standardization, or have different interpretations of such things as scale denominator and SRS notation. At a pragmatic level interaction with these services is required, warts and all.

A) Conditional Logic

Given the variability between the specifications (not to mention the implementations) one should not consider making conditional code to handle the differences.

B) Version Specific Proxy

The first approach is very straight forward; make direct allowance for different versions of a service. When asked to connect, construct the most appropriate proxy to match the service being communicated.

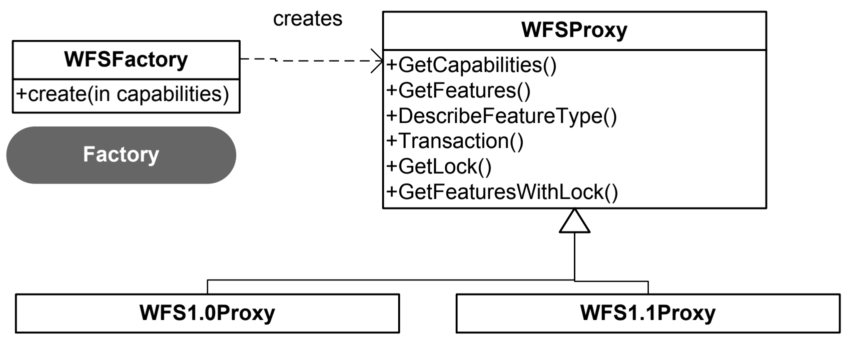


Figure 10 Negotiation using Factory

In the preceding diagram the flavors of WFS encountered during the OWS-3 project are illustrated. Version negotiation follows the guidelines described by the WFS specification.

The drawback to this approach is the very public nature in which additional versions (and quite frankly hacks) are made available. You do have the chance of inadvertently introducing dependence on a specific version into the rest of your code base.

C) Version Specific Stratagy

Use of a Stratagy object may also be considered to capture the version dependent functionality

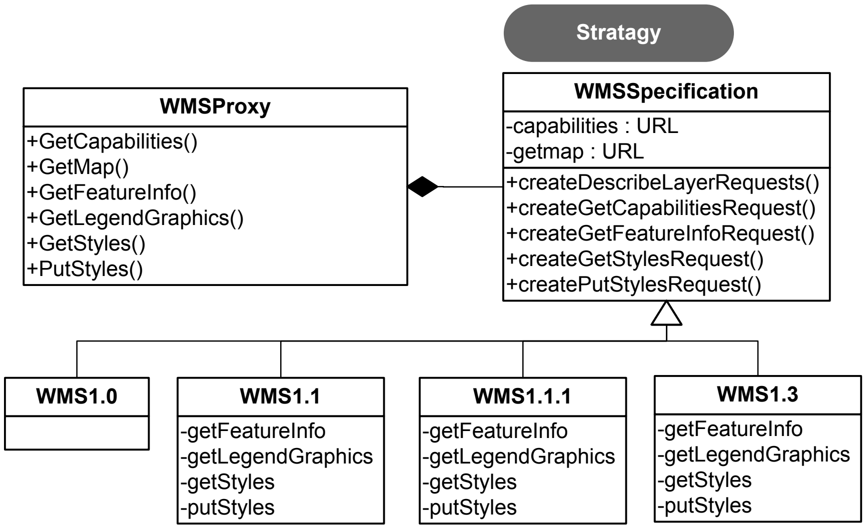


Figure 11 Version Negotiation using Strategy

In the above diagram the different WMS versions are accounted for by use of WMS Specification strategy object.

7.3 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 extensions are accommodated.

For many geospatial technology (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 recently developed systems are being 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 Geospatial Technology Platform and Components

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 bears 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 section, we discuss user interface.

8.1 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.

8.1.1 Search and Discovery System

The ability to search for data is fundamental to Service Discovery and Binding functionality. User interface of some sort 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 CS-W, 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.

8.1.2 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 services at this time are the Coverage Portrayal Service, which presents a WMS interface to the web, and the Data Aggregation Service, which presents remodeled WFS endpoints as an aggregated WFS.

8.1.3 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.

8.1.4 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.

8.1.5 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 may or may no be displayable upon receipt, depending upon format and the client’s capabilities.

8.1.6 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.

8.1.7 Geospatial Analysis Logic

A client that supports several OGC interfaces may be much more than a simple mapping tool. Several clients 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.

8.1.8 Modeling Tools or Interfaces

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

8.1.9 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.

9 User Interface Considerations

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 (beveled "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.

9.1.1 Interaction with Open Web Services

There are several guidelines to be aware of interacting with open web services:

- Allow the user to feel in control, techniques that allow the user interface to respond immediately to user interaction, such as direct manipulation, are encouraged.

- Gracefully handle latency, web services take some time to respond the user interface should not (even if the response to display a progress bar). A more extreme example would be streaming techniques that favor latency over throughput.

If possible endeavor to borrow from user interface concepts the user is already familiar with. As an example the web browser offers consistent model for web services where stop and refresh buttons are used to control the process.

9.1.2 Symbology

The use of appropriate Symbology straddles the boundary between a user interface and a geospatial concern. The emphasis here is on the ability to support the same context displayed appropriate to the user.

The use of Symbology the user has been trained with is expected to increase comprehension and reduce training costs.

9.1.3 Integration

The degree in which these services can be smoothly integrated (without manual intervention) depends largely on the quality of metadata. We have been able to achieve seamless results through the appropriate use of a catalog service, properly registered Symbology information and known WFS feature types.

The challenge is to provide sensible default values in an uncontrolled environment, and enough information to allow users to adapt. There is still some work to be done with respect to the WFS and WMS specifications in this regard.

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. During the course of the previous OWS-2 initiative, several languages were reviewed to fill this need.

The OWS-2 initiative was unable to identify an emerging standard and explored XUL, UIML and XIML as possible candidates. This topic is in need of more exploration.

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.

- 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 minimum feature-set for each web mapping use case, as detailed in the Overview (Section 5) above.
    - 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? What are the trade-offs for different groups of geospatial systems users? The Refractions GeoDSS client used in OWS-3 switch between different "perspectives" to accommodate change in workflow.
- 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 OWS-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 the Integrated Clients developed during the OWS3 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
    - Video display

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 Web Map Service (WMS)

The Web Map Service specification defines an interface with which a client can request visual representations of georeferenced images from a server. This allows data providers to present their features graphically without exposing the feature data itself. Some Web Map Services provide a mechanism that allows clients to provide their own Styled Layer Descriptor, used to visualize the layers.

This section describes the interactions that the Integrated Client makes with Web Map Servers.

10.1 Image Retrieval using a Web Map Service

A) Direct GetMap request

Initial contact with a Web Map Service usually begins with a GetCapabilities request. The response from this request is the Capabilities document that describes various aspects of the server, including contact and administrative information, operation support, and data that is available to be visualized.

The Integrated Client analyzes the Capabilities document and perhaps presents the user with some of the available information, for example, the list of layers.

When the Integrated Client wishes to display visualized data from the Web Map Service, it makes a GetMap request, including parameters indicating the desired data to be visualized; the styles that should be used to visualize the layers; the size and format of the image to be generated; the bounding box of the viewing area; and the coordinate reference system that the data should be visualized in. The server generates the requested image and returns it, or responds with a Service Exception if there is a problem with the request.

B) Use of Custom Styles

Integrated client makes use of DescribeLayer, on inspection of the URL it is determined that a WFS has been referenced. A request is made to retrieve the schema, allowing the creation of a user interface for SLD construction.

The SLD document is used in the creation of subsequent GetMap requests.

10.2 Issues and Tradeoffs with Web Map Service

10.2.1 Unable to Provide Support for SLD-enabled WMSs

The Styled Layer Descriptor specification defines a SLD-enabled Web Map Service that allows clients to provide their own SLD documents that can be used by the server to visualize the data instead of the servers own configuration.

In order for a client to properly generate an appropriate SLD document, the client needs to be able to look at the schema of the feature types behind each layer.

The problem encountered in usage is that access to these schemas is not standardized or commonly publicized. The SLD specification defines the DescribeLayer request for Web Map Services, allowing more information about the layer to be provided. If the Web Map Service uses a Web Feature Service as its source of data, the URL to the WFS may be provided in the DescribeLayer response. This only works in the case that the WFS exists and the data provider is willing to make their data public.

10.2.2 Differences Between Specification Versions

10.2.2.1 EPSG:4326

Between versions 1.1.1 and 1.3.0 of the Web Map Service specification, the definition of EPSG:4326 changed. The order of the axes in the bounding box parameter was swapped. This change was not obvious and has led to different services implementing support for EPSG:4326 in both ways, with no definite way to determine which axis order they are using.

10.2.2.2 Removal of SLD Hooks

Previous to the WMS 1.3.0 specification, there were hooks provided in the Capabilities document that were used by SLD-enabled Web Map Services. These include the operations DescribeLayer, GetLegendGraphic, GetStyles and PutStyles, as well as the UserDefinedSymbolization element, which provided the Capabilities document with the means to indicate that the service is SLD-enabled.

These elements have been removed from the WMS 1.3.0 specification, but the SLD specification has not been updated accordingly, creating a specification that is less functional than the one before it. Services that want to be SLD-enabled and support WMS 1.3.0 have no standardized way of communicating their SLD status.

10.2.3 Inconsistent Use of Scale Hints

The Web Map Service specification provides an element in the Capabilities document that indicates the minimum and maximum scales at which a layer should be requested. This provides the Integrated Client with a mechanism to determine whether a request should be made for the layer in the current viewing area. The schema of this element changed in 1.3.0.

Experience has shown that servers often provide different scale indications in their Capabilities document than the filters that the servers actually use themselves. One server encountered even declared the scale hints twice, using both 1.1.1 and 1.3.0 elements, but each with different values!

10.2.4 Poor Quality of Service

Experience has shown that the amount of layers in, mostly in Cubewerx WMS instances, is astounding. The Capabilities document takes a long time to download and parse, and the user is often overwhelmed by the selection of layers they can choose from.

In Cubewerx instances, because these layers are often cascaded, GetMap requests can often be very slow, or can even fail entirely if the original host server is down. While these instances are to be considered stateless (and a subsequent Capabillities request will show these layers removed) it is often too late from the perspective of an end-user.

There is no way to determine the quality of service behind a cascaded layer, and this can provide problems for users of Integrated Clients.

10.3 Implementation Comments

10.3.1 Intergraph

Since the Intergraph client is built on the commercially available GIS platform known as GeoMedia Professional, it is able to utilize existing interoperability components that allow for direct access to a WMS. The architecture for GeoMedia allows a user to connect to the native WMS data through a dataserver component. This component handles all initiation of requests and interpretation of responses from the native data format.

The user supplies the WMS endpoint to the connection service and the service makes a GetCapabilities request as the initial communication to the service. The list of available layers are then presented to the user. If the user wishes to view the data, for example, he or she will select the appropriate layers to be displayed and the appropriate GetMap requests are made and the returned image is displayed in a MapView window.

10.3.2 Refractions Research

Use of WMS is straight forward in the Refractions GeoDSS client (REFRACTIONS GEODSS CLIENT); a WMS capabilities URL may be dragged directly onto a map or supplied using a wizard. A series of WMS layers may be selected. Individual WMS layers also appear as the result of a search operation and may be dragged into the legend view directly.

Web Map Service support in Refractions GeoDSS client was implemented in the GeoTools project, leveraging other code available to provide support for coordinate reference systems and XML parsing. The rendering system detects when WMS Layers from the same server are next to each other and will collapse these down to a single request.

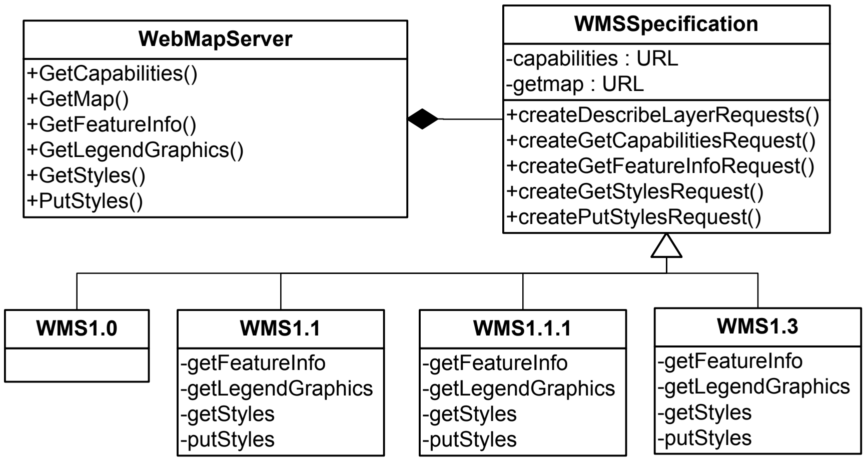


Figure 12 GeoTools Web Map Server Specification Support

In order to deal with the plentiful but subtle differences between the WMS specifications and improper server instances, we implemented a strategy object system that adapts itself to the current version of the server it is communicating with. The client performs version negotiation with the servers until it detects a version it can communicate with, or breaks off contact.

A separate technology was used to parse the XML responses from WMS which provides flexible support for the multiple versions of Capabilities document. It also tries to be as lenient as possible when it encounters nonconforming servers.

While the code in GeoTools provides support for all WMS versions, Refractions GeoDSS client does not use WMS 1.3.0, due to problems with EPSG:4326, noted in section 10.2.2.1.

10.4 Recommendations

Web Map Server is simply the widest deployed OWS service with an astounding variety of interesting information. Support for EPSG:4326 on WMS 1.3.0 is troublesome, but most servers that support version WMS 1.3.0 also support WMS 1.1.1. Providing support for SLD-enabled WMSs is exceptionally difficult to do well. Matters of performance may be addressed through use of QOS metrics in a catalog, but for now the timing of a capabilities response (or a sample image) is the safest bet.

Recommendations can be made for future revisions to the WMS specifications:

- Changes allowing for client generated styles: an explicit “SchemaURL” element in the capabilities document; description of the extent (or histogram) of individual schema attributes; definition of WMS 1.3.0 GetMap profile supporting Symbology Encoding documents

- Adopt a consistent interpretation meaning of SRS as outlined in section 11.3.1.

11 Web Feature Service (WFS)

Access to Feature information is a key ability of an integrated client; such information is available through WFS servers and is available in a variety of formats. This section documents the process of using a Web Feature Service directly from an Integrated Client.

Although the use of a Web Feature Service generally takes place in the context of a publish/find/bind scenario these examples will be limited to direct client/server communication after binding has already occurred.

11.1 Feature Retrieval using a Web Feature Service

The ability to directly communicate with a Web Feature Service provides an integrated client with a flexible mechanism for collaborative work

A) Discovery and Access to Feature Information

In the following example, an integrated client is used to connect to and retrieve feature information published in a Web Feature Service.

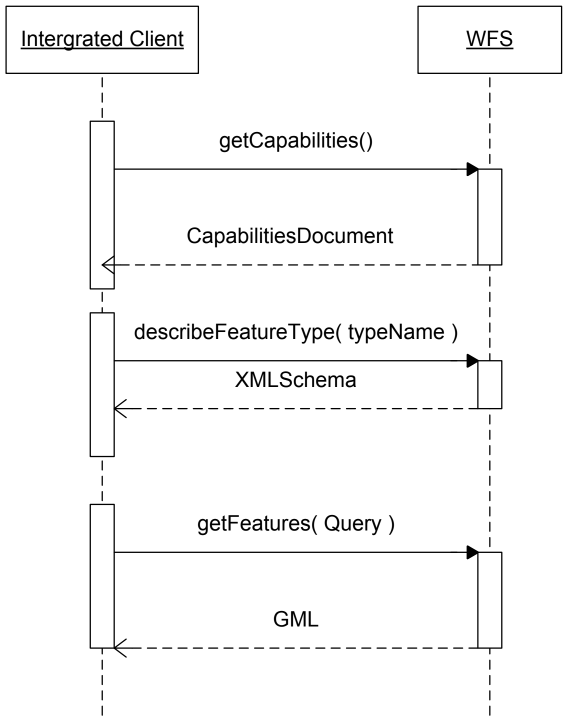


Figure 13 Feature Retrieval

In this example an integrated client is provided with a URL of a Web Feature Service. The Capabilities document is retrieved via a GetCapabilities request. The Capabilities document provides information detailing the available information by **TypeName**, and additional information is retrieved in the form of an XMLSchema document by way of a DescribeFeatureType request. Using the description of this FeatureType the client can now construct a WFS GetFeatures request and retrieve information in the form of GML.

Feature content may be accessed in this manner for both portrayal and analytical work.

B) Feature Visualization

There are three options available when considering how to visualize Features retrieved from a Web Feature Server.

1. The Style Layer Descriptor specification provides a formal definition of how Feature content may be portrayed as part of the definition of a **FeatureTypeRule** element.

   At the time of this writing SLD 1.0 was published, and several changes were planned for SLD 1.1. These changes do not impact the use of SLD described here in.

2. The GML specification itself provides visualization "defaults" that may be considered. Our impression is that these were intended for those making transformations of GML to other XML technologies such as SVG or X3D.

3. The useful subset of Style Layer Descriptor indicated above, FeatureTypeRule, has been isolated for use as a Symbology Encoding Document.

   It is hoped that the SLD RWG will continue this work as a way of moving the SLD specification forward.

There are differences between the defaults indicated between the GML and SLD specifications. At this time no known harmonization effort is underway. Due to the additional expressiveness present in the SLD approach we would like to recommend its use.

For a detailed discussion of portrayal please review the section on Feature Portrayal Service included in this document.

C) Direct Feature Access

In the following, an integrated client is used to directly retrieve Feature information without preamble. As part of validating the resulting GML Document the describeFeatureType operation will be called to generate a schema.

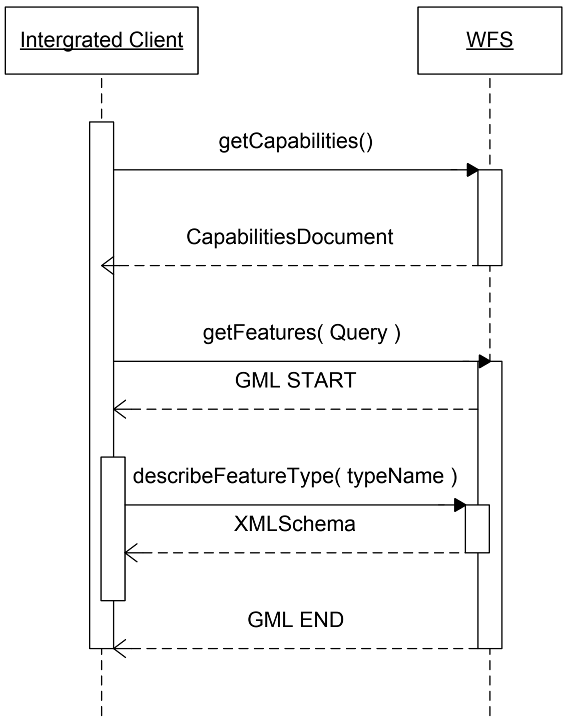


Figure 14 Direct Feature Access

It is important to be aware of the schema as it plays its established role in validation of an XML document. Care should be taken to ensure that the DescribeFeatureType operation is not called twice for both validation of the XML document and description of the presented Features.

