Open GIS Consortium Inc.

Date: 2003-01-18

Reference number of this OpenGIS® document: **OGC 03-025**

Supersedes document OGC 02-056r1

Version: 0.3

Category: OpenGIS® Discussion Paper

Editor: Joshua Lieberman, Syncline Inc.

OpenGIS® Web Services Architecture

Warning

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

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

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

File name: 03-025.doc

Contents

i. Preface iv

ii. Submitting organizations iv

iii. Document Contributor Contact Points iv

iv. Revision history v

v. Changes to the OpenGIS® Abstract Specification v

vi. Future Work v

Foreword vi

Introduction vii

1 Scope 1

2 Conformance 1

3 Normative references 1

4 Terms and definitions 3

5 Conventions 4
5.1 Normative Verbs 4
5.2 Abbreviated Terms 4
5.3 UML Notation 5
5.4 XML, XML Schema, and XML Namespaces 6
5.5 Xpath Notation 6

6 Role and Definition of Common Architecture 7
6.1 Open Distributed Processing 7
6.2 Service Oriented Architecture 9
6.3 Service Trading (Publish – Find –Bind) 9
6.4 Service Self-description 12
6.5 Service Stack 12
6.6 Service Classification and Semantics 13
6.7 Architecture-driven Design Guidelines 14

7 Enterprise Viewpoint 16
7.1 The OpenGIS Web Service Framework in the Enterprise 16
7.2 Components of the OpenGIS Web Services Framework 18
7.3 High-level use cases 19

8 Information Viewpoint 19

9 Computational Viewpoint 20
9.1 Elements of the Viewpoint 20

9.2 Interfaces 20
9.3 Interface Hierarchy 21
9.3.1 BasicRegistry Package....22
9.3.2 RepositoryAccess interface 22
9.3.3 Query Package 23
9.3.4 Presentation Package....23
9.3.5 Repository Update Package 23
9.3.6 Task Package....23
9.3.7 Transform Package....23
9.3.8 Interaction Package 23
9.4 Information Flows....23
9.5 Persistence and Statefulness....24

10 Service Definitions 24
10.1 Mapping....24
10.2 Object Handling....24
10.3 Registry_Catalog....24
10.4 Feature Handling 24
10.5 Image Handling....24
10.6 Sensor Web Enablement 25
10.7 Client....25

11 Service Combinations 25
11.1 Overview....25
11.2 Service Aggregation....26
11.3 Service Composition 26
11.4 Service Chaining 26

12 Future Areas of Work 27
12.1 Semantic Processing....27
12.2 Security Frameworks....28
12.3 Commerce Frameworks....28
12.4 Service Chaining 28

Annex A (informative) OWS1.2 Use Cases....29

Annex B (informative) Interface Definitions 35

Annex C (informative) Service Type Descriptions....42

Bibliography....50

i. Preface

This document is an Interoperability Program Report from the OGC Open Web Services (OWS1.2) Testbed. It specifies and discusses a common architectural framework for OGC Web Services. Included here is an overview of the framework itself and a presentation of the framework from several viewpoints, as well as a discussion of how present and future service types and instances can be developed from it.

ii. Submitting organizations

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

Syncline Inc.

iii. Document Contributor Contact Points

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

CONTACT	COMPANY	ADDRESS	PHONE/FAX	EMAIL
Joshua Lieberman	Syncline Inc.	373 Washington Street, Boston, MA 02108	Tel: 617-603-2209 Fax: 617-986-1001	jlieberman@syncline.com
Saul Farber	Syncline Inc.	373 Washington Street, Boston, MA 02108	Tel: 617-603-2248 Fax: 617-986-1001	saul@syncline.com
Jeff de la Beaujardiere	NASA	NASA Goddard Space Flight Center Code 933 Greenbelt MD 20771 USA	+1 301 286 1569	delabeau@iniki.gsfc.nasa.gov
Peter Vretanos	Cubewerx	200 Rue Montcalm, Suite R-13 Hull, Quebec J8Y 3B5 CANADA	+1 416 701 1985	pvretano@cubewerx.com
Stephane Fellah	PCI Geomatics	490 St. Joseph Blvd. Hull, Quebec J8Y 3Y7	+1 (819) 770-0022 ext 223	fellah@pcigeomatics.com