Direct access as a raw XML document is often associated with SVG based rendering systems, in which only XML transformations are applied to produce visualization.

Care should be taken to avoid this technique when extremely large schema documents are being used as the original XML document stream may timeout. During the course of OWS-3, schema files 12MB took 30 seconds to parse.

11.2 Feature Production and Modification

The ability to interact with and produce spatial information is an important role of an integrated client. The Open Web Services allow for this use through the use of the Web Feature Server Transaction operation.

Conformant Web Feature Servers are allowed to forgo implementation of the Transaction operation, and are still referred to as valid WFS servers. The abbreviation WFS-T is used to indicate fully transaction web feature servers.

Since this is the only opportunity for an Integrated Client to produce content, WFS-T support is viewed as essential to success.

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

A Transactional Web Feature Server (WFS-T) allows users to retrieve and modify feature data. After acquiring data the user can construct a transaction request consisting of a series of delete, update, and additions to be performed by the web feature server.

Two models of "success" are supported by WFS 1.0, partial success in which the modifications that could not be performed are indicated.

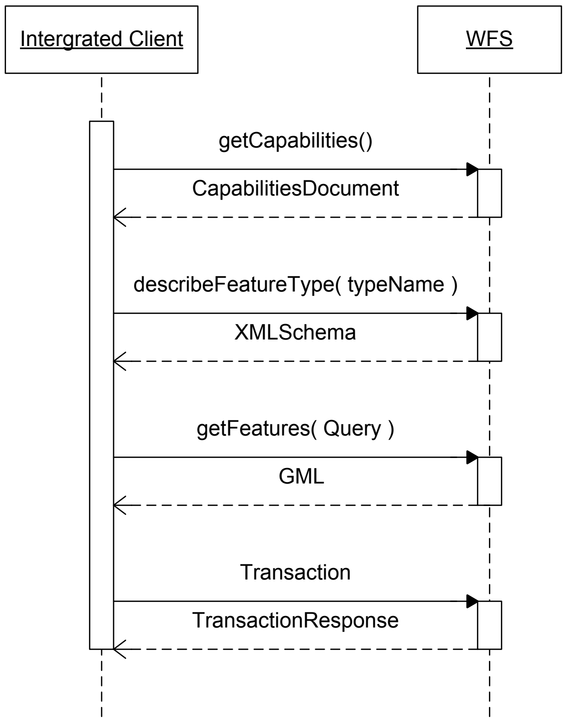


Figure 15 Transaction Support

Although the use of a WFS-T was not covered during our demo scenario, this document reflects the experience of those implementing support outside of the bounds of the OWS-3 experiment.

B) Web Feature Server Locking Support

In addition to direct access through the transaction operation the WFS specification outlines a simple locking system. These Locking facilities represent an interesting compromise going beyond short term locking, while falling short of a complete strong-transaction support system.

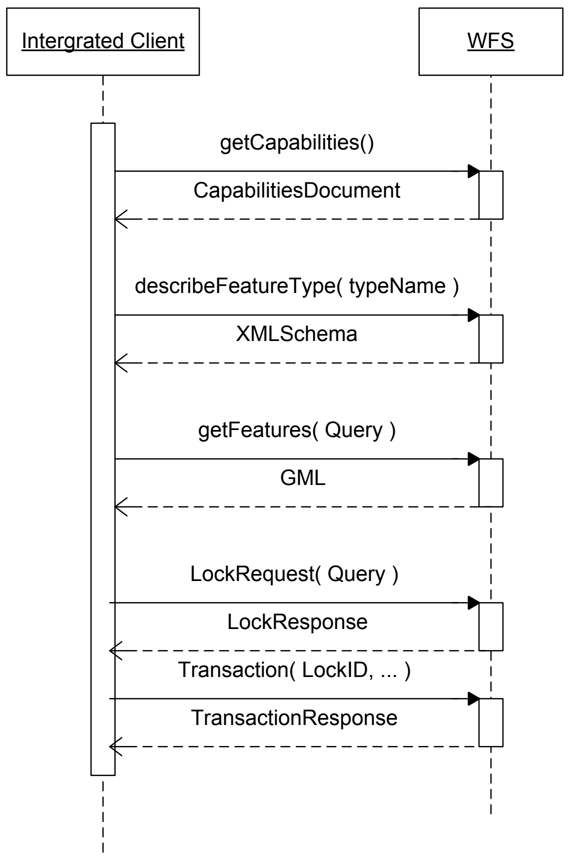


Figure 16 Locking

The locking model is based the use of tokens, which are obtained for a requested duration. These lock tokens are consumed during the course of a transaction. Depending on the method of use the Lock may be relinquished on just the features modified by the Transaction request.

While not providing facilities such as checkpoints, or conflict resolution this does represent a workable middle ground.

11.3 Issues and Tradeoffs with Web Feature Service Communication

There are several pitfalls in the current Web Feature Service specifications; areas where industry practice diverges from what one may expect from reading the specification.

11.3.1 Spatial Reference System

The Capabilities document produced by a Web Feature Service provides an SRS element associated with each FeatureType. The interpretation of this SRS tag is divided depending on the format returned:

- EPSG:4326
  The interpretation of this SRS is ambiguous in practice. The pragmatic solution is to interpret the ordinates in eastward/northward order, ignoring the specified order in the EPSG database.

  This follows industry practice, and the use of EPSG codes by the early WMS services. This is only a problem for a small percentage of the allowable codes. The approach is in conflict with the latest Web Map Server specification, and represents a difference between practice and correctness.

- URI
  Understood to indicate CRS information directly as specified by the European Petroleum Standards Group.

- WKT
  Provides the WKT representation of the information retrieved.

- AUTO:42000,0,0,0
  Reference to the AUTO projections maintained by the Web Map Server specifications (both AUTO, and AUTO2 are in use). The numbers used are “units”, “lat” and “long”. Please consult document the WMS 1.1.1 specification for more information.

The best advice we were offered over the course of our integrated client work was to treat “EPSG:1234” in the historical manner. Use the new URI format wherever possible (as its meaning is clear and consistent).

11.3.2 WFS 1.0 and GML3

One of the difficulties in communicating with the WFS services available over the course of this documents construction is the lack of full WFS 1.1 services. As such GML3 data (actually a Simple Feature profile of GML3) was made available via WFS 1.0 services.

11.3.3 Construction of Default Styles

The GML3 specification provides guidance for rendering defaults that conflict with those provided by the Style Layer Descriptor specification.

Further more the detail of information required to construct a good default style is not available by way of DescribeFeatureType. Such information would need to be combined with aggregate information about the data available for each attribute type.

As an example the construction of a simple sequential theme requires knowledge of the min and max values.

11.3.4 Invalid Schemas (and Conformance)

As noted in the interoptability experiments, the XML Schema information provided by DescribeFeatureType is occasionally inaccurate. We can only assume that many of these Web Feature Servers are used for specific applications where the correct processing is known to that parties involved.

In general we found conformance to WFS 1.0 specification to be very high.

11.3.5 Use of WFS-T

Several issues arise with respect to the use of WFS-T:

- The WFS 1.0 specification supports the idea PARTIAL_SUCCESS in response to a Transaction request. This model is not deterministic and interaction and its use is not advised.

- A WFS-T may perform additional integrity checks and produce a service exception based on application criteria (rather then simply specification conformance). Correct use of the service may still produce a service exception.

- When performing a Transaction operation on more then one server both transactions should succeed or neither.

This last point cannot be realized using the existing specification. A vendor extension should be feasible implementing a two-phased commit process. This would require an additional "Commit" operation to be called either by the CSW in the role of a Mediator, or by another WFS-T as part of fulfilling its Transaction request.

11.3.6 Use of Locking

A couple of practical difficulties arise out of the use of WFS Locking:

- Presentation to end-users, who expect to be able to work on a locked feature until their work is completed.

- The two operations allowing for Locks, GetLock and GetFeaturesWithLock, have little practical difference between them. We can only assume this was introduced as an optimization.

Given the impedance between user expectations and the WFS locking model two options may be considered. Reveal the timed nature in which modifications may take place or delay obtaining a lock until the user has assembled a series of modifications to apply.

The integrated client may remember which Features have been modified and limit the use of locking to an additional level of protection during the commit process. A lock, of limited duration, can be acquired for the modified Features – and then immediately used in a Transaction operation. This option forgoes the advantage of blocking other applications from making a modification while the end user is putting together a list of modifications.

11.3.7 Lack of Notification

There exists no establish mechanism for noticing when changes have been made to a Web Feature Server. The server provider by Galdos allows for the ability to cascade Transaction operations between Web Features Servers, and a client may include itself in this cascade. The Web Feature Server specification does indicate changes in the capabilities document via an update sequence timestamp. Periodic polling of this value may be used as a way to notice when any local caches of Feature data should be refreshed.

Adoption of these ideas has not been perused, and a “refresh” button is the only established solution.

11.4 Implementation Comments

11.4.1 Intergraph

Since the Intergraph client is built on the commercially available GIS platform known as GeoMedia Professional, it is able to utilize existing interoperability components that allow for direct access to a WFS. The architecture for GeoMedia allows a user to connect to the native WFS data through a dataserver component. This component handles all initiation of requests and interpretation of responses from the native data format.

The user simply supplies the WFS endpoint to the connection service and the service makes a GetCapabilities request as the initial communication to the service. Then depending on the user’s application and needs, a DescribeFeatureType request can be made to present the available data possibilities back to the user. If the user simply wishes to view the data, for example, he or she will select the feature types to be displayed and the appropriate GetFeature requests are made and displayed in a MapView window with default symbology. The display of the feature data can be easily modified to a more appropriate user-defined symbology.

The commercial GIS platform also supports basic transactions (WFS/T), insert, update, and delete. The user connects to a WFS/T site. The user then runs Begin WFS Update which brings the WFS/T data into a R/W access connection. The user can then update/insert/deleted in the Access warehouse. The last step is to run Complete WFS Update. This command will then build the correct requests and send them to the WFS/T site. We do not implement the optional Lock Feature request or GetFeatureWithLock (which is also optional) at this time.

11.4.2 Refractions Research

Communication with a Web Feature Server is straightforward; it is processing GML that presents most of the challenge. We found it very helpful to fetch the XML Schema from DescribeFeatureType prior to processing a GetFeatures request. In many situations this represented the only option to prevent timeouts from occurring (only the XML Schema files used in this demo took over a minute to parse).

We have explored four approaches to processing GML for display:

- XML Technologies: SAX, DOM or a pull parser can provide access to XML content at a very low level of abstraction. XML Schema information can be leveraged to determine which elements to pay attention to, pull parsers represent the most direct way to take advantage of this information.

- High level XML Technologies: XPath and XMLSLT in order to transform GML into SVG for display, the information in XML Schema can be used to construct the required XPath expressions

- Java Binding Technologies: Many choices exist for directly converting an XML Schema into java beans with an associated binding service, at the time of writing JAXB served as a good example. This approach is poorly suited to the needs of a Integrated Client, there is some delay involved between generating source code, compiling and being able to handle the request.

- Schema Assisted Parsing: Using knowledge of the schema you can also set up handlers for the base GML types, the Application Schema returned by DescribeFeatureType is constructed by extending the base GML types and by tracking the extensions one can locate a suitable handler.

It is assumed for the above discussion that your display is based on a portrayal model similar to SLD.

Communication with a WFS-T presents a different set of challenges; we ended up maintaining a list of local unapplied modifications. This list of modifications can be used to post-process the content coming from subsequent WFS requests. When the user chooses to commit a WFS Transaction request can be constructed. It should be noted that communication with a WFS-T offering Partial Success is non deterministic.

In order to support different configurations (WFS 1.0 & GML2 vs WFS 1.0 and GML3) we made distinct plug-in for each. The correct plug-in was chosen based on initial version negotiation. This represents an expansion of the negotiation defined by the OWS Commons to include examination of the Capabilities Document.

11.5 Recommendations

The direct use of Web Feature Server in an integrated client is possible, and recommended. None of the issues mentioned above our serious enough to prevent deployment.

The use of PARTIAL_SUCCESS in WFS 1.0 is not recommended, and we are pleased to see it removed from WFS 1.1.

12 Web Coverage Service (WCS)

A Web Coverage Service is used to discover and retrieve coverage data in the form of georeferenced image files. This section documents the process of using a Web Coverage Service directly from an Integrated Client.

12.1 Direct Access to a Web Coverage Service

The ability to directly communicate with a Web Coverage Service provides an integrated client with a means of working with "coverages". A coverage is digital geospatial information representing space-varying phenomena.

A) Web Coverage Retrieval

In the following example, an integrated client is used to discover and access coverage information.

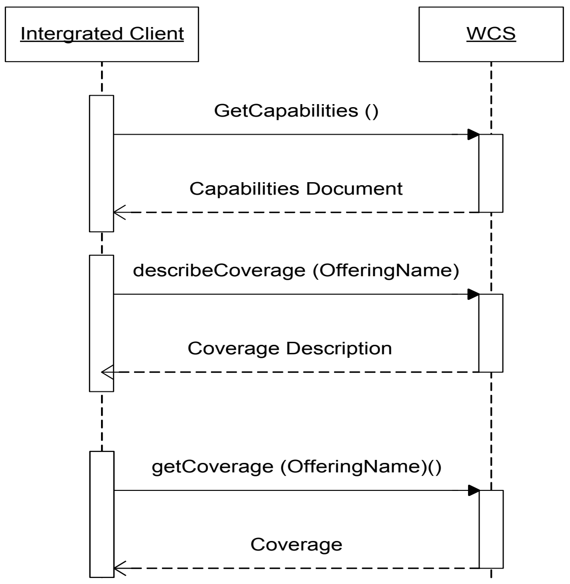


Figure 17 Coverage Retrieval

In this example an integrated client is provided with a URL of a Web Coverage Service. The Capabilities document is retrieved via a GetCapabilities request. The Capabilities document provides information detailing the available information by "coverageOfferingName", and additional information is retrieved in the form of an XML document by way of a DescribeCoverage request. Using the description of this coverage offering, the client can now construct a WCS GetCoverage request and retrieve a result in a form of one of the formats supported by the WCS.

12.2 Issues and Tradeoffs with Web Feature Service Communication

Although used and implemented within the OWS-3 testbed, Web Coverage Services were not a focal point of the demonstrations for the integrated clients. Part of the reason for this might be due to the spotlight given to WCS during the OWS-2. It is believed that many of the issues and concerns dealing with a coverage service were discussed and dealt with during that interoperability testbed.

12.3 Implementation Comments

12.3.1 Intergraph Corporation

The user supplies the WCS endpoint to the connection service and the service makes a GetCapabilities request as the initial communication to the service. The list of available coverage offering are then presented to the user along with a number of metadata elements describing each coverage offering. The user selects the appropriate coverage and enters information about how to retrieve the image including bounding box, sampling, and return format information. Then a GetCoverage request is issued and the image is returned to the client.

Once the image is returned to the client, it is written to local memory and stored in a warehouse connected to the commercial GeoMedia platform. At this point the recently acquired image is added to the MapView legend using standard commercial rendering capabilities based on the return format of the image.

12.4 Recommendations

Since the WCS was not a focal point in this testbed, it was not addressed to the detail that would reveal items to be discussed as future recommendations. However, due to the efforts undertaken as part of the OWS-2 interoperability experiments, many recommendations have / are already being addressed.

13 Catalog Service for the Web (CS-W)

It is believed that the ability to discover geospatial resources is an important role for any integrated client. The OGC Catalog Service for the Web is an abstract specification. That abstract specification serves as the basis for specifications that can be implemented called profiles. OGC has currently defined three profiles of the catalog specification:

- CSW Record profile
- ebRIM profile, and
- ISO profile.

At its core, an ebRIM catalog stores two kinds of objects: extrinsic objects and service objects. There are two classes of extrinsic objects: those that exist by virtue of a service, and those that exist independently of a service. Examples of the latter include data sets, schemas and symbology encoding documents. Examples of extrinsic objects that exist by virtue of a service include layers (WMS), feature types (WFS) and observation offerings (SOS).

A service and its related objects typically make their way into a catalog via a "harvesting" operation. The harvesting operation generally ingests a capabilities response from a service and populates the catalog with a service object and a set of extrinsic objects that represent the collection of objects on which that service "operates".

13.1 Direct Access to a Catalog Service for the Web (CS-W)

Direct communication with a Catalog Service for the Web allows a user to query and retrieve information about various web services based on criteria such as bounding box and keywords. This allows the user to review the metadata about a given service prior to executing a request for information.

In the following example, an integrated client is used to connect to and retrieve metadata published through a Catalog Service for the Web. It is recognized that this example is only a small scope of the catalog retrieval functions and is meant only as an example of an implementation within the framework of the OWS-3 testbed.

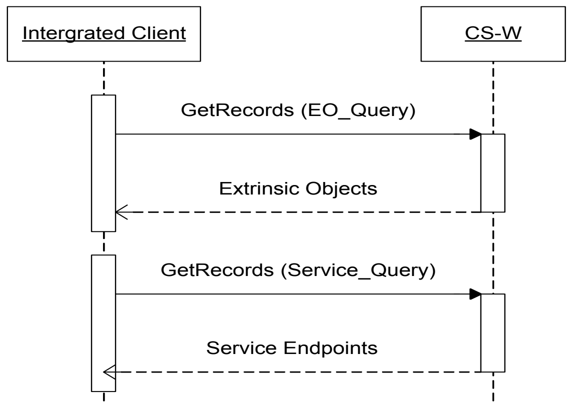


Figure 18 Catalog Retrieval

In this example an integrated client is provided with the URLs of the Catalog Service endpoints. Based on user input, a query is constructed with which a GetRecords request is made to find Extrinsic Objects based on the object type (FeatureType, WMS_Layer, ObservationOffering). The response will include the objects identified along with metadata about the object including an id for each object. Another GetRecords request can then be constructed to build a Service request to identify the service endpoints for each object. With this information the user has the information needed to bind to the appropriate data.

13.2 Issues and Tradeoffs with Catalog Service for the Web Communication

This section presents a summary of interoperability issues discovered while working with the CubeWerx and IONIC ebRIM 2.5 catalogs.

13.2.1 Locating Extrinsic Objects by Object Type

The ebRIM model provides an attribute named "objectType" on the "ExtrinsicObject" element. ebRIM 2.5 restricts the value of this attribute to a URN of type UUID. This UUID is meaningful only to a particular catalog instance and its meaning must be derived by lookup from the ebRIM classification structure. This obfuscation makes it awkward, we believe, to simply find extrinsic objects by type.

The participants in the OWS-3 Common Architecture thread agreed that for the purposes of the initiative we would deviate from the ebRIM 2.5 profile and use simple English names for the value of "objectType". For the purpose of this testbed, the following values were recognized:

Service	Object Type
WMS	WMS_Layer
WFS	FeatureType
SOS	ObservationOffering
GVS	VideoFeed

A recent version of the ebRIM profile document provides more flexibility for "objectType", in that it may take on a URN of the form:

urn:ogc:specification:csw-ebrim:objectType:Dataset:*

Here are representative URNs proposed by CubeWerx that may be considered in future ebRIM catalog implementations:

Service	Object Type
WMS	urn:ogc:specification:csw-ebrim:objectType:Dataset:WMS_Layer
WFS	urn:ogc:specification:csw-ebrim:objectType:Dataset:WFS_Feature

13.2.2 Association Types

A key concept of the ebRIM catalog model is the "Association". Clients make use of an association to relate an extrinsic object to a service. To promote catalog interoperability, we feel it is important to standardize a core set of association types. Here are interoperability issues we encountered in working with associations:

- "operatesOn" vs. "OperatesOn": We noted variations in the capitalization of the association name.

- "OperatesOn" vs. "HasForContents": Does an SOS "OperatesOn" an "ObservationOffering" or does it "HasForContents" same?

13.2.3 Case of Keywords

We believe it should be possible for a catalog to ignore case in user-specified keywords. More generally, this means allowing for the possibility of case-insensitive treatment of the value of "Literal" elements in the OGC Filter expression.

13.2.4 Predicate on Spatially Restricted Searches

When specifying a geometry to bound a search, it is important for Catalog clients and services to agree on the meaning of spatial predicates. For instance, for the use case of finding extrinsic objects that lie within a boundary of interest, is the appropriate predicate "Contains" or "ContainedBy"? After some discussion in the Common Architecture thread on this specific issue, we agreed on the latter. Lesson: common understanding of the meaning of these spatial predicates is essential to achieving interoperability.

13.2.5 Service Type

As argued previously, it is convenient to be able to locate extrinsic objects by type via the "objectType" attribute. One could also argue that it should be relatively easy to locate service objects by type. The ebRIM model does not provide an attribute that might be called "serviceType" on service objects. However, it does provide for the use of profile-defined "slots". Although the Common Architecture thread participants agreed on the usefulness of including of a slot to define the service type, this was not actually implemented and tested for OWS-3.

13.2.6 AccessURI Attribute in Service Binding Elements

Intergraph noted that the "accessURI" attribute returned on a "ServiceBinding" element sometimes included a query string. For example:

```
<ebxml:ServiceBinding
  id="urn:uuid:836c333e-189d-459d-8571-92c883d1dde0"
  accessURI="http://vast.uah.edu:8080/ows/dopplerSos?service=
     SOS&amp;version=0.0.31&amp;request=GetCapabilities">
```

Our understanding is that the value of "accessURI" should be set to what is generally referred to a "prefix", that is that portion of a URI to which standard service parameters are appended. The prefix may include a query string consisting of one or more proprietary parameters if required by a vendor.

13.2.7 Exceptions

Intergraph noted differences in the schema for the exceptions returned by the catalogs. Although this was discussed briefly during the initiative, it seems there may be some ambiguity as to the proper schema and reported version of an exception response.

ServiceExceptionReport Example

```
<?xml version="1.0" encoding="utf-8"?>
<ServiceExceptionReport
    version="1.2.0" xmlns="http://www.opengis.net/ogc"
    xmlns:xlink="http://www.w3.org/1999/xlink">
  <ServiceException code="InvalidParameter">
    Exception text...
  </ServiceException>
</ServiceExceptionReport>
```

ExceptionReport Example

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<ows:ExceptionReport
    version="0.2.3"
    xmlns:ows="http://www.opengeospatial.net/ows">
  <ows:Exception exceptionCode="CWWRS-00002">
    <ows:ExceptionText>Exception text...</ows:ExceptionText>
  </ows:Exception>
</ows:ExceptionReport>
```

13.3 Implementation Comments

13.3.1 Intergraph Corporation

Intergraph believes that the ability to discover geospatial resources is an important role of an integrated client. Indeed, for the integrated client that Intergraph built and demonstrated for OWS-3, resource discovery via the OGC catalog plays a central role.

Intergraph's work in the OWS-3 initiative focused on the ebRIM 2.5 profile. IONIC and CubeWerx provided ebRIM 2.5 catalog service implementations to the OWS-3 participants.

In its OWS-3 client implementation, Intergraph focused on the discovery of service-related objects, as service binding was our ultimate goal. We relied on vendor-provided web pages for fulfilling harvesting needs.

The catalog search facility that Intergraph built into its client has its foundation in two queries:

4. Find all extrinsic objects that meet the specified criteria.
5. Find the service which "operates on" a selected extrinsic object.

To find extrinsic objects of interest, Intergraph's client allows the user to specify the following criteria:

- extrinsic object type
- one or more keywords (including wildcards)
- a geo-spatial bounding box

The OGC filter expression in this ebRIM query represents the encoding of these criteria. Also note the "maxRecords" attribute on the root "GetRecords" element:

```
<?xml version="1.0" encoding="UTF-8"?>
<GetRecords maxRecords="3" outputFormat="application/xml;
      charset=UTF-8" outputSchema="EBRIM" version="2.0.0"
```

```
       xmlns="http://www.opengis.net/cat/csw">
       <Query typeNames="ExtrinsicObject">
         <ElementName>/ExtrinsicObject</ElementName>
         <Constraint version="1.0.0">
           <ogc:Filter xmlns:ebxml="urn:oasis:names:tc:ebxml-
      regrep:rim:xsd:2.5"
             xmlns:gml="http://www.opengis.net/gml"
             xmlns:ogc="http://www.opengis.net/ogc">
             <ogc:And>
               <ogc:PropertyIsEqualTo>

      <ogc:PropertyName>/ExtrinsicObject/@objectType</ogc:Propert
      yName>
                 <ogc:Literal>FeatureType</ogc:Literal>
               </ogc:PropertyIsEqualTo>
               <ogc:Or>
                 <ogc:PropertyIsLike escape="\" singleChar="_"
      wildCard="%">
                   <ogc:PropertyName>
                     /ExtrinsicObject/Name/LocalizedString/@value
                   </ogc:PropertyName>
                   <ogc:Literal>%school%</ogc:Literal>
                 </ogc:PropertyIsLike>
                 <ogc:PropertyIsLike escape="\" singleChar="_"
      wildCard="%">
                   <ogc:PropertyName>

      /ExtrinsicObject/Description/LocalizedString/@value
                   </ogc:PropertyName>
                   <ogc:Literal>%school%</ogc:Literal>
                 </ogc:PropertyIsLike>
               </ogc:Or>
               <ogc:Contains>
                 <ogc:PropertyName>

      /ExtrinsicObject/Slot[@name='FootPrint']/ValueList/Value[1]
                 </ogc:PropertyName>
                 <gml:Box srsName="EPSG:4326">
                   <gml:coordinates>-80,30 -
      70,40</gml:coordinates>
                 </gml:Box>
               </ogc:Contains>
             </ogc:And>
           </ogc:Filter>
         </Constraint>
       </Query>
  </GetRecords>
```

The service binding task is fulfilled by this query: Given an extrinsic object of interest (e.g., a layer, feature type or observation offering), find the service which "operates on" it:

```
<?xml version="1.0" encoding="UTF-8"?>
<GetRecords maxRecords="1"
  outputFormat="application/xml; charset=UTF-8"
  outputSchema="EBRIM" version="2.0.0"
  xmlns="http://www.opengis.net/cat/csw">
  <Query typeNames="Service Association">
    <ElementName>/Service</ElementName>
    <Constraint version="1.0.0">
      <ogc:Filter
          xmlns:ebxml="urn:oasis:names:tc:ebxml-
 regrep:rim:xsd:2.5"
          xmlns:gml="http://www.opengis.net/gml"
          xmlns:ogc="http://www.opengis.net/ogc">
        <ogc:And>
          <ogc:PropertyIsEqualTo>

 <ogc:PropertyName>/Association/@associationType</ogc:Proper
 tyName>
            <ogc:Literal>OperatesOn</ogc:Literal>
          </ogc:PropertyIsEqualTo>
          <ogc:PropertyIsEqualTo>

 <ogc:PropertyName>/Association/@sourceObject</ogc:PropertyN
 ame>
            <ogc:PropertyName>/Service/@id</ogc:PropertyName>
          </ogc:PropertyIsEqualTo>
          <ogc:PropertyIsEqualTo>

 <ogc:PropertyName>/Association/@targetObject</ogc:PropertyN
 ame>
            <ogc:Literal>
              urn:uuid:58958c26-5156-46f7-b432-e9d17acf359f
            </ogc:Literal>
          </ogc:PropertyIsEqualTo>
        </ogc:And>
      </ogc:Filter>
    </Constraint>
  </Query>
</GetRecords>
```

Here, the extrinsic object of interest (the "target object") is specified by its UUID, which was determined by the previous query. An ebRIM "association" named "OperatesOn" specifies the relationship between the service object (the "source object") and the extrinsic object.

13.3.2 Refractions Research

The support for catalog services in the Refractions integrated client took two forms: an embedded window to the catalogs web interface and a predefined query to discover a specific document associated with a feature type.

13.3.2.1 Embedded Catalog Web Interface

The Refractions integrated client has the ability to display web sites through an OLE bridge to the systems native web browser. Once displayed, the user can navigate the interface provided, and perform any queries it supports. Once the user has discovered a service of interest, simply clicking a link to the capabilities document, or dragging it onto the map editor will import the service. There are four catalog services currently embedded in the Refractions client: CubeWerx Catalog, Earth Observations Service Support Environment Portal, Ionics RedSpider Catalog2, NASAs Earth-Sun System Gateway.

13.3.2.2 Discovery of a Document

A symbology encoding document was used by the Refractions client as described in section 14. These documents were stored in the CubeWerx Catalog, and dynamically discovered by the client. The following query was precanned, and the desired feature type name was substituted in when the request was made. This method of discovery allowed the documents to change and move throughout development, without requiring updates to the application.

```
<GetRecords
    xmlns="http://www.opengis.net/csw"
    xmlns:ogc="http://www.opengis.net/ogc"
    xmlns:ebrim="urn:oasis:names:tc:ebxml-regrep:rim:xsd:2.5"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation=
    "http://www.opengis.net/csw
    http://schemas.opengis.net/csw/2.0.0/CSW-discovery.xsd
    http://www.opengis.net/ogc
    http://schemas.opengis.net/filter/1.0.0/filter.xsd
    urn:oasis:names:tc:ebxml-regrep:rim:xsd:2.5
    http://www.pvretano.com/schemas/ebrim/2.5/rim.xsd"
    version="2.0.0"
    service="CSW"
    outputFormat="text/xml"
    outputSchema="urn:oasis:names:tc:ebxml-regrep:rim:xsd:2.5">
  <Query typeNames="ebrim:ExtrinsicObject=E1
      ebrim:Association=A
      ebrim:ExtrinsicObject=E2">
    <ElementName>/E2</ElementName>
    <Constraint>
      <ogc:Filter>
        <ogc:And>
          <ogc:PropertyIsEqualTo>
            <ogc:PropertyName>
              /E1/@objectType
            </ogc:PropertyName>
            <ogc:Literal>FeatureType</ogc:Literal>
          </ogc:PropertyIsEqualTo>
          <ogc:PropertyIsEqualTo>
            <ogc:PropertyName>
```