Richard Martell	Galdos Systems, Inc.	1155 West Pender St Vancouver, BC V6E 2P4	+1 (604) 484-2750	rmartell@galdosinc.com
Jérôme Sonnet	Ionic Software s.a.			jerome.sonnet@ionicsoft.com
George Percivall	NASA/ GST, Inc.	Goddard Space Flight Center, Greenbelt MD	+ 1 (301) 286-4073	percivall@gsfc.nasa.gov
John Davidson	Image Matters LLC	105 S. King St. Leesburg, VA 20175	+1 (703) 669-5510	johnd@imagemattersllc.com

iv. Revision history

Date	Release	Author	Paragraph modified	Description
2002-08-25	0.2	Joshua Lieberman	First Draft	First Draft of Common Architecture DIPR for OWS 1.2
2003-01-20	0.3	Joshua Lieberman	First IPR version	Revisions for TC presentation

v. Changes to the OpenGIS® Abstract Specification

The OpenGIS® Abstract Specification does not require changes to accommodate the technical contents of this document. Attention is drawn, however, to the possibility that future revisions of this document may require revisions to Topic 12 and other AS topics.

vi. Future Work

Improvements in this document are desirable to reflect the experience of those implementing this service architectural framework in their own service types and instances.

Foreword

Attention is drawn to the possibility that some of the elements of this part of OGC 03-025 may be the subject of patent rights. The Open GIS Consortium Inc. shall not be held responsible for identifying any or all such patent rights.

This OWS 1.2 edition replaces in part the OWS 1.1 edition of a common architecture document (OGC 02-022), OGC Web Services Initiative – Reference Architecture for Phase 1 Testbed. It also supersedes (OGC 02-056) OWS 1.2 Common Architecture: Overview and Computational Viewpoint

OGC 03-025 consists of the following parts, under the general title: OpenGIS Web Services Architecture

- *Part 1: Role and Definition of Common Architecture*
- *Part 2: Enterprise Viewpoint*
- *Part 3: Information Viewpoint*
- *Part 4: Computational Viewpoint*
- *Part 5: Service Definitions*
- *Part 6: Service Combinations*
- *Part 7: Engineering Viewpoint*
- *Part 8: Technology Viewpoint*
- *Part 9: Future Areas of Work*

Introduction

One of the main activities of the OGC Web Services testbed is captured in an Interoperability Program Report (IPR) that describes a common architectural framework for web-based geospatial services. This framework specifies the scope, objectives and behaviour of a system and its functional components which are common to all such services and extensible for specific services and service types. To the extent that this is also a reference architecture in the spirit of ISO 19119, it should be independent of particular technology choices. The framework presented in this report does, however, incorporate technology choices which have been defined for, or have evolved out of implementation experiences in the OWS 1.2 testbed.

The architectural framework presented in this Interoperability Program Report addresses the design requirements for enabling interoperation between instances of "Web Services" deployed using a large class of OpenGIS Web Services interface specifications. Based on the requirements and corresponding Use Cases in OGC 02-057 Architecture Requirements DIPR, this document specifies mechanisms, rules, and patterns for:

- Defining web service types
- Defining web service implementations
- Specifying distributed computing interactions
- Establishing distributed computing workflow
- Publishing shared semantics (e.g. taxonomies)
- Implementing specific distributed computing technologies in order to balance functionality and interoperability.

Unless otherwise noted, the reference architecture described in this report is consistent with the terms and concepts put forth in ISO 19119 (Geographic Information – Services) and ISO 10746 (Reference Model for Open Distributed Processing, RM-ODP). Following the RM-ODP international standard for architecting open distributed processing, this document describes the OWS1 reference architecture in terms of the viewpoints described in Table 1.1.

RM-ODP Viewpoint	Areas of concern
Enterprise viewpoint	Purpose and scope Policies Responsibilities
Information viewpoint (addressed in OGC 03-026)	Information models Schemas Semantics of information
Computational viewpoint	Functional decomposition Interfaces

	Operations Binding rules
Engineering viewpoint (also addressed in the OGC Messaging Framework DIPR, etc)	Infrastructure required to support distribution
Technology viewpoint (also addressed in the SOAP and UDDI Experiment DIPR's)	Choice and suitability of technology to support system distribution

OpenGIS® Web Services Architecture

1 Scope

This OpenGIS® Interoperability Program Report (IPR) is a partial description of OGC Web Services. It is a specification and description of a common architectural framework for the design and implementation of Open Distributed Processing applications based on a Web Services platform. This IPR provides an architectural overview, information and engineering viewpoints of the framework, and specific service examples in UML, XML Schema, and XML vocabulary clauses. Other clauses describe architectural extensions for specific processing requirements.

2 Conformance

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

3 Normative references

The following normative documents contain provisions that, through reference in this text, constitute provisions of this specification. 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.

Beech, David, Maloney, Murry, Mendelson, Noah, Thompson, Harry S., "XML Schema Part 1: Structures", May 2001, W3C Recommendation, http://www.w3c.org/TR/xmlschema-1.

Bray, Hollander, Layman, eds., "Namespaces In XML", January 1999, W3C Recommendation, http://www.w3.org/TR/2000/REC-xml-names.

Clark, James, DeRose, Steve, "XML Path Language (XPATH), Version 1.0", November 1999, W3C Recommendation, http://www.w3c.org/TR/XPath.

CGI, *The Common Gateway Interface,* National Center for Supercomputing Applications, <http://hoohoo.ncsa.uiuc.edu/cgi/>

Cox, S., Cuthbert, A., Lake, R., and Martell, R. (eds.), "OpenGIS Implementation Specification #02-009: OpenGIS® Geography Markup Language (GML) Implementation Specification, version 2.1.1", April 2002

EPSG, *European Petroleum Survey Group Geodesy Parameters*, Lott, R., Ravanas, B., Cain, J., Girbig, J.-P., and Nicolai, R., eds., <http://www.epsg.org/>

FGDC-STD-001-1988, *Content Standard for Digital Geospatial Metadata (version 2),* US Federal Geographic Data Committee, <http://www.fgdc.org/metadata/contstan.html>

IETF RFC 2045 (November 1996), *Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies*, Freed, N. and Borenstein N., eds., <http://www.ietf.org/rfc/rfc2045.txt>

IETF RFC 2119 (March 1997), *Key words for use in RFCs to Indicate Requirement Levels*, Bradner, S., ed., <ftp://ftp.isi.edu/in-notes/rfc2119.txt>.

IETF RFC 2616 (June 1999), *Hypertext Transfer Protocol – HTTP/1.1*, Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., and Berners-Lee, T., eds., <http://www.ietf.org/rfc/rfc2616.txt>

IETF RFC 2396 (August 1998), *Uniform Resource Identifiers (URI): Generic Syntax*, Berners-Lee, T., Fielding, N., and Masinter, L., eds., <http://www.ietf.org/rfc/rfc2396.txt>

ISO 8601:1988(E), *Data elements and interchange formats - Information interchange - Representation of dates and times*.

ISO/IEC 10746-3:1996. *Open Distributed Processing – Reference Model: Architecture*. Available [online]: <http://www.iso.ch/iso/en/ittf/PubliclyAvailableStandards/s020697e.zip>.

ISO/IEC 13235-1:1998. *Open Distributed Processing – Trading Function: Specification*.

ISO/IEC FDIS 14769:2000. *Open Distributed Processing – Type Repository Function.* Available [Online]: <http://www.cs.helsinki.fi/Lea.Kutvonen/ODP/dokumentit/FDIS14769.pdf>.

ISO 19115, *Geographic information — Metadata*

OGC AS 12 (January 2002), *The OpenGIS Abstract Specification Topic 12: OpenGIS Service Architecture (Version 4.3)*, Percival, G. (ed.), <http://www.opengis.org/techno/specs.htm>

UCUM, *Unified Code for Units of Measure,* Schadow, G. and McDonald, C. J. (eds.), http://aurora.rg.iupui.edu/~schadow/units/UCUM/

Vretanos, Panagiotis (ed.), "OpenGIS Implementation Specification #01-067: Filter Encoding Implementation Specification", May 2001

XML 1.0 (October 2000), *Extensible Markup Language (XML) 1.0* (*2nd edition)*, World Wide Web Consortium Recommendation, Bray, T., Paoli, J., Sperberg-McQueen, C.M., and Maler, E., eds., <http://www.w3.org/TR/2000/REC-xml>