```
              /E1/Name/LocalizedString/@value
            </ogc:PropertyName>
            <ogc:Literal>%s</ogc:Literal>
          </ogc:PropertyIsEqualTo>
          <ogc:PropertyIsEqualTo>
            <ogc:PropertyName>
              /A/@sourceObject
            </ogc:PropertyName>
            <ogc:PropertyName>
              /E2/@id
            </ogc:PropertyName>
          </ogc:PropertyIsEqualTo>
          <ogc:PropertyIsEqualTo>
            <ogc:PropertyName>
              /A/@associationType
            </ogc:PropertyName>
            <ogc:Literal>Symbolizes</ogc:Literal>
          </ogc:PropertyIsEqualTo>
          <ogc:PropertyIsEqualTo>
            <ogc:PropertyName>
              /E1/@id
            </ogc:PropertyName>
            <ogc:PropertyName>
              /A/@targetObject
            </ogc:PropertyName>
          </ogc:PropertyIsEqualTo>
        </ogc:And>
      </ogc:Filter>
    </Constraint>
  </Query>
</GetRecords>
```

13.4 Recommendations

A facility to discover geospatial resources via a catalog is most certainly an essential component of an integrated client. However, the value of such a facility can be hindered when catalogs offer their content in dissimilar fashions. In order to promote the best prospects for interoperability, catalogs must adopt common data structures for the objects they store. For the ebRIM information model, that means using common templates for the ExtrinsicObject and Service objects.

In this initiative, we were pleased to be able to achieve a fair degree of interoperability between the IONIC and CubeWerx catalogs. Some of the problems that we encountered are outlined in the preceding clauses. To learn more about obstacles to achieving interoperability, it would have been useful to have had access to additional catalog services. Also, we would have found it informative for the catalog service implementers themselves to conduct and report on their own interoperability experiments.

In this initiative, our work centered on catalogs built on the ebRIM profile. We understand there is also considerable interest in the CSWRecord and ISO catalog profiles and recommend they be explored as part of an integrated client in a future effort.

When used for client development, the catalog should be invisible. The catalog is the source of the extra information that allows a client to operate without pestering the user for additional information.

When used explicitly by the user it is important to present the discovery workflow in an order that make sense from the standpoint of a
user.

14 Feature Portrayal Service (FPS)

The new Feature Portrayal Service specification describes a service that is used to remotely visualize features from a Web Feature Service. Clients indicate the WFS and feature types to be used, and provide styling information. This is useful when the data from the WFS cannot be visualized locally, possibly because the data is not familiar, or because visualization of the symbology is not known.

14.1 Portrayal (Image) Retrieval using a Feature Portrayal Service

A) Feature Portrayal Transparent Workflow

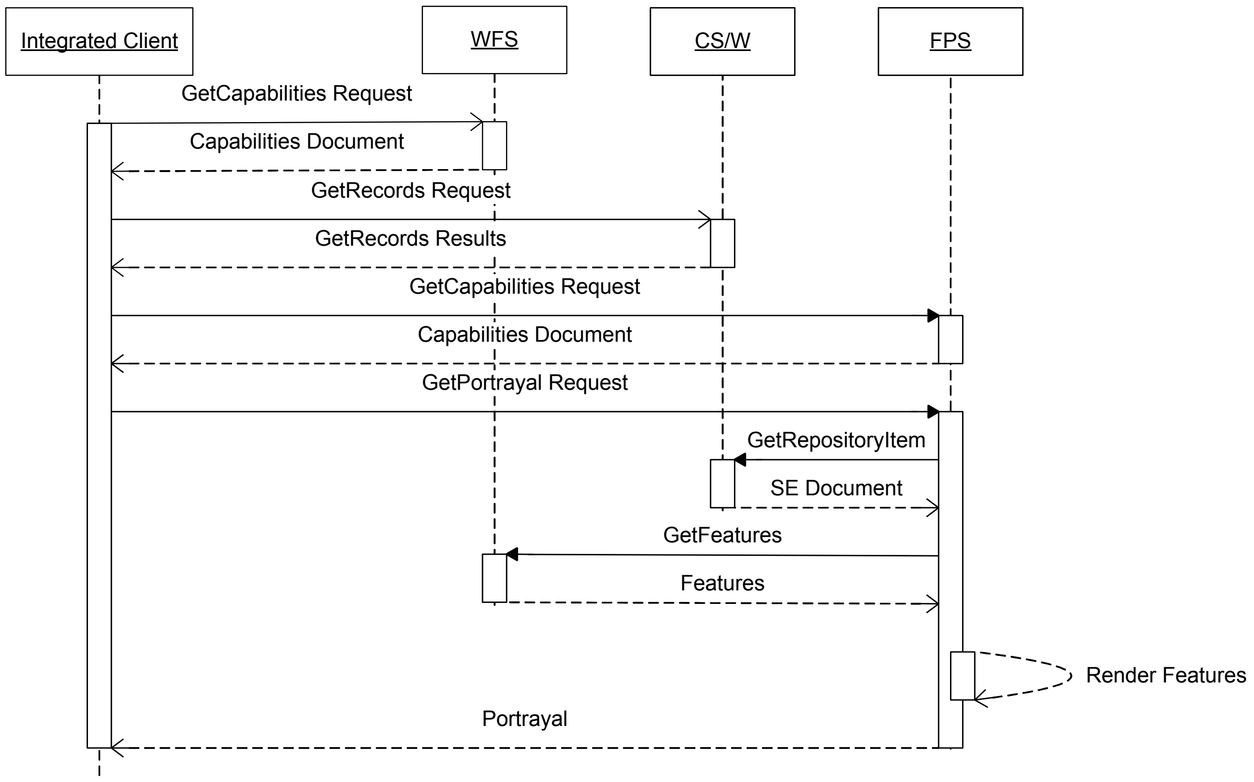


Figure 19 Image Retrieval Through an FPS

The above diagram illustrates the sequence of requests and responses that are used to retrieve a Portrayal from a Feature Portrayal Service.

The first transaction is a GetCapabilities request to the WFS server containing the features that are to be displayed. To use a FPS an integrated client must create (or locate) an appropriate Symbology Encoding document. In the above example a Catalog Service is queried to obtain a SE document. After retrieving the Capabilities document from the FPS, a GetPortrayal request can be constructed. This request consists of the URL of the

WFS, the URL of the SE document, and the typename of the features to be rendered. The request is then sent to the FPS to retrieve a Portrayal.

The Feature Portrayal Service retrieves the Symbology Encoding document, as well as the features from the WFS and renders them. This visualization is returned to the client to be displayed to the user.

14.2 Issues and Tradeoffs with Feature Portrayal Service

The only issue encountered was the varying time of responses from Feature Portrayal Services. Depending on the WFS used, a GetPortrayal request sometimes took as long as six minutes. The large size of the MSD3 feature type schemas was definitely a factor, but performance also determined the size of the bounding box used in the request.

14.3 Implementation Comments

14.3.1 Intergraph

Once the Intergraph integrated client is given the WFS endpoint (either by catalog query or user key-in) containing the data to be portrayed, the user is allowed to supply a symbology encoding document that defines the portrayal for each feature type served by the WFS. Then the user selects the appropriate Feature Portrayal Service and supplies addition information such as the height, width, and geographical bounding area. Then a GetPortrayal request is made to the service and a image is returned.

Once the image is returned to the client, it is written to local memory and stored in a warehouse connected to the commercial GeoMedia platform. At this point the recently acquired image is added to the MapView legend using standard commercial rendering capabilities.

14.3.2 Refractions Research

Feature Portrayal Service support was used in Refractions GeoDSS client to support two workflows; the visualization of WFS 1.0 with GML3 content, and the display of information using MIL2525B Symbology.

An important aspect of this implementation is the seamless access of the catalog for the discovery of Symbology Encoding documents. A user control preference (EMS or MIL2525B) is used to determine which request is made. After this single configuration operation all subsequent use of FPS is automatic.

14.4 Recommendations

The approach used for the Feature Portrayal Service has been successful. The separation of concerns represented by the use of the SE document and the OWS Context document represent an exciting possibility for moving the SLD specification forward.

The FPS represents a service of pure process (it does not hold any intrinsic resources of its own). This is the important difference between an FPS and a WMS. It is hoped that this example of transparent service chaining will be expanded to work with other services.

15 Data Aggregation Service (DAS)

A Data Aggregation Service is used to both remodel schemas from one data model to another as well as aggregating two or more WFS datasets. The output is another WFS endpoint that is accessed with the same GetCapabilities, DescribeFeatureType, and GetFeature requests. This section documents the process of using a Data Aggregation Service directly from an Integrated Client.

15.1 Direct Access to a Data Aggregation Service

Direct communication with a Data Aggregation Service occurs with an integrated client as a WFS interface. One additional command (SetMappingFile) is available to allow the integrated client to configure the Data Aggregation Service in order to set up the Mapping File and WFS endpoints to be used. Once the SetMappingFile request is made, all additional interaction is through the same interfaces as a WFS endpoint.

A) Data Aggregation Workflow

In the following example, an integrated client is used to connect to and retrieve feature data published as a WFS in a Data Aggregation Service.

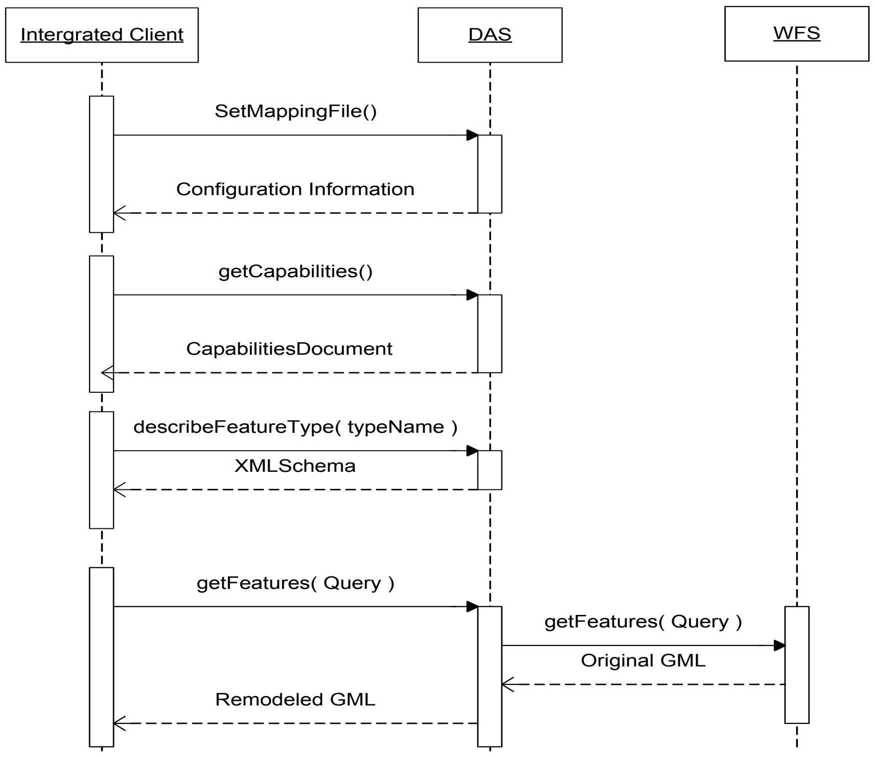


Figure 20 Remodeled Feature Retrieval

In this example an integrated client is provided with the URLs of the Data Aggregation Service, one or more WFS endpoints, and the Mapping Definition File. A SetMappingFile request is made to the DAS endpoint to inform the service of the WFS endpoints and the mapping definition to be used. After the DAS is configured, the CapabilitiesDocument is retrieved via a GetCapabilities request. The CapabilitiesDocument provides information detailing the available information by **typeName**, and additional information is retrieved in the form of an XMLSchema document by way of a describeFeatureType request. Using the description of this FeatureType the client can now construct a WFS GetFeatures request and retrieve information in the form of GML.

15.2 Issues and Tradeoffs with Web Feature Service Communication

15.2.1 Mapping File Definition

The Mapping Definition File is not currently robust enough to handle many of the mappings needed in basic remodeling activities. It does appear the structure is there to include many of these and they could be added rather easy. However, more thought should be given to exactly what should be included in the context of remodeling over the web.

15.2.2 Filter Capabilities

During the OWS-3 testbed, we attempted to execute the Data Aggregation Service against a particular dataset and it eventually failed. This appeared to be due to the fact that we were accessing a WFS that served a set of world wide features (e.g. all the roads in the world). This quickly and clearly identified the fact that the DAS should be enhanced to include a mechanism of defining an area of interest (or spatial filter) to limit the returned feature to only the instances within the interest area. In addition, the possibility exist for additional filters outside of spatial filters could also be implemented.

15.3 Implementation Comments

15.3.1 Intergraph Corporation

Since the DAS endpoint is published to look like a WFS, we use the existing WFS access capabilities within the GeoMedia client to bind and retrieve the mapped data (See the section detailing the WFS interfaces).

In addition, the client provides an interface that allows the user to populate fields with the endpoints of the Mapping Definition File and the WFS URLs to be mapped and aggregated. Then the user selects a button that makes a SetMappingFile request to the DAS and configures the DAS with the supplied information.

15.4 Recommendations

15.4.1 Mapping Definition Enhancements

Additional work needs to be done to enhance the capabilities defined in the Mapping Definition File. It is understood that it is possible to define the capabilities to map schemas so they are so complex that it make them difficult to use. However, during the OWS3 testbed there were some use cases identified that the current mapping definition and DAS service were unable to handle due to current limitations. Some of these included the following:

1. Ability to map enumerated values such as codelists.

2. Apply functions to the remapping to perform such operations as converting from feet to meters.

This process should take the time to plan what operations make sense for the DAS to perform and then attempt to model a schema for the mapping definition and publish a specification defining the mapping definition separate from the service.

15.4.2 Data Aggregation Service Configuration

Currently, the Data Aggregation Service can be configured by the user through an extended (and hidden) request known as “SetMappingFile”. This request tells the Data Aggregation Service that until otherwise requested, use the information in this SetMappingFile request. The request contains the name of the mapping definition file and the WFS endpoint(s) to be used by the service.

Although this solution worked for the testbed, it does not seem consistent. If the DAS is truly an extension of a WFS, then the current WFS specification requests should be extended to include the SetMappingFile request or extend current requests (GetCapabilities, DescribeFeatureType, GetFeature) to allow optional parameters to inform a DAS of the endpoints to use. However, if this should be its own service then a separate specification should be written and defined.

16 GeoVideo Service (GVS)

A GeoVideo Service provides an integrated client access to live or archived video data and associated technical and geospatial metadata. This section documents the process of accessing a geovideo services from an Integrated Client.

16.1 Connecting to a GeoVideo Service

The GeoVideo Service as currently implemented does not lend itself well to the publish/find/bind archetype, and so only the client/server interaction is described below.

16.1.1 Discovery of Object Schemas

A getGVSObjectDescr() request is issued to retrieve an array of CObjectDescr objects. These objects provide access to metadata regarding the role of each object, as well as it's attribution.

16.1.2 Discovery of Feed Data

Once the schema is known, this information can be used to discover the feed the user is interested in either tabularly, or graphically.

A) Tabular Discovery

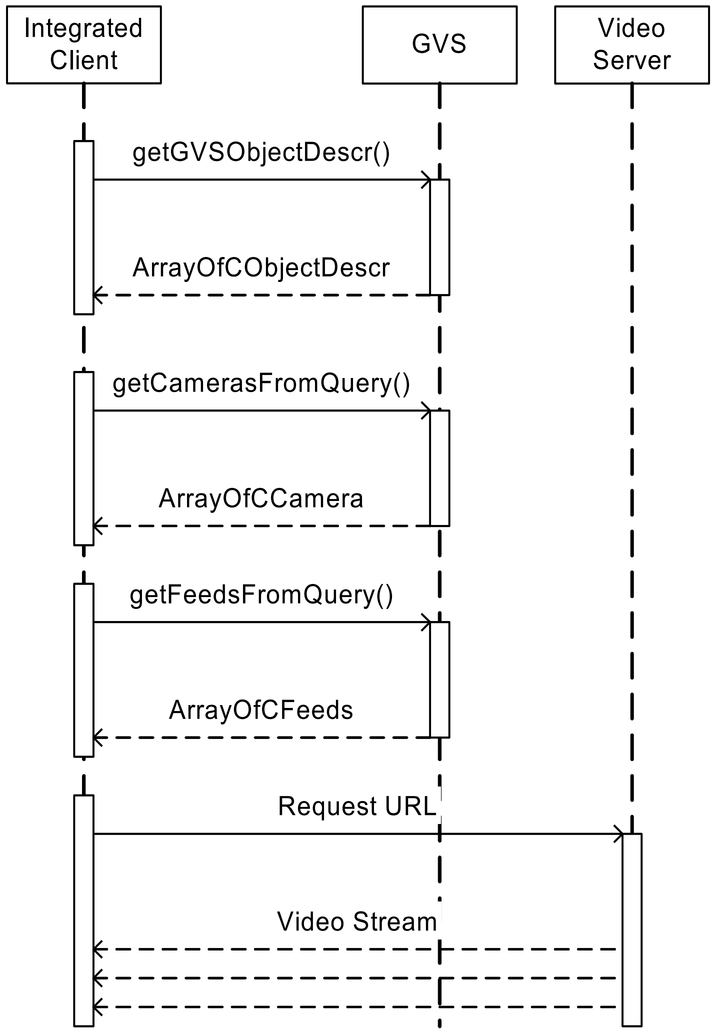


Figure 21 Tabular Discovery

To retrieve a list of geovideo objects, two method calls are available, getFeedsFromQuery and getCamerasFromQuery, which return arrays of feed and camera objects respectively. The syntax for making the requests is the same. A CQueryObject is created with the objectName reflecting the object of interest ('feed' or 'camera'). A criteria string, conforming to the syntax of an SQL where clause is used to narrow the search, and an ArrayOfCSpaceTimeBounds object describes the spatial-temporal extents of interest. The CSpaceTimeBounds object contains flags indicating whether the search is to match videos that start before or end after the given times, span the times or fall entirely between them. It also defines the time endpoints, and a bounding box to restrict the search.

In the case of the getFeedsFromQuery request, the result is an ArrayOfCFeed object that contain data about the feed, including a URL that can be used to request the geovideo stream.

The result of a getCamerasFromQuery request is an ArrayOfCCamera object. The camera id field can be used in a getFeedsFromQuery request. The purpose of these methods is to distinguish the properties of a feed from those of a camera.

This method of discovery is illustrated in the following diagram.

B) Graphical Discovery

Graphical discovery allows the client to display a live set of points over a map, and select one based on it's location.

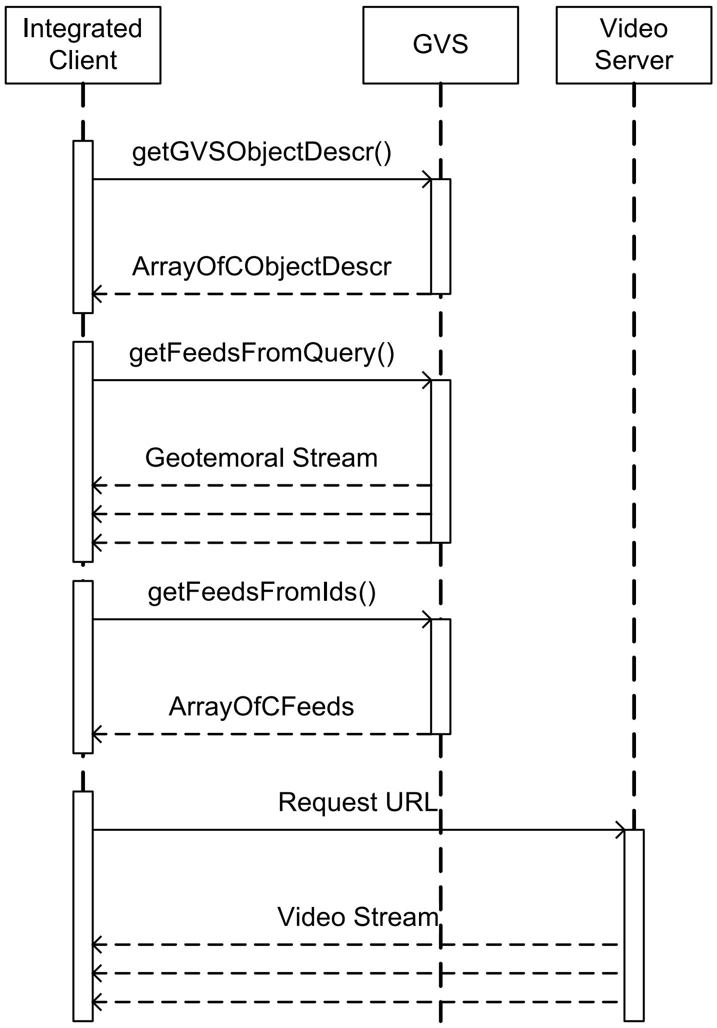


Figure 22 Graphical Discovery

There is no method defined to allow a stream to be requested to include only those feeds within a specified area. A getFeedsFromQuery request is used, with the added condition of "source like '%boundingbox%'" added to the criteria clause. This will provide an ArrayOfCFeed object as before, but the feeds included will be those that contain id, temporal and location data for multiple geovideo feeds.

C) Connecting to a Video Stream

After a feed has been selected as described above, the feed data can be retrieved using a getFeedsFromIds. Each CFeed object contains a 'URL' field that is used to connect to the geovideo stream. The geotemporal data is embedded in the closed-caption field of the video stream and must be extracted and parsed.

16.2 Issues and Tradeoffs with GeoVideo Service Communication

16.2.1 WSDL Interface

The WSDL interface used allowed for rapid development of a demonstratable geovideo server and client. This interface, however, does not lend itself well to the publish/find/bind interface favored by the OGC. In this case, the WSDL document stands in for the standard capabilities document, in as much as it defines the applicable operations. It does not provide any means of imbedding metadata, such as available camera types, formats or even a bounding box. This metadata must be retrieved through additional requests to the service. As the interface stands, there is no means, short of a full scan of all objects offered by the service, to retrieve such aggregate data as temporal or geospatial extents.

16.2.2 Area of Interest Feed

The current method of acquiring a stream containing geotemporal data on multiple geovideo feeds was implemented to expedite development and does not fully meet the original goal of subscribing to a feed based on feed type and area of interest. It assumes the preexistence of such a feed, and provides no method for limiting the number or type of feed described within the stream, instead providing filtering capabilities for the feed itself.

Additionally, requiring a specific query element puts restrictions on the implementation of the server; any feed with a source containing "boundingbox" will be treated as an area stream.

16.2.3 Video / Geotemporal Formats

The only format currently supported by the service uses MPEG4 as the video stream, and imbeds a variation on standard National Marine Electronics Association GPS data sentences. This allows participants to make use of existing tools and applications to parse and display the data instead of spending time defining and implementing redundant specifications.

16.3 Implementation Comments

16.3.1 Intergraph

The Intergraph client uses a specialized scripting component designed to understand the WSDL interface in order to build request and format the responses. Based on user selections, a set of properties are set on the object and the appropriate request is made. The response is processed and formatted into a display that the user can easily understand.

The user supplies the GVS endpoint (either through a catalog search or key-in of a known address) and a set of query properties. Using the query properties, a GetFeedsFromQuery request is made and a list of the available feeds are returned and displayed to the user. The user can select one of the displayed feeds and see the metadata information about the selected feed retrieved from the GetFeedsFromQuery. Once the appropriate feed is identified and the user chooses to view the video, the integrated client instantiates the Windows Media Player component and sets the URL property. In addition to the video playing, a closed caption string is also sent through the video stream. The client captures an event (_ScriptCommand event) from the WMP component when the closed caption is updated. The event contains the closed caption string that is then parsed to obtain the geolocation information. A graphical component is then built based on the geolocation information and displayed in the MapView window. As the closed caption (geolocation information) is updated, so is the graphical component in order to show the tracking of the video source.

16.3.2 Refractions Research

16.3.2.1 WSDL / SOAP

The Refractions integrated client utilized the JAXRPC libraries included in the Java Web Services Development Pack from Sun. This package included a tool that statically generated SOAP proxy objects based on the services WSDL document. The biggest restriction with this approach is the lack of robustness resulting from static code generation. Not only were the structures of request and response objects fixed, but the service URL as well. This meant that every time the service changed in any manner, the code had to be regenerated, recompiled and retested. A further problem occured when the service substituted a CFeed element for the abstract CObject element using the syntax <CObject xsi:type="CFeed">. When parsing the type value, the default namespace was not applied, and thus the element failed to validate. This problem was solved by breaking the original generic getObjectsFromQuery request into two requests: getFeedsFromQuery and getCameraFromQuery.

Another library considered for use was Axis2 from the Apache Software Foundation, but it had not yet reached it's 1.0 release, and was not yet stable enough.

16.3.2.2 Feed Parsing and Display

The original proposal indicated that Java Media Framework would be used in the client to display the video stream provided by the service. Three such projects were considered, and are described in Table 1.

Table 1: JMF projects considered

Project	Description	Reason for Rejection
Omnivideа FOBS4JMF	Open source JMF wrapper for ffmpeg CODECs.	No closed caption support.
Jffmpeg	Open source JMF combining pure java and ffmpeg ported CODECs.	No closed caption support, poor community support.
alphaWorks MPEG-4 for JMF	Pure Java implementation of MPEG-4 CODEC	Licensing incompatibilities.

Ultimately, the short timeline precluded us from using either FOBS4JMF or Jffmpeg, as the caption support that was available in their underlying C libraries had not yet been added to the java wrappers.

Windows Media Player was embedded in the client using the OLE bridge available in SWT. This allowed us to show a familiar video interface, while leveraging WMPs caption support to provide notification of caption events in the stream. While the use of WMP allowed the client to be developed in a reasonably short timeframe, it did compromise the platform independence of the final application.

16.4 Recommendations

16.4.1 Use NEMA Standards

The reuse of existing standards, from RIFF and NEMA, were used effectively to speed the development process. The NEMA standards were repurposed from their traditional role in a GPS data stream and included in the MPEG-4 closed captioning fields. This example of combining existing standards should be an encouraged process in future projects.

Although our creation of the *$GVDTL sentence* represents an extention to the NEMA standard the implementation could be brought to their attention. An alternative is using several of their *sentences* in succession to capture location, time and target.

16.4.2 Web Services Descriptor Language (WSDL)

The use of WSDL to define a service is in keeping with the current fashion of constructing an Service Oriented Architecture (SOA) system. The presentation of Open Web Service should be upgraded for presentation to members of the public trained in modern SOA technologies.

The use of WSDL presented many technical challenges, and presents an example of how the venerable architecture defined in the late nineties can still be relevant. While WSDL may not be an appropriate choice of technology its exploration, and possible guidelines for use should be explored.

17 Sensor Observation Service (SOS)

A sensor is used to observe a number of types of data including smoke, chemical, radiation, and weather. This section documents the process of using a Sensor Observation Service directly from an Integrated Client.

For the purpose of the OWS-3 testbed, the Integrated Client was limited to in-situ sensors. However, the discussions within this document can be applied to most sensor types.

17.1 Direct Access to a Sensor Observation Service

Direct communication with a Sensor Observation Service opens an integrated client to a host of data possibilities.

A) SensorML Observations

In the following example, an integrated client is used to connect to and retrieve observations information published in a Sensor Observation Service in SensorML format. An integrated client is provided with a URL of a Sensor Observation Service. The CapabilitiesDocument is retrieved via a GetCapabilities request. The CapabilitiesDocument provides information detailing the available offerings. This information includes the type of phenomenon measured and the general location of the sensor. Given a sensor id from the list of offerings, a GetObservation request is made to retrieve the readings provided by the sensor.

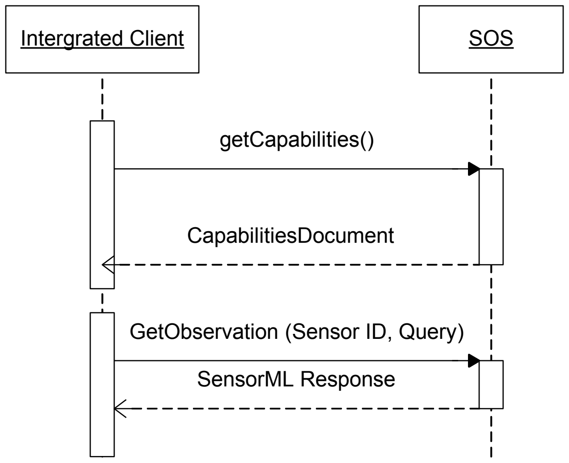


Figure 23 Observation Retrieval (SensorML)

B) TML Observations

In the next example, the response to the request is an endpoint to a TML stream. An integrated client is provided with a URL of a Sensor Observation Service. The CapabilitiesDocument is retrieved via a GetCapabilities request. The CapabilitiesDocument provides information detailing the available offerings. This information includes the type of phenomenon measured and the general location of the sensor. Given a sensor id from the list of offerings, a GetObservation request is made to retrieve the endpoint to the TML stream provided by the sensor. The integrated client connects to the TML stream and reads the latest sensor readings.

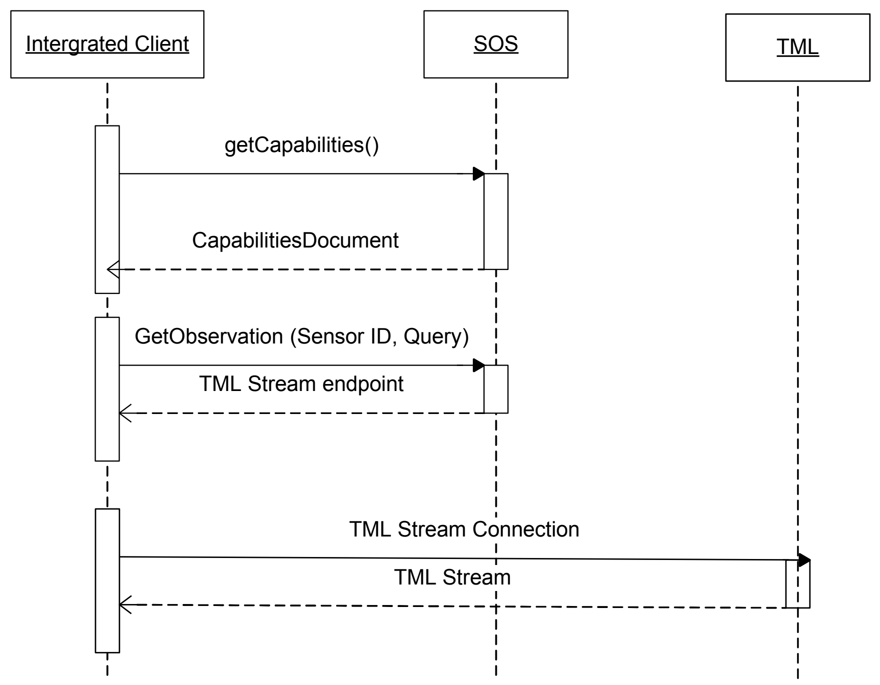


Figure 24 Observation Retrieval (TML)

17.2 Issues and Tradeoffs with Sensor Observation Service Communication

17.2.1 Observation Response Types

One of the difficulties encountered during the integration of the client with the Sensor Observation Service was the variety of response types. There are two ways in which responses can affect a client:

A) The format of the observation response

The format of the observation response can vary in many different ways. The common observation model allows this by providing a “header” that describes the format of the content. Although this “header” provides a mechanism for interpreting the data, it leaves the format very open for the implementer to create a customized response. This can be good in that it is open enough to allow for the many types of sensor data. But it places a lot of burden on the integrated client to be able to understand and process a number of formats.

B) The format of the response

Another issue with the SOS response is seen in the format of the response itself. The integration of the client and the SOS began by examining the responses from different service providers. What was discovered was that one provider’s response was in the form of the Common Observation Model while the other was in the form of a TML stream. It is understood that this provides mechanisms to “snapshot” the sensor’s observation as well as “stream” its data continually. However, from a client’s perspective, this seems to be two different implementations and requires a client to develop software to handle the responses through differing paths.

17.2.2 Geolocation Information

Due to the variety of sensor types, it can be difficult to determine the location of the sensors. Part of the reason for this is the software’s need to differentiate between the location of the sensor and the location of the observation. For example, an in-situ sensor that represents a weather station measuring temperature would be represented as a point with its geo-location information identified in the Capabilities response. Although the sensor can measure a number of phenomena, it is still a single sensor at a single location. However, a group of systems can be built such that they are represented as a single offering but are treated as multiple sensors. In this case, the geo-location is represented as a bounding box showing the complete area covered by all sensors in the offering. It appears that these types of sensors identify their geo-location as part of the phenomenon in the GetObservation response. However, there is no requirement for location to be part of that list of phenomenon and if it is included, the format could take a number of formats.

An integrated client must first determine if geo-location information is available from the GetCapabilities response. If it is available, does it actually represent the location of the observation? To answer that question, the client must then access the available phenomena and determine if the observation includes location information. This is currently being accomplished by looking for phenomenon with names resembling Latitude, Longitude, Height, Altitude, etc. However, again there is no required way to represent geo-location information. The burden of determining this is placed on the client software with no help in determining what or how should be searched to find the location information. This seems inconsistent with OGC since location information is key to most, if not all, of the current services being developed.

17.3 Implementation Comments

17.3.1 Intergraph Corporation

The Intergraph client uses a component designed to build request and format the responses. Specific properties are set on the object then either a GetCapabilities or GetObservation request is made. The response is processed and formatted into a list that can be easily traversed.

The user simply supplies the SOS endpoint and the integrated client makes a GetCapabilities request as the initial communication to the service. Then the user is presented with a list of offerings available from the given endpoint. The user selects the desired offering and supplies a timeframe in the form of a time instance or time range. A GetObservation request is then constructed and a list of observations is returned detailing the phenomenon. The observation data is stored in the GeoMedia environment in the form of a data query. By storing the data this way, the user can then view the information in a data window or plot the location in a MapView window.

17.4 Recommendations

Whether it is for military, local governments, or emergency response, the direct use of Sensor Observation Services will become an increasing part of the open web service environments. It is for this reason that the interfaces for sensor observations must become more consistent. The response formats need to express location information consistent with one another. This is especially true when it comes to the observation response. When a client wants to see if location information is included in the phenomenon, it should be easy to locate by identifying a particular property or a particular phenomenon name.

It is understood that implementers may provide information back in different formats similar to the way a WMS can return an image as a jpeg or png format. However, a client should be able to expect the responses from a GetCapabilities and GetObservation requests to be consistent to the point that both can be easily parsed and a determination made as to its ability to handle the returned information. This involves more than simply informing the client of the format, but also where to obtain the location of the observation and more consistency in the tags that are used to reference particular information.

18 Sensor Planning Service (SPS)

A Sensor Planning Service is used to task a particular sensor in order to get the desired observation. This can involve tasking a wind sensor to take a reading at a given time or tasking a camera to zoom in for a better view through its video feed. This section documents the process of using a Sensor Planning Service directly from an Integrated Client.

18.1 Direct Access to a Sensor Planning Service

Direct communication with a Sensor Planning Service allows an integrated client to control and task a given sensor.

A) Sensor Planning Service Example

In the following example, an integrated client is used to connect to and request information published in a Sensor Planning Service.

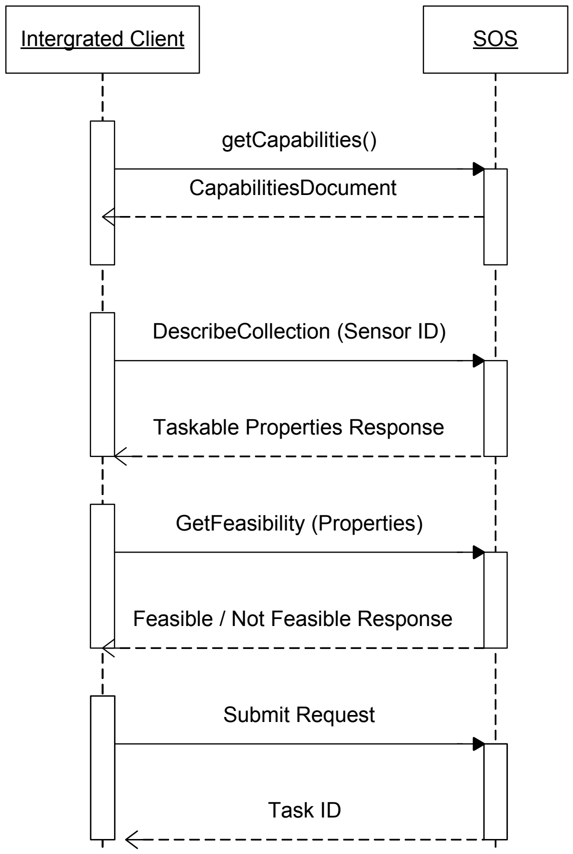

Intergrated Client
SOS
getCapabilities()
CapabilitiesDocument
DescribeCollection (Sensor ID)
Taskable Properties Response
GetFeasibility (Properties)
Feasible / Not Feasible Response
Submit Request
Task ID

Figure 25 Planning Service Interface

In this example a integrated client is provided with a URL of a Sensor Planning Service. The CapabilitiesDocument is retrieved via a GetCapabilities request. The CapabilitiesDocument provides information detailing the available offerings. This information includes the type of phenomenon measured and the general location of the sensor. Given a sensor id from the list of offerings, a DescribeCollection request is made to retrieve the phenomenon available for tasking on the sensor. The user enters the information detailing the desired task and submits a GetFeasibility request to see if it is feasible to ask the sensor to perform the task. If it is, the user can then make a Submit request to task the sensor. Although not available on all taskable sensors, many will communicate back to the user that the task has completed through a notification service.

18.2 Issues and Tradeoffs with Sensor Planning Service Communication

18.2.1 Notification Responses

Although there are not a lot of issues with the Sensor Planning Service, there seems to be room for improvement in the area of notification responses. This involves the question of how a client can know when the requested task has been completed. Some tasking events happen almost immediately and the results can be seen right away such as tasking a camera to pan or zoom. In these cases, the idea of notification (although still a possibility for the sake of consistency) is not seen as a necessity for the user. However, in the cases where the user is submitting a task that may not occur until some time in the more distant future, such as tasking a UAV to collect imagery over a certain area with cloud cover at a minimum, the end of the task would be unknown. The user should expect some form of notification detailing the task has been completed and possibly what needs to be done to retrieve the information.

Currently, there are several methods involving notification from simple email to subscribing to a notification service. And although each of these serves the purpose of informing the user, it makes it difficult for client developers to know how to share this information with the user and interface to open-ended methods of notification.

18.3 Implementation Comments

18.3.1 Intergraph Corporation

The Intergraph client uses a component designed to build request and format the responses. Specific properties are set on the object then the appropriate request is made. The response is processed and formatted into a list that can be easily displayed.

The user simply supplies the SPS endpoint and the integrated client makes a GetCapabilities request as the initial communication to the service. Then the user is presented with a list of offerings available from the given endpoint. The user selects the desired offering and a DescribeCollection request is made in order to display the taskable properties for the given sensor. The user sets the desired properties and a GetFeasibility request is issued to determine if the request is valid. If the response to the GetFeasibility request is true, then the user can Submit the request and receive a task id in the response.

18.4 Recommendations

The use of sensors in monitoring activity and readings will become increasingly more popular in the web service environment. It is for this reason that we recommend a continued effort to study and publish additional Sensor Planning Services in order to get a better understanding of the role of planning / tasking as it pertains to the many sensor types currently available.

19 Open Web Services Context Document

The use of an OWS Context document allows collaboration between end-users using different OGC compatible clients. The use of a CSW to share context documents is recommended.

Specifically an “OWS Context Document” states how a specific grouping of one or more WMS Layers, WFS Feature Types or WCS Coverages can be described as an XML document. These descriptions are fairly complete allowing for inline style specification using SLD elements. An OWS Context also includes information about the area of interest, location and projection in addition to the usual set of descriptive metadata. There is also section for vendor specific extensions.

Over the course of our OWS-3 demonstration this facility was used to share a common operational picture of a disaster relief operation. We did not have time to explore the use of CSW for storage and retrieval, and instead relied on email and a WIKI page to share context documents.

Both the **WMS Context Document** and **Location Organizer Folder** (LOF) were considered for this roll. WMS Context did not meet our needs, and development of LOF appears to have ceased. OWS Context is the logical successor to these technologies.

An OWS Context document consists of the following:

- General element, description of the context including title, metadata, and area of interest information (bounding box and SRS). There is some support for vendor specific information at this level

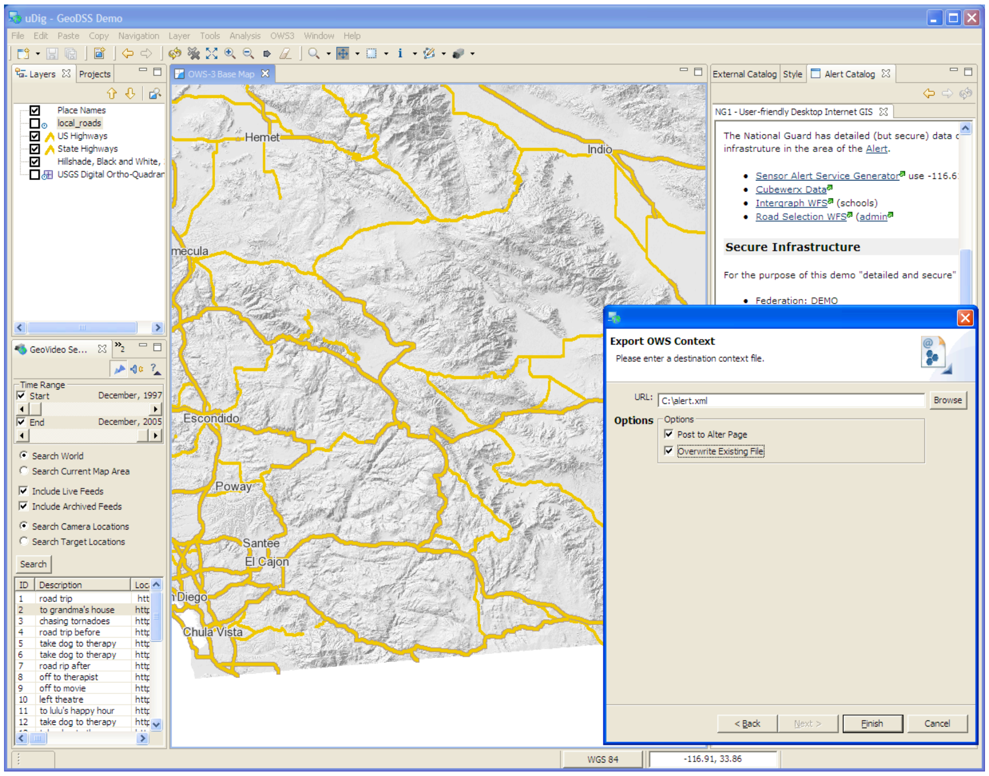
- ResourceList element, includes Layers, FeatureTypes and Coverages and associated styling information. Enough information is provided to connect to the associate service.

The OWS Context document should be thought of as a “bookmark”, as such the services associated with each layer will need to be contacted for their Capabilities before use.

19.1 Use of Context Document

The ability to share a common operating picture based on open web services is of great benefit and has many exciting opportunities for collarobration.

A) OWS Context Document Generation and Publication



Layers: ☑ Place Names; ☐ local_roads; ☑ US Highways; ☑ State Highways; ☑ Hillshade, Black and White,; ☐ USGS Digital Ortho-Quadran

Time Range: ☑ Start December, 1997; ☑ End December, 2005

(●) Search World ( ) Search Current Map Area

☑ Include Live Feeds ☑ Include Archived Feeds

(●) Search Camera Locations ( ) Search Target Locations

Search

ID	Description	Loc
1	road trip	htt
2	to grandma's house	http
3	chasing tornadoes	http
4	road trip before	http
5	take dog to therapy	http
6	take dog to therapy	http
7	road rip after	http
8	off to therapist	http
9	off to movie	http
10	left theatre	http
11	to lulu's happy hour	http
12	take dog to therapy	http

NG1 - User-friendly Desktop Internet GIS

The National Guard has detailed (but secure) data c infrastruture in the area of the Alert.

- Sensor Alert Service Generator use -116.6
- Cubewerx Data
- Intergraph WFS (schools)
- Road Selection WFS (admin

Secure Infrastructure

For the purpose of this demo "detailed and secure"

- Federation: DEMO

Export OWS Context

Please enter a destination context file.

URL: C:\alert.xml [Browse]

Options: ☑ Post to Alter Page ☑ Overwrite Existing File

< Back | Next > | Finish | Cancel

Figure 26 Open Web Service Document Generation

Client captures the current area of interest and displayed information and generates a OWS Context document. The document is submitted as an intrinsic object to a CS-W through the use of the catalog Transaction operation.

Often an integrated client will make use of additional layers not covered by OWS services (such as a legend, scale bar or local files) these layers are simply not included in the generated document.

In the course of this the OWS-3 demo we made use of a Wiki page for publication.

B) Discovery and Loading of a OWS Context Document

Client retrieves a context document, nominally through a catalog service. In the OWS-3 demo we made use of a Wiki page attachment.

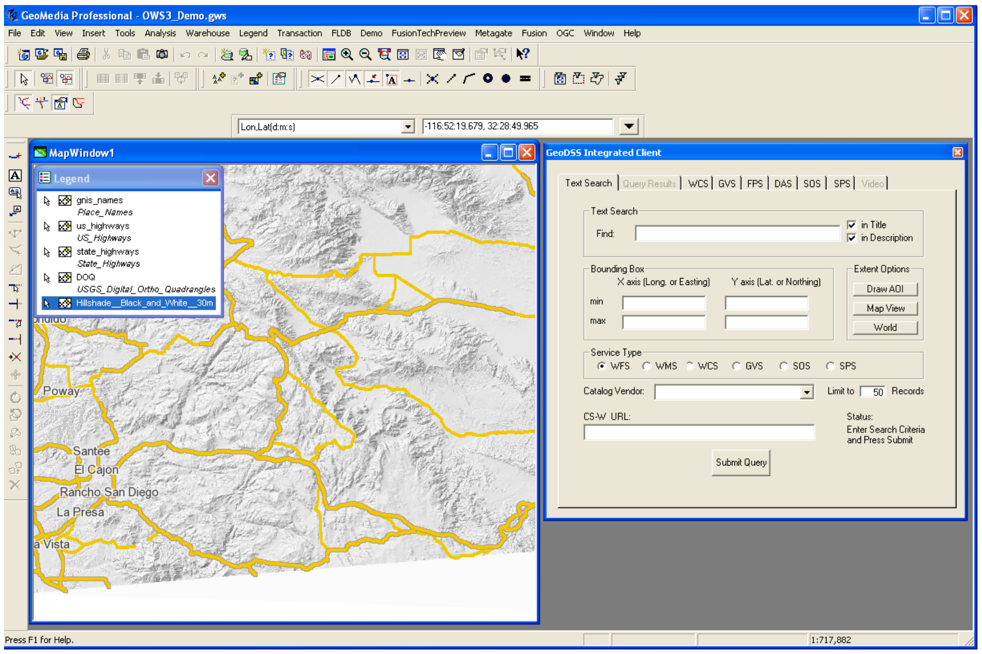


Figure 27 Import of Open Web Context Document

Since the OWS context document is not a heavy weight solution, some form of import process will be required by which an application will build up an internal representation of the provided context. During the course of this process the associated services will need to be contacted.

Both client implementations made use of a wizard to support this interaction, and supported direct access to a context document saved on the local file system. The refractions GeoDSS client made use of an embded browser allowing users to directly click on context documents attached to a Wiki page.

19.2 Issues and Tradeoffs with OWS Context Documents

The use of OWS Context Documents is hampered by a number of small issues. We expect the following to be resolved before standardization.

19.2.1 Clarification of SRS

The expected visualization could be made more explicit. There is a choice between displaying a visual based on SRS mentioned as an attribute of the Bounding Box element, and making a “vote” based on the number of SRS indicated as current for WMS layers.

We chose to make use of the Bounding Box SRS attribute, and ignore the currently indicated SRS chosen by the WMS layers. Indeed we were unsure if this would be used to indicate the requirement for client side raster reprojection on the part of an integrated client. Some WMS servers are known to serve up different information based on the projection, so this idea is not as far fetched as it sounds.

In all likelihood we simply missed where this information was mentioned exactly. We feel that the choice made was sensible.

19.2.2 Inconsistent Level of Detail

The context document has evolved from the WMS Context Document (which captured enough information to display an image with only references to OWS Operations). With the OWS Context Document the choice was made to require an integrated client to retrieve the Capabilities Document of the services utilized. This choice is sensible given the limitations a client is under, but its application is not consistent through-out.

The detail represented in the following items is inappropriate, given the fact that we will be retrieving the Capabilities Document before making further requests:

- SRS
  The use of SRS with the Layer element is especially problematic. Many of the services used allow for hundreds of supported projections. Carrying this dead weight around was made all the more annoying by our decision to ignore it (mentioned above).
  This issue is compounded by the child/parent Layer relationship maintained by the WMS Capabilities Document. Often the same list of SRS information will be duplicated several times in a single Context Document.

- Style
  Once again the list of available styles in not required by an application that expects to retrieve the Capabilities Document.

19.2.3 Extensibility

In its present format the OWS Context document does not represent an extensible medium. While this may only be an issue during these Interoperability Experiments (where new Open Web Services are proposed) it did represent a limitation we encountered.

We had planned to use the following additional services:

- Feature Portrayal Service: we repurposed FeatureType, and simply ignored the style information. A client should be able to define how to visualize the provided content, including making use of a FPS in that process.
  We explored the use of the extension elements as a mechanism to record the FPS used for portrayal.

- GeoVideo Service

Given the construction of the schema on a limited set of service types are supported. It is hoped that the contribution of a context document entry could be incorporated into the process of defining a new Open Web Service.

19.3 Implementation Comments

19.3.1 Intergraph

Although not originally in the scope of the Intergraph client, it was recognized that the inclusion of at least a read capability for the context document was important to show interoperability between the two clients as well as with the various services.

Since the demonstrations for this testbed only required the exchange of WMS and WFS services, we were able to again bonus off the commercial WMS and WFS dataservers. The Intergraph client read and parsed the OWS context document to retrieve the information needed to create a connection object with the GeoMedia platform for each of the unique endpoints. Once the connection was made, the software would automatically generate a legend entry for the given feature type or layer. If multiple feature types or layers were listed for a single endpoint, the software would simply use the same connection object and generate a new legend entry.

Once all data elements had been retrieved and added to the legend, the software would zoom the MapView window to the appropriate scale and size based on the values retrieved from the context document.

19.3.2 Refractions Research

Implementation was straightforward, given the limitations of the services in the demo we did not bother to generate inline style information for services mentioned. Parsing and processing of the document was achieved without undo effort. We made use of an additional checkbox on the export wizard to indicate that the result should be published.

Given the workflow used in the Demo we also did not make use of the Style information recorded for the FeatureType entries. These entries were all handled by the Feature Portrayal Service (FPS), and the Symbology Encoding Documents looked up according to application preferences at runtime.

We must also apologize for the following mistake:

- The double negative indicated by the “hidden” attribute confused us, that is hidden=1 is used to make something visible. We got it exactly wrong, and would like to recommend a “positive” phrasing: visisble=true or on=true

To make use of a OWS Context document we intercepted clicks in an embedded browser. When we noticed that a OWS Context document was being navigated to we loaded a new map rather then switching pages.

19.4 Recommendations

The OWS Context document is a very useful piece of technology, and we would like to encourage the further development of these ideas.

The separation of concern achieved by the use of a Context Document with a Symbology Encoding Document shows great promises for moving the Style Layer Descriptor specification forward. Many of the SLD 1.1 proposals, for the definition of new layers or inline GML, could safely be relocated to the OWS Context document.

20 Digital Rights Management (DRM)

Distributed Access Control System (DACS) allows resources and web services to be protected behind a single sign-in system. It can require that a user submit a proper username and password to access the protected resource. It can also present the user with notices that must be acknowledged before access can be granted.

20.1 Open Web Services access through DRM Protection

A) DRM using Distributed Acces Control System (DACS)

Accessing a resource protected by a DACS instance involves performing the request multiple times, each time examining the response and taking appropriate action.

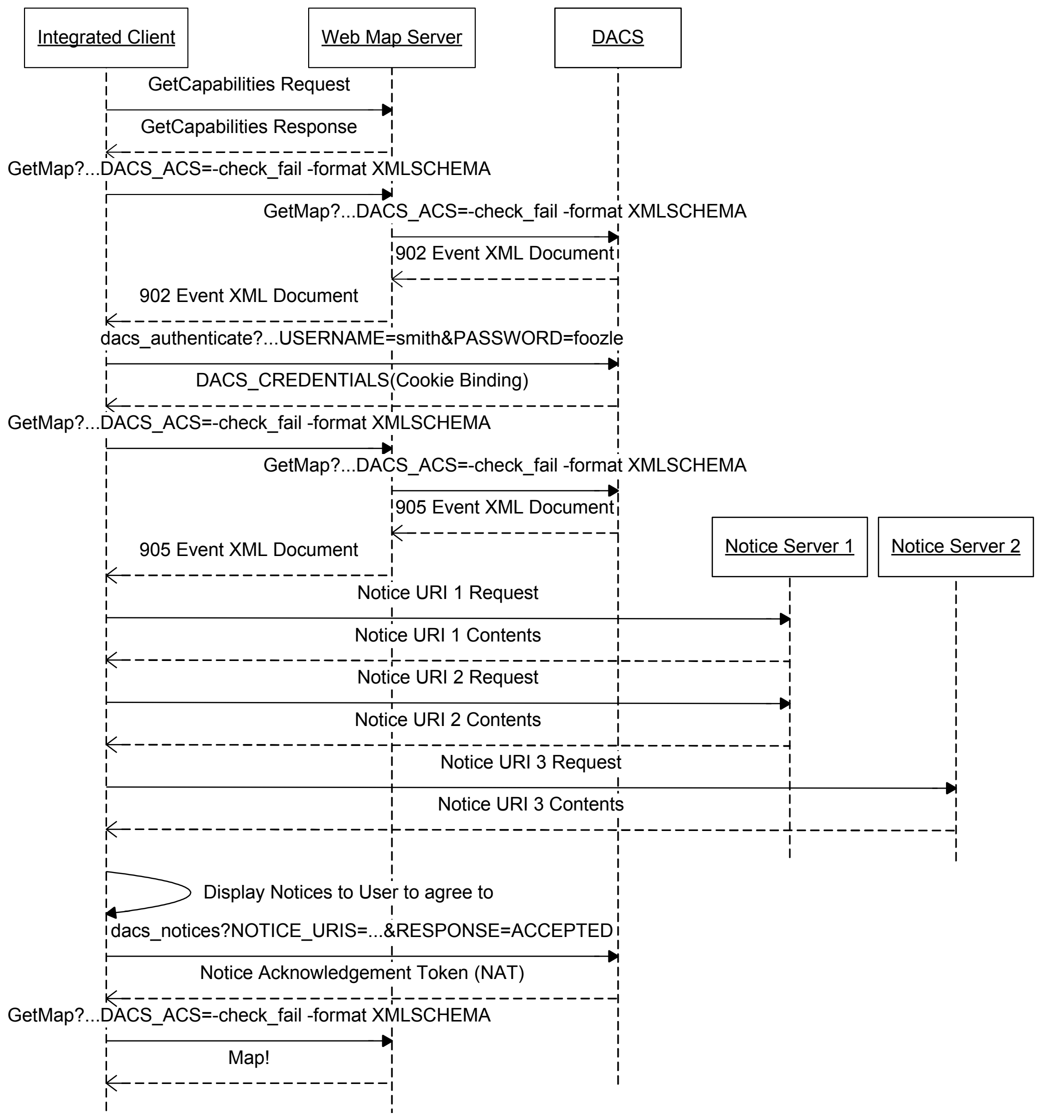

Integrated Client
Web Map Server
DACS
GetCapabilities Request
GetCapabilities Response
GetMap?...DACS_ACS=-check_fail -format XMLSCHEMA
GetMap?...DACS_ACS=-check_fail -format XMLSCHEMA
902 Event XML Document
902 Event XML Document
dacs_authenticate?...USERNAME=smith&PASSWORD=foozle
DACS_CREDENTIALS(Cookie Binding)
GetMap?...DACS_ACS=-check_fail -format XMLSCHEMA
GetMap?...DACS_ACS=-check_fail -format XMLSCHEMA
905 Event XML Document
905 Event XML Document
Notice Server 1
Notice Server 2
Notice URI 1 Request
Notice URI 1 Contents
Notice URI 2 Request
Notice URI 2 Contents
Notice URI 3 Request
Notice URI 3 Contents
Display Notices to User to agree to
dacs_notices?NOTICE_URIS=...&RESPONSE=ACCEPTED
Notice Acknowledgement Token (NAT)
GetMap?...DACS_ACS=-check_fail -format XMLSCHEMA
Map!

Figure 28 DRM Enabled Workflow

The above diagram shows the sequence of events for performing a GetMap request on a WMS that is protected by a DACS instance, requiring authentication and acknowledgement of notices.

At first a GetCapabilities request is made to the WMS instance. It is possible, although uncommon, for the Capabilities document to be protected by DACS. That case is not taken into consideration here.

A GetMap request is constructed and sent to the WMS instance. The WMS forwards the request to the DACS instance. If the DACS server responds with a 902 event (authentication required) or a 905 event (notice acknowledgement required), the event is forwarded to the Integrated Client for handling.

If the Integrated Client must provide authentication, it can ask the user to enter the required username and password. It then sends a request for authentication directly to the DACS server. If the username and password are correct, the DACS instance responds with a set of credentials that the Integrated Client must used in each successive communication with the WMS.

If the Integrated Client must provide notice acknowledgement, it can display each of the notices to the user. If the user agrees, a notice acknowledgement request is sent to the DACS service. The service responds with a Notice Acknowledgement Token (NAT) that must be used in each successive communication with the WMS.

Once all the preconditions have been satisfied, the WMS will return the requested GetMap image.

B) DRM using a Local Proxy

The use of DRM protected services is also available to clients that have not yet adopted DRM measures. The following technique represents an experiment conducted by the OWS DRM thread using opaque chaining.

A local proxy was provided, in the form of a Java applet. All DRM based communication was handled by this local proxy. The proxy opened up a port on the local machine which appeared to the GeoDSS client as a straightforward WMS instance.

This was a successful example of opaque chaining from the perspective of the client application, and represents a creative transitional approach. This approach was so successful that we cannot comment on the specific DRM techniques used, although we suspect they involved the use of a SOAP based WMS.

20.2 Issues and Tradeoffs

20.2.1 Cascading Credentials

Cascading protected resources can be bothersome from the user's perspective. If WMS A contains cascaded layers from WMS B, and both are protected by separate DACS jurisdictions, the user may have to log-in twice: once to access WMS A and again when WMS A tries to access WMS B.

From the perspective of an integrated client a DRM procedure may be engaged multiple times to obtain all the needed credentials.

20.3 Implementation Comments

The only problem with implementing DACS authentication and notification was encountered when a third party plug-ins were involved. In order to detect that authentication is required; the response from the server must be examined before it is passed on to the code that wishes to do something with it. If the response cannot be intercepted, DACS handling cannot be performed.

20.4 Recommendations

It is recommended that a security module be designed and implemented for an Integrated Client, performing handling, retrieval, and persistence of usernames and passwords.

21 Test Considerations and Results

This section provides an overview of the tests performed during the OWS-3.

21.1 Issues and Opportunities

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.

21.2 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.

21.3 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.

21.3.1 Web Feature Service (WFS) TIEs

	Intergraph Client	Refractions Client
Intergraph MSD3	✓ Success	✓ Success
CubeWerx VMAP0	✓ Success	✓ Success
NASA Ames (CaSIL)	✓ Success	
Galdos		
Refractions		✓ Success

21.3.2 Web Map Service (WMS) TIEs

	Intergraph Client	Refractions Client
Intergraph (dcmetro)	✓ Success	✓ Success
NASA Ames (CaSIL)	✓ Success	✓ Success
NASA WMS (onEarth global_mosaic)	✓ Success	✓ Success
Terra Service (UrbanArea)	✓ Success	✓ Partial Success
Terra Service (DOQ)	✓ Success	✓ Success
DEM (onEarth DEM)	✓ Success	✓ Success

21.3.3 Web Coverage Service (WCS) TIEs

	Intergraph Client	Refractions Client
UAH GOES	✓ Success	
UAH SRTM		
SPOT Image satellite imagery		

21.3.4 Feature Portrayal Service (FPS) TIEs

	Intergraph Client	Refractions Client
Object FX FPS	✓ Success	✓ Success
Galdos FPS	✓ Success	✓ Success

21.3.5 Data Aggregation Service (DAS) TIEs

	Intergraph Client	Refractions Client
Intergraph DAS	✓ Success	
CAST DAS		

21.3.6 Sensor Observation Service (SOS) TIEs

	Intergraph Client	Refractions Client
UAH Weather Stations	✓ Success	
UAH Plume	✓ Success	

IRIS Smoke/Chemical SOS	✓ Success	
3ETI Chemical Sensors	✓ Partial Success	

21.3.7 Sensor Planning Service (SPS) TIEs

	Intergraph Client	Refractions Client
University Muenster (IFGI)	✓ Success	
NASA Ames		

21.3.8 GeoVideo Service (GVS) TIEs

	Intergraph Client	Refractions Client
Intergraph GVS	✓ Success	✓ Success

21.3.9 Catalog Services for the Web (CS-W) TIEs

	Intergraph Client	Refractions Client
Ionic Catalog	✓ Success	✓ Partial Success
CubeWerx Catalog	✓ Success	✓ Success
NASA Catalog		✓ Partial Success
ESA Catalog		✓ Partial Success

21.3.10 Context Document TIEs

	Intergraph Client	Refractions Client
OWS Context (during demo)	✓ Partial Success	✓ Success

21.3.11 Digital Rights Management (DRM) TIEs

	Intergraph Client	Refractions Client
DACS / DRM		✓ Success

21.3.12 Sensor Alert Service (SAS) TIEs

	Intergraph Client	Refractions Client
3eti		✓ Success

22 Summary

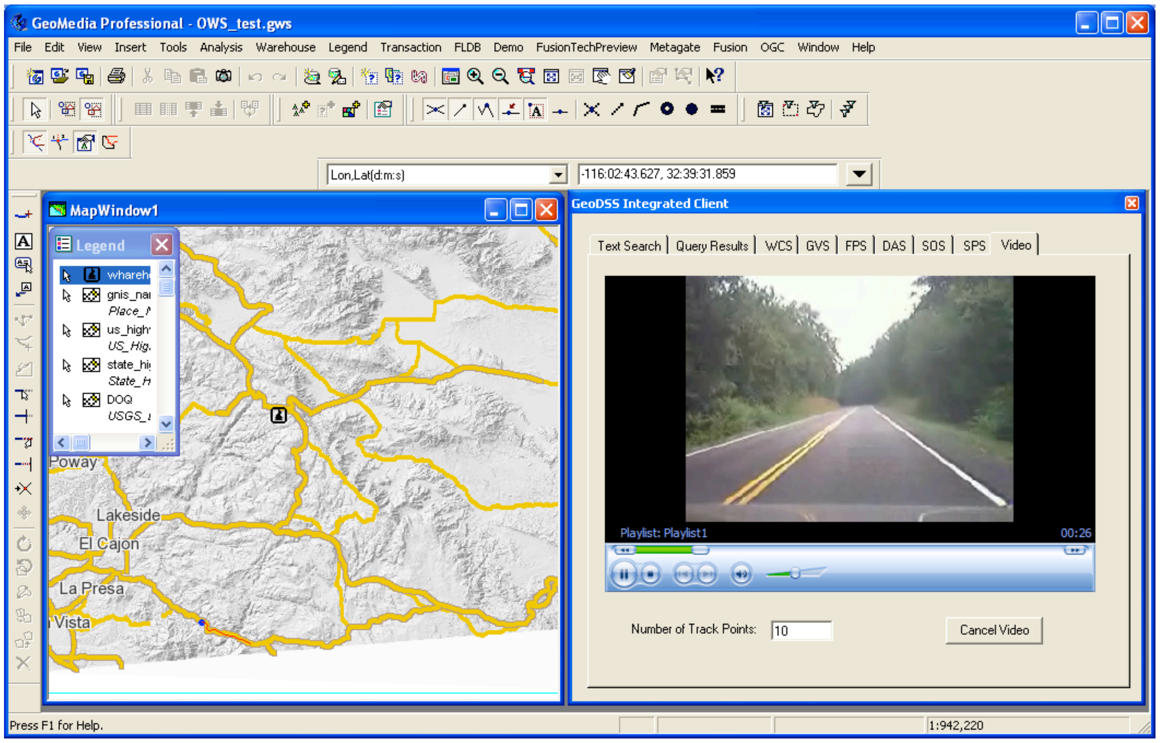
In this document, we have addressed a variety of topics concerned with clients that integrate the ability to access several OGC-compliant services within a single application, and merge the acquired data into a single map or information display. We have touched on various aspects of client architecture, 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 two 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 are the best way to gain deeper insight into the relative merits of different approaches to client creation.

Annex A: TIE Details

1 Intergraph Corporation

1.1 Intergraph GVS


GeoMedia Professional - OWS_test.gws
File Edit View Insert Tools Analysis Warehouse Legend Transaction FLDB Demo FusionTechPreview Metagate Fusion OGC Window Help
Lon,Lat(d:m:s)
-116:02:43.627, 32:39:31.859
MapWindow1
Legend
Poway
Lakeside
El Cajon
La Presa
Vista
GeoDSS Integrated Client
Text Search | Query Results | WCS | GVS | FPS | DAS | SOS | SPS | Video
Playlist: Playlist1
00:26
Number of Track Points: 10
Cancel Video
Press F1 for Help.
1:942,220

url:	http://gvs.intergraph.com/GVSBeta/Service.asmx
Status:	Accessed the WSDL document and was able to see a list of offerings. From the list of offerings we could access the video stream and pull off the geolocation information in the closed caption string. Once the video was playing we were able to track the location of the camera in the MapView. We were also able to track multiple camera feeds supplied by the service within the MapView window.

1.2 GML SAT (CAST)

url:	http://ogc.cast.uark.edu:9000/SAT/servlet
Interaction:	The ability to bring up the browser interface to the tool has been embedded in the Ingr client and can be launch from the user interface.

1.3 FPS (Galdos)

url:	http://ows3.demo.galdosinc.com/fps/http
wfs url:	http://regis.intergraph.com/wfs/casil/request.asp
feature type(s):	schoolsa, hlthfaca
se url:	http://mse.galdosinc.com/ows3/se/EvacSchools-EMS-SE.xml http://mse.galdosinc.com/ows3/se/HealthFac-EMS-SE.xml
status:	Successfully requested and received both a png and a jpg image. That image was then automatically saved in a GeoMedia[?] warehouse and displayed in the MapView[?].

1.4 FPS (ObjectFX)

url:	http://demo.objectfx.com/OWS3/fps
wfs url:	http://www.bsc-eoc.org/cgi-bin/bsc_ows.asp
feature type(s):	BBS_PT
se url:	http://demo.objectfx.com/OWS3/encodings/Birds-BBS_PT-SE.xml
status:	Successfully requested and received a png image of the BBS_PT feature. That image was then automatically saved in a GeoMedia[?] warehouse and displayed in the MapView[?].

1.5 DAS (Ingr)

url:	http://zx10.ingr.com/DAS/request.asp
date:	9/19
status:	Successfully aggregated data using a mapping definition file created by hand. This TIE included modifying the schema of one WFS (change attribute names, insert

	new attributes, etc.) and then unioning with another WFS URL to create a single output.

1.6 DAS (CAST)

DAS:	unknown
Status:	Untested

1.7 Data

Drag and Drop of WMS and WFS urls is supported. WFS 1.0 supported, WFS 1.1 unsupported, we will make use of FPS to display this content. We do not currently make use of WMS SLD Post.

1.7.1 WFS-TestData

Intergraph

WFS:	http://regis.intergraph.com/wfs/casil/request.asp?SERVICE=WFS&VERSION=1.0.0&REQUEST=GetCapabilities
Status:	Successfully located the data through the CS-W query, connected to the service and displayed the data in the MapView. This data was also used to feed into the Galdos FPS.

Galdos

WFS:	unknown
Status:	Untested, no URL given for testing.

CubeWerx

WMS:	http://demo.cubewerx.com/demo/cubeserv/cubeserv.cgi?service=WFS&datastore=Foundation&request=GetCapabilities
Status:	Success.

Refractions GeoServer[2]

WFS:	http://www.refractions.net:8082/geoserver/wfs/GetCapabilities
Status:	Success

1.7.2 Web Map Server

Public WMS urls used in demo:

Terra Server:

WMS:	http://terraservice.net/ogccapabilities.ashx?version=1.1.1&request=GetCapabilities&service=wms
Status:	Success.

NASA OnEarth[?]

WMS:	http://onearth.jpl.nasa.gov/wms.cgi?request=GetCapabilities&service=wms
Status:	Success.

NASA Ames

WMS:	http://sggate.arc.nasa.gov:9518/cgi-bin/casil-wms
Status:	Success.

NASA Ames

WMS:	http://sggate.arc.nasa.gov:9518/cgi-bin/casil-wms?request=GetCapabilities&service=wms
Status:	Success.

1.7.3 Web Coverage Service

UAH GOES

WMS:	http://vast.uah.edu:8080/sttserv/servlet/ServicesServlet
Status:	Success.

UAH SRTM

WMS:	http://vast.uah.edu:8080/ows/srtmWcs
Status:	Not tested.

SPOT Imagery

WMS:	http://ws.spotimage.com/sisa_wcs_sd/coverage/SS5_031105
Status:	Not tested.

1.8 Other Services

1.8.1.1 Ionic Catalog

url:	http://dev.ionicsoft.com:8080/catalog230/wrs/WRS
status:	Able to successfully query the Ionics catalog looking for WFS, WMS, and SOS entries. Received numerous results and utilized this interface during the demo.

1.8.1.2 Cubewerx Catalog

url:	http://demo.cubewerx.com/ows3/catalog/cwwrs.cgi
status:	Able to successfully query the Cubewerx catalog and receive results. Primarily used CubeWerx catalog to perform WFS queries.

1.8.1.3 NASA Catalog

url:	http://orion.compusult.net/wes/serviceManagerCSW/csw
Status:	Not tested.

1.8.1.4 ESA Catalog

url:	http://services.eoportal.org/portal/order/PrepareOperation.do?serviceId=CatalogueServiceId&serviceName=Products&operation=Search
Status:	Not tested.

1.8.2 Sensor Observeration Service

1.8.2.1 UAH Weather

url:	http://vast.uah.edu:8080/ows/weather
GetCapabilities[?]:	Able to get the capabilities and pull offerings out.
GetObservation:	Able to request the observations based on time span and see the readings as well as map them on the MapView.

1.8.2.2 UAH Plume

url:	http://vast.uah.edu:8080/ows/plumeSos
GetCapabilities[?]:	Able to get the capabilities and pull offerings out.
GetObservation:	Able to request the observations based on time span up to 24 hours and plot the plume points on the MapView. In addition we wrote additional software to take these SOS points and build Shape point and polygons

	out of them.

1.8.2.3 IRIS Chemical / Smoke

url:	http://demo.transducerml.org:8080/ogc/sos
GetCapabilities[?]:	Able to get the capabilities and pull offerings out.
GetObservation:	Successfully executed the GetObservation request that returned an endpoint to a TML stream. The client then connected to that stream and retrieved the phenomenon readings.

1.8.2.4 3ETI Chemical

url:	http://ren.3eti.net:8080/ogc2/GetCapabilities.ogc
GetCapabilities[?]:	Able to get the capabilities and pull offerings out.
GetObservation:	Some inconsistencies between the 3ETI response and UAH. We took a response from 3ETI, cached it, and made a few corrections. From the cache, we were able to retrieve the appropriate observation.

1.8.3 Sensor Planning Service

1.8.3.1 University Muenster (IFGI)

url:	http://mars.uni-muenster.de:8080/52nSPS/SPS
GetCapabilities[?]:	Able to get the capabilities and pull offerings out.
DescribeCollection:	Successfully requested the phenomenon that could be tasked on the camera.
GetFeasibility/Submit:	Successfully formatted request and submitted task.

1.8.3.2 NASA Ames

url:	http://sgqtss.arc.nasa.gov:9519/SpsServlet/SpsServer
Status:	Not tested.

1.8.4 DACS/DRM in Refractions GeoDSS client

This service was outside the scope of our Statement of Work.

url:	unknown
Status:	Unsupported

1.8.5 Sensor Alert Service

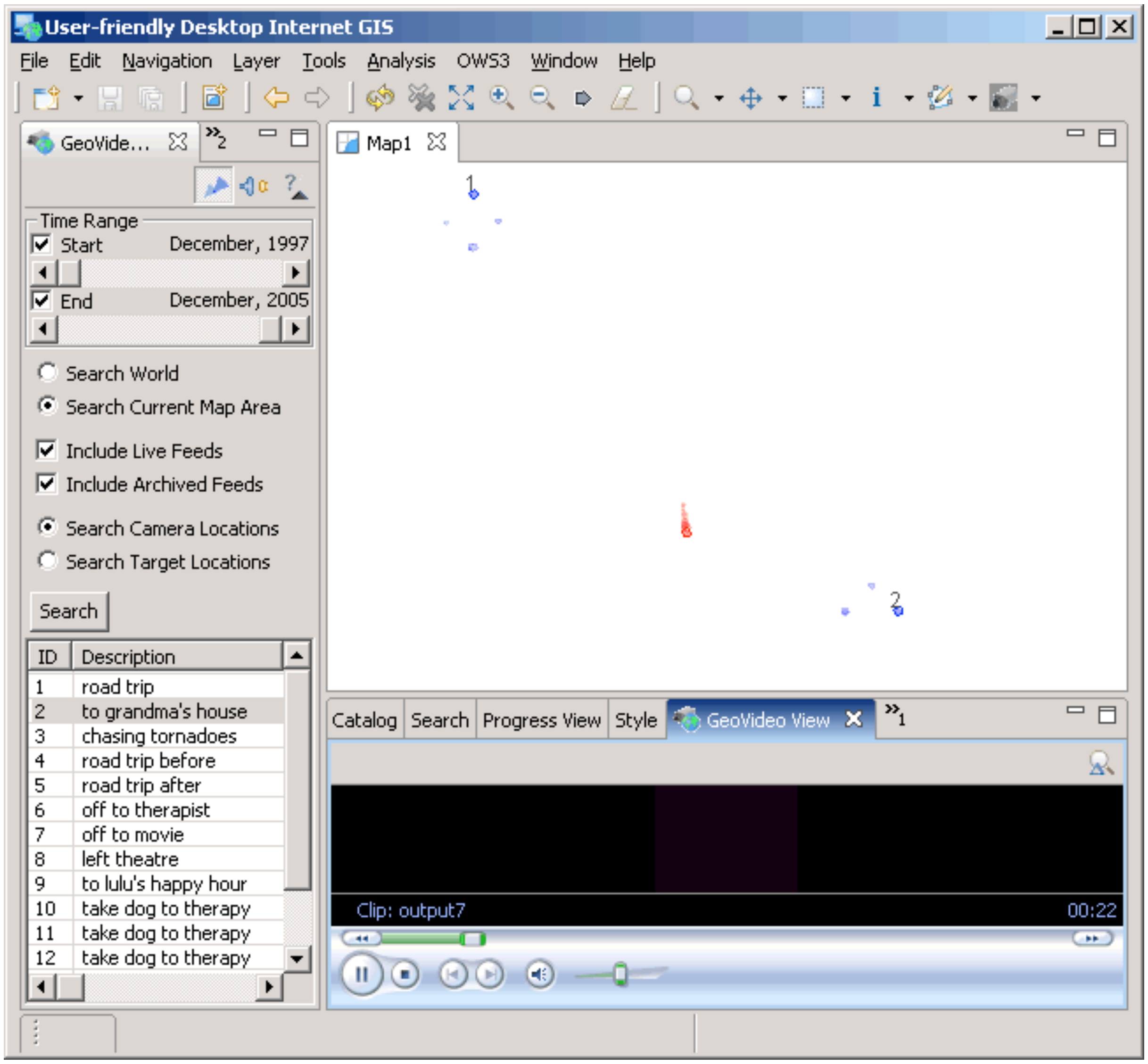
This service was outside the scope of our Statement of Work.

1.8.5.1 3eti

SAS:	http://ren.3eti.net:8080/sas/pushlet/aex.html
Sensor Alert Generator:	http://ren.3eti.net:8080/sas/ConfigureTmlMessage.jsp
Status:	Not tested.

2 Refractions Research

2.1 Intergraph GVS


User-friendly Desktop Internet GIS
File Edit Navigation Layer Tools Analysis OWS3 Window Help
GeoVide...
Map1
1
2
Catalog Search Progress View Style GeoVideo View

Time Range
[✓] Start December, 1997
[✓] End December, 2005

( ) Search World
(●) Search Current Map Area

[✓] Include Live Feeds
[✓] Include Archived Feeds

(●) Search Camera Locations
( ) Search Target Locations

Search

ID	Description
1	road trip
2	to grandma's house
3	chasing tornadoes
4	road trip before
5	road trip after
6	off to therapist
7	off to movie
8	left theatre
9	to lulu's happy hour
10	take dog to therapy
11	take dog to therapy
12	take dog to therapy

Clip: output7 00:22

url:	http://www.geovideoservice-dev.com/GVSBeta/service.asmx?WSDL
date:	9/29
WSDL:	Able to connect to service as described by WSDL
Feed Search:	supported for simple searches
Feed Subscribe:	supported, demoed with sample data

Feed in Area:	supported, demoed with sample data

Notes:

- Note that Windows Media Player doesn't allow capturing the video from the embedded player.

2.2 FPS (Galdos)

page:	GaldosFPSImplementation
url:	http://ows3.demo.galdosinc.com/fps/http?request=GetCapabilities&service=FPS
Capabilities:	Success
GetPortrayal?:	Unable to retrieve image.
Notes:	Interaction was previously fully supported. Recent changes to the service rendered the client unable to request an image successfully.

2.3 FPS (ObjectFX)

For the final TIW we waiting on the availablility of SE documents to complete the workflow.

Service:	http://demo.objectfx.com/OWS3/fps?Request=GetCapabilities&service=fps&version=0.0.30
Capabilities:	Able to parse above capabilities document
GetPortrayal?:	success against CubeWerx? (airports) and Intergraph (schools) WFS using MIL2525B

Reference Requests: * GET Example

- Post Example
- Post Example
- SE Document

2.4 DAS (Ingr)

DAS:	http://zx10.ingr.com/DAS/request.asp
Status:	Server error when loading: Error Type: msxml3.dll (0x80072EFD)

2.5 DAS (CAST)

DAS:	unknown
Status:	Untested

2.6 Data

Drag and Drop of WMS and WFS urls is supported. WFS 1.0 supported, WFS 1.1 unsupported, we will make use of FPS to display this content. We do not currently make use of WMS SLD Post.

2.6.1 WFS-TestData

Intergraph

WFS:	http://regis.intergraph.com/wfs/casil/request.asp?SERVICE=WFS&VERSION=1.0.0&REQUEST=GetCapabilities
Status:	Read-only access tested successfully, both with client and through FPS service.

Galdos

WFS:	unknown
Status:	Untested, no URL given for testing.

Refractions GeoServer[?]

WMS:	http://www.refractions.net:8082/geoserver/wfs/GetCapabilities
Status:	Success

2.6.2 Web Map Server

Public WMS urls used in demo:

Terra Server:

WMS:	http://terraservice.net/ogccapabilities.ashx?version=1.1.1&request=GetCapabilities&service=wms
Status:	Blank image returned.

NASA OnEarth[?]

WMS:	http://onearth.jpl.nasa.gov/wms.cgi?request=GetCapabilities&service=wms (used by GeoTango[?])
Status:	Success.

Nasa Ames

WMS:	http://sggate.arc.nasa.gov:9518/cgi-bin/casil-wms
Status:	Success.

NASA Ames

WMS:	http://sggate.arc.nasa.gov:9518/cgi-bin/casil-wms?request=GetCapabilities&service=wms
Status:	Success.

CubeWerx[?] WMS

WMS:	http://demo.cubewerx.com/demo/cubeserv/cubeserv.cgi?service=WFS&datastore=Foundation&request=GetCapabilities
Status:	Success.

Refractions WMS

WMS:	http://mapserver.refractions.net/cgi-bin/wms_casil
Status:	Success.

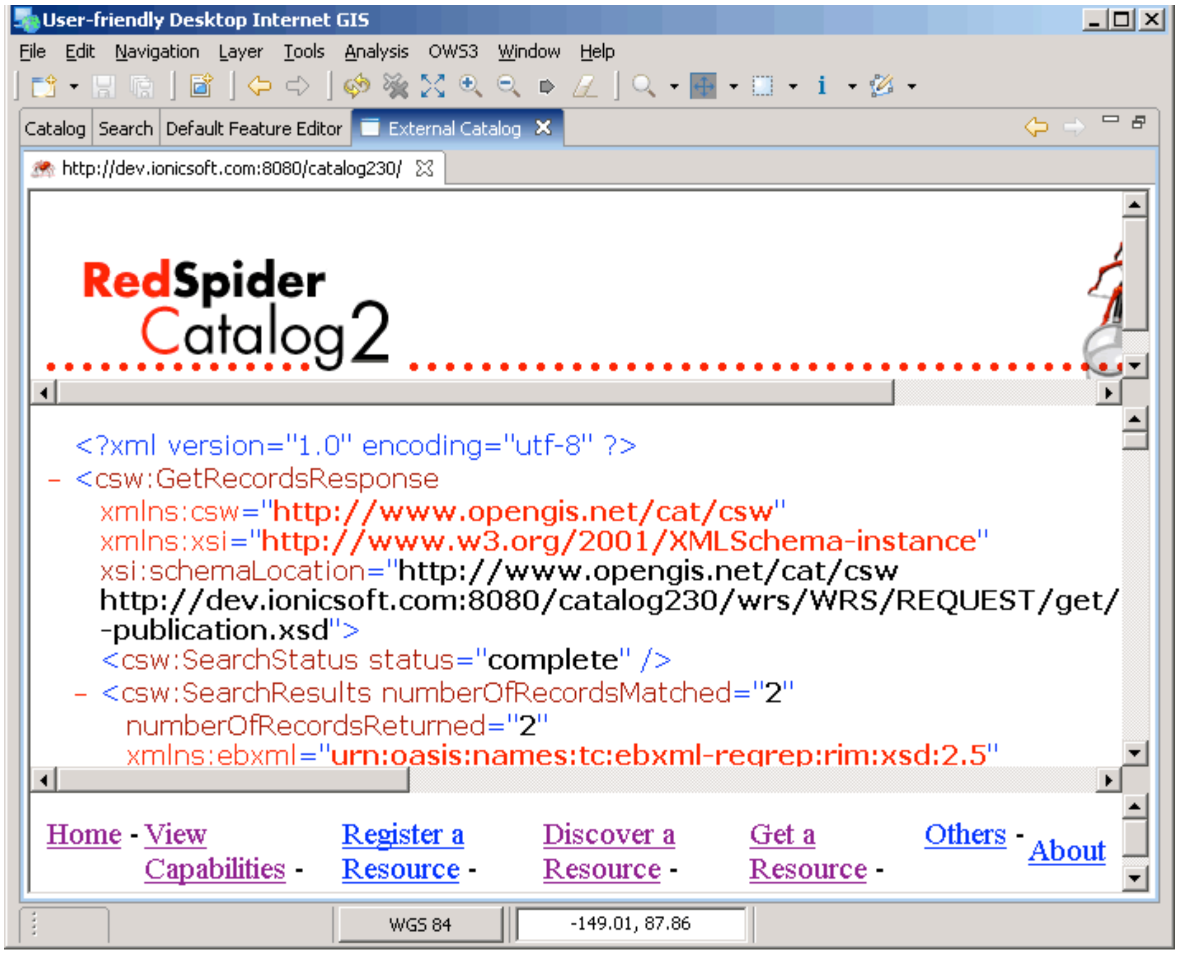
Note for intergration with SE document work WMS would need to be changed to allow for a schemaURL. Several workarounds are known for servers such as Galdos & GeoServer.

2.7 Other Services

The following are out of scope, but limited interaction may prove possible.

2.7.1.1 Ionic Catalog

url:	http://dev.ionicsoft.com:8080/catalog230
Status:	Successfully embedded inside a browser window in Refractions GeoDSS client. Results of queries returned in unsupported format.


User-friendly Desktop Internet GIS
File Edit Navigation Layer Tools Analysis OWS3 Window Help
Catalog Search Default Feature Editor External Catalog
http://dev.ionicsoft.com:8080/catalog230/
RedSpider Catalog2
Home - View Capabilities - Register a Resource - Discover a Resource - Get a Resource - Others - About
WGS 84
-149.01, 87.86

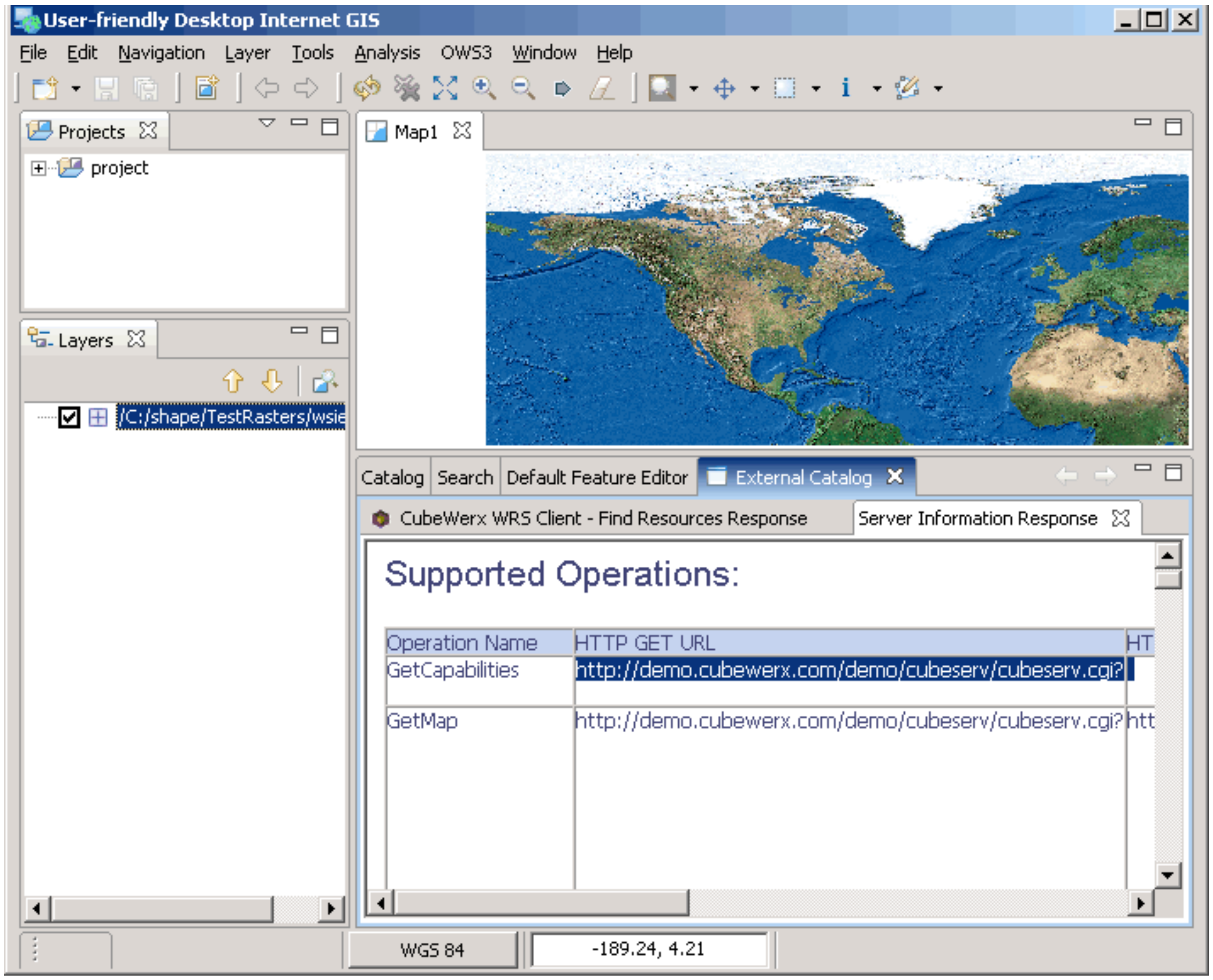
```
<?xml version="1.0" encoding="utf-8" ?>
- <csw:GetRecordsResponse
    xmlns:csw="http://www.opengis.net/cat/csw"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.opengis.net/cat/csw
    http://dev.ionicsoft.com:8080/catalog230/wrs/WRS/REQUEST/get/
    -publication.xsd">
    <csw:SearchStatus status="complete" />
  - <csw:SearchResults numberOfRecordsMatched="2"
      numberOfRecordsReturned="2"
      xmlns:ebxml="urn:oasis:names:tc:ebxml-regrep:rim:xsd:2.5"
```

2.7.1.2 Cubewerx Catalog

Support for catalog is required by our workflow, but is not in our statement of work. We will do a couple one off searches to facilitate the use of FPS.

url:	http://demo.cubewerx.com/ows3/catalog/cwwrs.cgi?request=GetCapabilities&Service=WRS&Version=2.0.0
Status:	n/a - this is a WRS url
url:	http://demo.cubewerx.com/ows3/catalog/Forms/wrs_login.php
SE Lookup:	Static query enables dynamic lookup of SE documents by feature type.
Interaction:	Successfully embedded inside a browser window in Refractions GeoDSS client. Service results returned can be loaded into Refractions GeoDSS client through drag-and-drop, or clicking the link.

We have a conflict between our workflow requirements and our SOW, we will support limited interaction with the following catalogs inorder to aquire SE docuemnts. In the aid of a better demo we have taken to including the web interfaces of the following catalogs.


User-friendly Desktop Internet GIS
File Edit Navigation Layer Tools Analysis OWS3 Window Help
Projects
project
Layers
/C:/shape/TestRasters/wsie
Map1
Catalog Search Default Feature Editor External Catalog
CubeWerx WRS Client - Find Resources Response
Server Information Response
WGS 84
-189.24, 4.21

Supported Operations:

Operation Name	HTTP GET URL	HT
GetCapabilities	http://demo.cubewerx.com/demo/cubeserv/cubeserv.cgi?	
GetMap	http://demo.cubewerx.com/demo/cubeserv/cubeserv.cgi?	htt

2.7.1.3 NASA Catalog

url:	http://orion.compusult.net/wes/serviceManagerCSW/csw
Status:	Successfully embedded inside a browser window in Refractions GeoDSS client. Results of queries returned in unsupported format.

2.7.1.4 ESA Catalog

url:	http://services.eoportal.org/portal/order/PrepareOperation.do?serviceId=CatalogueServiceId&serviceName=Products&operation=Search
Status:	embedded in Refractions GeoDSS client browser, no search results found for testing

2.7.2 DACS/DRM in Refractions GeoDSS client

22.1.1.1 Metalogic DACS

There is a WMS available that requires both DACS authentication and notice acknowledgement: https://demo.fedroot.com/mapserver/cgi-bin/mapserver?Request=GetCapabilities&SERVICE=WMS&VERSION=1.1.1

Values for authentication:

- Federation: DEMO
- Jurisdiction: METALOGIC
- Jurisdiction URL: https://demo.fedroot.com/metalogic/dacs
- Username: smith
- Password: foozle

If you try to access the above WMS with Refractions GeoDSS client, you should be prompted for authentication. After you authenticate, you should be presented with two licences that you must agree to. After that, you should get the image back.

22.1.1.2 University of Federal Armed Forced, Munich, Security Center

An alternative security arrangement is provided by the UFAFM Security Center. This loads a DRM enabled applet in a local browser, and allows proxy connections. The Security Center must be loaded in a browser outside of the GeoDSS Client, then the

1. Login to Security Center: http://iisdemo.informatik.unibw-muenchen.de/ows3demo/wms-client.cubewerx/. A new user can be created to log in with.
2. Open up Secure Data: http://localhost:2780/?REQUEST=GetCapabilities&SERVICE=WMS&VERSION=1.1.0
3. The layers RESTRNTS:Navteq and SHOPPING:Navteq were used for testing and demonstration

2.7.3 Sensor Alert Service

2.7.3.1 3eti

SAS:	http://ren.3eti.net:8080/sas/pushlet/aex.html
Sensor Alert Generator:	http://ren.3eti.net:8080/sas/ConfigureTmlMessage.jsp
Status:	Successfully embedded generator and alert service into internal browser.

Annex B: OWS Context Document

The OWS Context document represents the progress of an ongoing experiment. The following information was used during the OWS-3 experiment and is not authoritative.

Indeed the following example is incorrect in the use of the layer property **hidden**.

alert.xml

This following xml document was used to provide an operational context during the course of the OWS-3 demonstration.