4 Terms and definitions

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

operation
specification of a transformation or query that an object may be called to execute [OGC AS 12]

interface
named set of **operations** that characterize the behavior of an entity [OGC AS 12]

service
distinct part of the functionality that is provided by an entity through **interfaces** [OGC AS 12]

service instance
server
actual implementation of a **service** or conceptual role as recipient of an operation request.

client
software component that can invoke an **operation** from a **server** or conceptual role as originator of an operation request.

request
invocation of a **server operation** by a **client**

binding
specific syntax and parameter values used by a **client** to invoke a specific **server operation**

response
result of an **operation** returned from a **server** to a **client**

map
pictorial representation or portrayal of geographic data

spatial reference system (SRS or CRS)
a projected or geographic coordinate reference system

service capabilities
service-level metadata describing the **types**, **operations**, **content,** and **bindings** available at a **service instance**. Organization, classification, and presentation of those entities may also be conveyed by the capabilities information.

capabilities schema
XML schema which prescribes and constrains the syntax and vocabulary for the expression of service capabilities in XML.

capabilities XML
specific instance of service-level metadata describing a **service instance**.

5 Conventions

5.1 Normative Verbs

In the sections labeled as normative, the key words "**required**", "**shall**", "**shall not**", "**should**", "**should not**", "**recommended**", "**may**", and "**optional**" in this document are to be interpreted as described in [IETF RFC 2119].

5.2 Abbreviated Terms

DCP	Distributed Computing Platform
DTD	Document Type Definition
HTTP	Hypertext Transfer Protocol
IETF	Internet Engineering Task Force
MIME	Multipurpose Internet Mail Extensions
OGC	Open GIS Consortium
OWS	OGC Web Service
URL	Uniform Resource Locator
WMS	Web Map Service
XML	Extensible Markup Language
API	Application Program Interface
COM	Component Object Model
CORBA	Common Object Request Broker Architecture

COTS	Commercial Off The Shelf
DCE	Distributed Computing Environment
DCP	Distributed Computing Platform
DCOM	Distributed Component Object Model
IDL	Interface Definition Language
ISO	International Organization for Standardization
OGC	Open GIS Consortium
UML	Unified Modeling Language
1D	One Dimensional
2D	Two Dimensional
3D	Three Dimensional
WSDL	Web Services Definition Language