```
<OWSContext id="geodss.105941146" version="0.0.13"
    xmlns="http://www.opengis.net/oc"
    xmlns:ogc="http://www.opengis.net/ogc"
    xmlns:ows="http://www.opengis.net/ows"
    xmlns:param="http;//www.opengis.net/param"
    xmlns:sld="http://www.opengis.net/sld"
    xmlns:xlink="http://www.w3.org/1999/xlink"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.opengis.net/oc oc_0_0_13.xsd"
    >
  <General>
    <Window height="554" width="640"/>
    <ows:BoundingBox crs="EPSG:4326">
      <ows:LowerCorner>-116.98403345713423
     32.47969273632223</ows:LowerCorner>
      <ows:UpperCorner>-116.14598041168348
     33.44713996832372</ows:UpperCorner>
    </ows:BoundingBox>
    <Title>OWS-3 Base Map</Title>
    <ows:ServiceProvider>
      <ows:ProviderName>OWS-3 GeoDSS Thread</ows:ProviderName>
      <ows:ServiceContact>
        <ows:IndividualName>null</ows:IndividualName>
      </ows:ServiceContact>
    </ows:ServiceProvider>
  </General>
  <ResourceList>
    <Layer hidden="1" queryable="1">
      <Server service="OGC:WMS" title="Microsoft TerraServer Map
     Server" version="1.1.1">
        <OnlineResource method="GET"
     xlink:href="http://terraservice.net/ogccapabilities.ashx"
     xlink:type="simple"/>
      </Server>
      <Name>DOQ</Name>
      <Title>USGS Digital Ortho-Quadrangles</Title>
      <SRS>EPSG:26905</SRS>
```