5.3 UML Notation

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

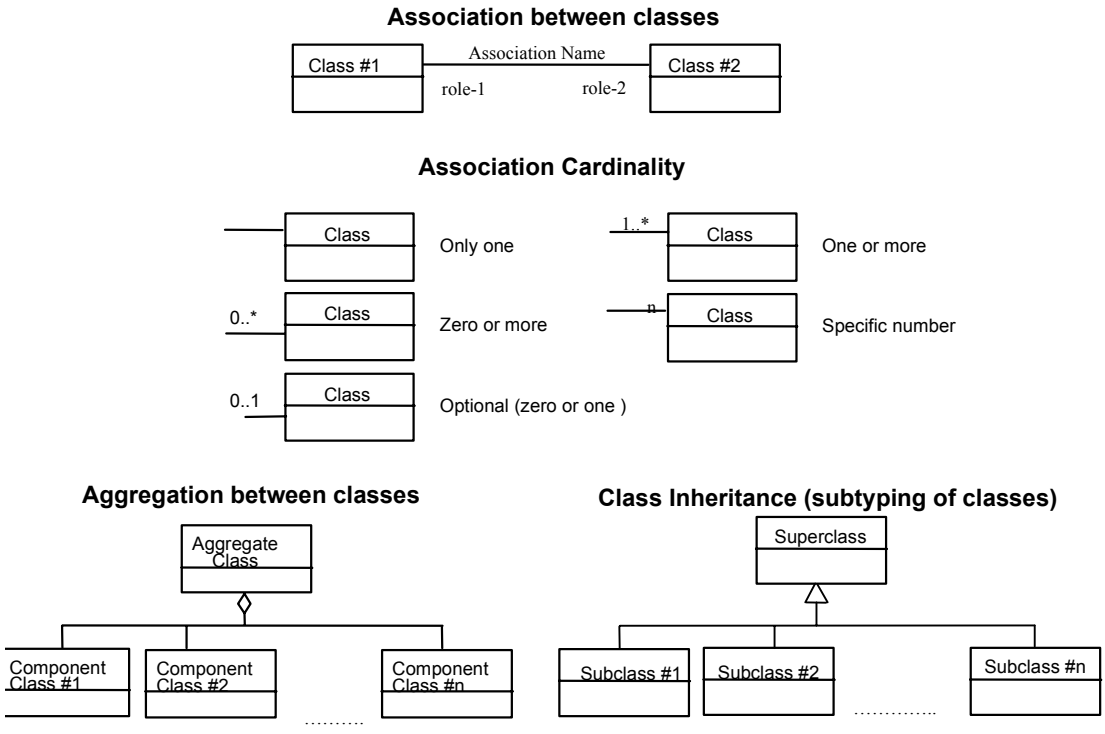


Figure 1 — UML notation

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

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

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

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

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

a) CharacterString – A sequence of characters

b) Integer – An integer number

c) Double – A double precision floating point number

d) Float – A single precision floating point number

5.4 XML, XML Schema, and XML Namespaces

XML Schema 1.0 is used in this document to define the syntax and vocabulary of XML documents. It is not assumed that XML documents will be fully validated against their respective schemas for normal service invocation. It is assumed, however, that XML parsing and validation will follow normal XML syntax rules, for example regarding XML Namespaces and their definition within XML documents. Specific namespace URI's are suggested and their standardization is encouraged. Specific namespace prefixes area used, but their standardization is not suggested in order to maintain maximum flexibility in aggregating XML from different namespaces.

5.5 Xpath Notation

To refer to specific portions of an XML document, this specification uses a subset of the syntax defined by [W3C XPath]. In particular:

`A/B` refers to all elements <B> which are an immediate child of any element <A> in the document.

`/A` refers to the root element <A> of the document.

`C/@D` refers to the value of the attribute D of any element <C> in the document.

6 Role and Definition of Common Architecture

OpenGIS Web Services (OWS) are individual components of dynamic geospatial computing applications; they are also parts of an overall paradigm for building solutions to geospatial “problems”. This paradigm - the Spatial Web - imposes both conceptual and implementation constraints on how OWS works. Conceptual constraints include service orientation, n-tier distribution capability, self-description, and stateless operation; these constraints generally address functionality.

Implementation constraints include use of common XML encodings, HTTP transports, tightly defined interface syntax, specific information models for service descriptions and other metadata; these constraints generally address interoperability. Taken together, these constraints form a common basis or architectural framework to guide the successful and interoperable implementation of OWS instances. From a practical viewpoint, the more of these common constraints can be satisfied once for all services, the less work it will be either to implement existing service types or to define new ones. As with any functional web, this should lead more quickly to the efficiencies of both scale and diversity (many nodes of many different capabilities) which give the Web much of its value.The OWS Common Architecture defines a framework of guiding concepts, terminology, fundamental patterns and organizing principles for implementation and deployment. The OWS Common Architecture establishes common interfaces, exchange protocols, and services that can be utilized by any application. OpenGIS® Implementation Specifications provide guidance to application developers on how to build their applications to comply with the OWS Architecture. OpenGIS® Services are implementations of services that conform to OpenGIS® Implementation Specifications. Compliant applications, called OpenGIS® Applications, can then "plug into" the framework to join the enterprise operational environment. This loosely coupled approach to enterprise development results in very agile systems.

By building applications to common interfaces, each application can be built without build-time or run-time dependencies on other applications. New applications and services can be added, modified, or replaced without impacting any other applications. In addition, operational workflows can be changed dynamically, allowing rapid response to crises conditions.

The following subclauses present useful and recognized patterns for OpenGIS Web Services and their interactions.

6.1 Open Distributed Processing

The concept of distributing computing functions across a network in a dynamic fashion has been addressed by the **Reference Model for Open Distributed Processing** (RM-ODP), an international standard for architecting open, distributed processing systems.

RM-ODP provides both a conceptual framework for such systems, and implementation patterns or rules.

The RM-ODP model identifies the top priorities for architectural specifications and provides a minimal set of requirements—plus an object model—to ensure system integrity. Five standard *viewpoints* are defined; the viewpoints address different aspects of the system and enable the 'separation of concerns':

- **Enterprise** viewpoint: articulates a "business model" that should be understandable by all stakeholders; focuses on purpose, operational objectives, policies, enterprise objects, etc.
- **Information** viewpoint: focuses on information content and system behaviour (i.e. data models, semantics, schemas)
- **Computational** viewpoint: captures component and interface details without regard to distribution
- **Engineering** viewpoint: exposes the distributed nature of the system and provides standard definitions to describe engineering constraints
- **Technology** viewpoint: describes where to apply the technologies/products of choice and allows for conformance testing against the architectural specification

There are a number of documents associated with the OWS1.2 initiative which express various components of these viewpoints. The present document serves both as an overview to those documents and to fill in those common aspects of architecture that are not covered in more specific discussions.

The Enterprise Viewpoint is addressed by a combination of the OGC mission and the business requirements articulated in Section 7 and Annex A as well as the other individual requirements and scope documents developed as part of OWS1.2.

The Information Viewpoint is addressed specifically in OGC 03-026 Service Information Model IPR, OGC 03-024 Registry IPR, and other information model / metadata documents.

The Computational Viewpoint forms the core of the present report, by providing an object model for the implementation-independent definition of OWS interfaces.

The Engineering Viewpoint is also covered in this report, as well as by other documents. It focuses on the requirements for mechanisms by which interfaces are knit together into a processing network such as the Spatial Web. These include encodings / serializations, messaging frameworks, transactions, exception-handling, persistence, etc. These mechanisms are described in as platform-neutral a manner as possible.

The Technology Viewpoint is distinguished from the Engineering Viewpoint only to the extent that the above engineering mechanisms can be defined separately from technology. A common architectural framework should in principle be defined in a technology – neutral fashion. In practice, such a framework can only enhance interoperability by

recommending the best presently available common technologies for achieving this purpose, so discussions of specific technology do enter necessarily into portions of the present document.

6.2 Service Oriented Architecture

The Service Oriented Architecture (SOA) pattern provides the fundamental roles of service provider and service consumer within a distributed computing network. This pattern emphasizes that computing tasks or solutions may be devised from performance of an ad hoc configuration of individual services, about which only the types (e.g interfaces and capabilities) and holdings (e.g. content) need be known. It focuses component definition towards providing or consuming a defined service. SOA also focuses the basis for interaction between components on those same defined services, namely in the form of service requests, service responses, or service exceptions.

It is this paradigm which essentially underpins the Computational and Information Viewpoints of OGC Web Services presented in this and associated reports.

6.3 Service Trading (Publish – Find –Bind)

Service trading is a fundamental concept that addresses the discovery of available service instances. The ODP Trading function facilitates the offering and the discovery of interfaces which provide services of particular types: a trader implementation records service offers and matches requests for advertised services. Publishing a capability or offering a service is called “export”. Matching a service request against published offers or discovering services is called “import”. This can also be depicted in an equivalent manner as the “Publish – Find – Bind” (PFB) pattern of service interaction.

The Trading function is elaborated in a separate document (ISO/IEC 13235-1) and refined somewhat in the OMG Trading specification, which is technically aligned with the computational view of the ODP trading function. Most importantly, a trader supports dynamic (i.e. run-time) binding between service providers and requesters, since sites and applications are frequently changing in large distributed systems. The fundamental roles and interactions are depicted in Figure 2.[1] The equivalent PFB terminology is shown as well (blue). A trader registers service offers from exporter objects and returns service offers upon request to importer objects according to some criteria.

[1] Many readers will recognize Figure 2: most of the recent web services white papers include similar diagrams that map onto it directly. In many cases ‘find’, ‘bind’, and ‘publish’ substitute for ‘import’, ‘service interaction’, and ‘export’, respectively.

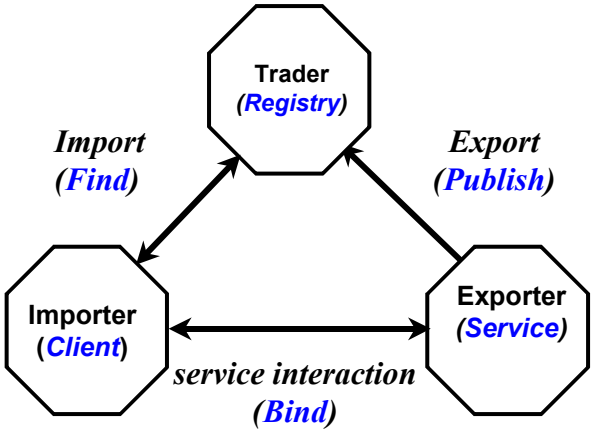


Figure 2 Service trading ([2],[4])

In the RM-ODP framework, there are three fundamental roles:

trader - a role which registers service offers from exporter objects and returns service offers upon request to importer objects according to some criteria.

exporter - a role which registers service offers with the trader object

importer - a role which obtains service offers, satisfying some criteria, from the trader object.