```
<SRS>EPSG:26906</SRS>
<SRS>EPSG:26910</SRS>
<SRS>EPSG:26911</SRS>
<SRS>EPSG:26912</SRS>
<SRS>EPSG:26913</SRS>
<SRS>EPSG:26914</SRS>
<SRS>EPSG:26915</SRS>
<SRS>EPSG:26916</SRS>
<SRS>EPSG:26917</SRS>
<SRS>EPSG:26918</SRS>
<SRS>EPSG:26919</SRS>
<SRS>EPSG:26920</SRS>
<SRS>EPSG:4326</SRS>
<FormatList>
  <Format current="1">image/jpeg</Format>
  <Format>image/jpeg</Format>
</FormatList>
<StyleList>
  <Style current="1">
    <Name>UTMGrid</Name>
    <Title>UTMGrid</Title>
  </Style>
  <Style>
    <Name>GeoGrid</Name>
    <Title>GeoGrid</Title>
  </Style>
  <Style>
    <Name>UTMGrid_Red</Name>
    <Title>UTMGrid_Red</Title>
  </Style>
  <Style>
    <Name>GeoGrid_Red</Name>
    <Title>GeoGrid_Red</Title>
  </Style>
  <Style>
    <Name>UTMGrid_Green</Name>
    <Title>UTMGrid_Green</Title>
  </Style>
  <Style>
    <Name>GeoGrid_Green</Name>
    <Title>GeoGrid_Green</Title>
  </Style>
  <Style>
    <Name>UTMGrid_Blue</Name>
    <Title>UTMGrid_Blue</Title>
  </Style>
  <Style>
    <Name>GeoGrid_Blue</Name>
    <Title>GeoGrid_Blue</Title>
  </Style>
  <Style>
    <Name>UTMGrid_Cyan</Name>
    <Title>UTMGrid_Cyan</Title>
```