In effect a Trader plays the role of "matchmaker" in a service-based architecture, as suggested by the informal sequence diagram in Figure 3.

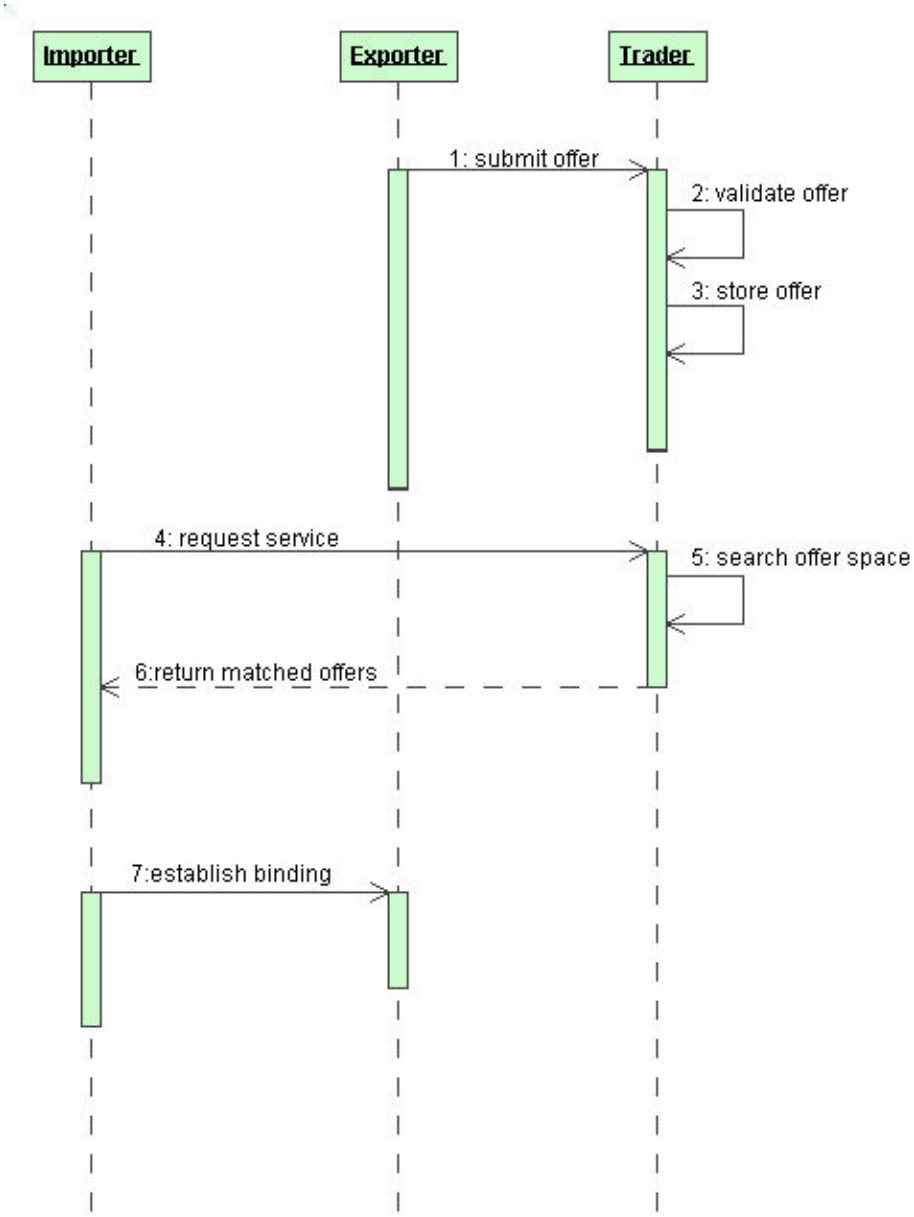


Figure 3. Trading Interactions

To export (i.e. publish a service offer), an object gives the trader a description of a service, including a description of the interface at which that service instance is available. To import (i.e. find suitable service offers), an object asks the trader for a service having certain characteristics. The trader checks against the descriptions of services and responds to the importer with the information required to bind with a service instance. Preferences may be applied to the set of offers matched according to service type, some constraint expression, and various policies. Application of the preferences can determine the order used to return matched offers to the importer.