```
        </Style>
        <Style>
          <Name>GeoGrid_Cyan</Name>
          <Title>GeoGrid_Cyan</Title>
        </Style>
        <Style>
          <Name>UTMGrid_Magenta</Name>
          <Title>UTMGrid_Magenta</Title>
        </Style>
        <Style>
          <Name>GeoGrid_Magenta</Name>
          <Title>GeoGrid_Magenta</Title>
        </Style>
        <Style>
          <Name>UTMGrid_White</Name>
          <Title>UTMGrid_White</Title>
        </Style>
        <Style>
          <Name>GeoGrid_White</Name>
          <Title>GeoGrid_White</Title>
        </Style>
        <Style>
          <Name>UTMGrid_Black</Name>
          <Title>UTMGrid_Black</Title>
        </Style>
        <Style>
          <Name>GeoGrid_Black</Name>
          <Title>GeoGrid_Black</Title>
        </Style>
        <Style>
          <Name>UTMGrid_Gray</Name>
          <Title>UTMGrid_Gray</Title>
        </Style>
        <Style>
          <Name>GeoGrid_Gray</Name>
          <Title>GeoGrid_Gray</Title>
        </Style>
      </StyleList>
    </Layer>
    <Layer hidden="0" queryable="1">
      <Server service="OGC:WMS" title="CASIL Base Map"
     version="1.1.1">
        <OnlineResource method="GET"
     xlink:href="http://mapserver.refractions.net/cgi-
     bin/wms_casil?" xlink:type="simple"/>
      </Server>
      <Name>hillshade_bw</Name>
      <Title>Hillshade, Black and White, 30m</Title>
      <SRS>EPSG:26909</SRS>
      <SRS>EPSG:26910</SRS>
      <SRS>EPSG:26911</SRS>
      <SRS>EPSG:4269</SRS>
      <SRS>EPSG:4326</SRS>
      <FormatList>
        <Format current="1">image/gif</Format>
        <Format>image/gif</Format>
        <Format>image/png</Format>
        <Format>image/png; mode=24bit</Format>
        <Format>image/jpeg</Format>
        <Format>image/wbmp</Format>
        <Format>image/tiff</Format>
      </FormatList>
      <StyleList>
      </StyleList>
    </Layer>
    <Layer hidden="1" queryable="1">
      <Server service="OGC:WMS" title="CASIL Base Map"
     version="1.1.1">
        <OnlineResource method="GET"
     xlink:href="http://mapserver.refractions.net/cgi-
     bin/wms_casil?" xlink:type="simple"/>
      </Server>
      <Name>state_highways</Name>
      <Title>State Highways</Title>
      <SRS>EPSG:26909</SRS>
      <SRS>EPSG:26910</SRS>
      <SRS>EPSG:26911</SRS>
      <SRS>EPSG:4269</SRS>
      <SRS>EPSG:4326</SRS>
      <FormatList>
        <Format current="1">image/gif</Format>
        <Format>image/gif</Format>
        <Format>image/png</Format>
        <Format>image/png; mode=24bit</Format>
        <Format>image/jpeg</Format>
        <Format>image/wbmp</Format>
        <Format>image/tiff</Format>
      </FormatList>
      <StyleList>
        <Style current="1">
          <Name>default</Name>
          <Title>default</Title>
        </Style>
      </StyleList>
    </Layer>
    <Layer hidden="1" queryable="1">
      <Server service="OGC:WMS" title="CASIL Base Map"
     version="1.1.1">
        <OnlineResource method="GET"
     xlink:href="http://mapserver.refractions.net/cgi-
     bin/wms_casil?" xlink:type="simple"/>
      </Server>
      <Name>us_highways</Name>
      <Title>US Highways</Title>
      <SRS>EPSG:26909</SRS>
      <SRS>EPSG:26910</SRS>
      <SRS>EPSG:26911</SRS>
      <SRS>EPSG:4269</SRS>
      <SRS>EPSG:4326</SRS>
      <FormatList>
        <Format current="1">image/gif</Format>
        <Format>image/gif</Format>
        <Format>image/png</Format>
        <Format>image/png; mode=24bit</Format>
        <Format>image/jpeg</Format>
        <Format>image/wbmp</Format>
        <Format>image/tiff</Format>
      </FormatList>
      <StyleList>
        <Style current="1">
          <Name>default</Name>
          <Title>default</Title>
        </Style>
      </StyleList>
    </Layer>
    <Layer hidden="0" queryable="1">
      <Server service="OGC:WMS" title="NASA Ames ECOSAT CaSIL WMS
     Server" version="1.1.1">
        <OnlineResource method="GET"
     xlink:href="http://sggate.arc.nasa.gov:9518/cgi-bin/casil-
     wms?" xlink:type="simple"/>
      </Server>
      <Name>local_roads</Name>
      <Title>local_roads</Title>
      <SRS>EPSG:26741</SRS>
      <SRS>EPSG:26742</SRS>
      <SRS>EPSG:26743</SRS>
      <SRS>EPSG:26744</SRS>
      <SRS>EPSG:26745</SRS>
      <SRS>EPSG:26746</SRS>
      <SRS>EPSG:26747</SRS>
      <SRS>EPSG:26910</SRS>
      <SRS>EPSG:26911</SRS>
      <SRS>EPSG:26941</SRS>
      <SRS>EPSG:26942</SRS>
      <SRS>EPSG:26943</SRS>
      <SRS>EPSG:26944</SRS>
      <SRS>EPSG:26945</SRS>
      <SRS>EPSG:26946</SRS>
      <SRS>EPSG:4326</SRS>
      <FormatList>
        <Format current="1">image/png</Format>
        <Format>image/png</Format>
        <Format>image/jpeg</Format>
        <Format>image/gif</Format>
        <Format>image/png; mode=24bit</Format>
        <Format>image/wbmp</Format>
```

```
        <Format>image/tiff</Format>
      </FormatList>
      <StyleList>
      </StyleList>
    </Layer>
    <Layer hidden="1" queryable="1">
      <Server service="OGC:WMS" title="CASIL Base Map"
     version="1.1.1">
        <OnlineResource method="GET"
     xlink:href="http://mapserver.refractions.net/cgi-
     bin/wms_casil?" xlink:type="simple"/>
      </Server>
      <Name>gnis_names</Name>
      <Title>Place Names</Title>
      <SRS>EPSG:26909</SRS>
      <SRS>EPSG:26910</SRS>
      <SRS>EPSG:26911</SRS>
      <SRS>EPSG:4269</SRS>
      <SRS>EPSG:4326</SRS>
      <FormatList>
        <Format current="1">image/gif</Format>
        <Format>image/gif</Format>
        <Format>image/png</Format>
        <Format>image/png; mode=24bit</Format>
        <Format>image/jpeg</Format>
        <Format>image/wbmp</Format>
        <Format>image/tiff</Format>
      </FormatList>
      <StyleList>
      </StyleList>
    </Layer>
  </ResourceList>
</OWSContext>
```