6.4 Service Self-description

The fully developed Spatial Web of the future will consist of a full array of components and full specialization for each role of the service trader pattern. In the sparse and incomplete present, however, it is necessary for each service to be able to represent itself to potential consumers as if it were a simplified service trader. Even in the case where there exist registry services to function as specialized service traders, the administrative infrastructure rarely exists for these services to be authoritative; it is still advantageous for a service to be able to "publish itself" by providing a single authoritative source of its own service description upon registry request. In essence, each service serves as its own leaf node in a developing distributed or "federated" registry topology.

In light of this requirement that every service also function as a registry, it is necessary to make the requirement as light a burden as possible on service implementers. This generally means defining a minimal or "basic" registry interface which fulfills the leaf node registry function of self-description and no more.

6.5 Service Stack

Having established the "why" pattern of interaction between ODP components (e.g. service provision) and the "who" pattern (e.g. service consumer <-> service provider), there remains the "how" of establishing a meaningful communication between two software instances connected by a wire. One way to represent this might be as a sequence of translations, for example from the service requester to the wire and then in reverse from the wire to the service responder, then back again. A more useful and less repetitive pattern is to group the translations by commonality across different communications, then arrange the groups vertically from most hardware-specific at the bottom, to most implementation-independent at the top. This is also known as a protocol stack and is most familiar in the context of Open Transport. A specific example pertaining to Web Service interoperability is shown in Figure 4 from OGC Document 02-022. Conceptual service interaction layers are shown on the left, while their connections to specific technology are shown on the right.

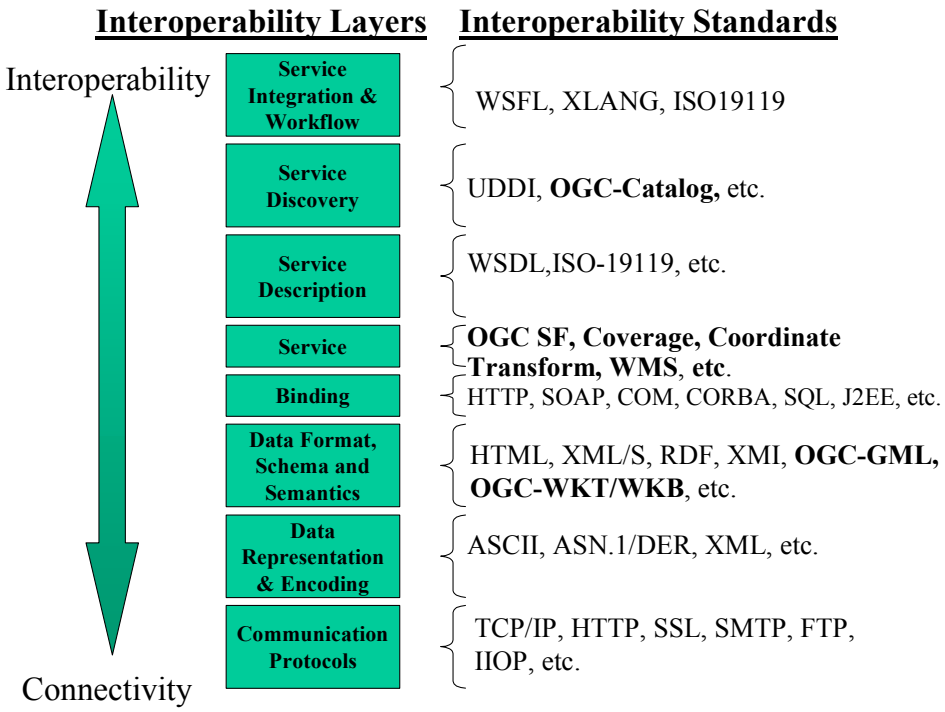


Figure 4. Service Interoperability Stack

6.6 Service Classification and Semantics

At some point, it becomes necessary to step above the level of defining the mechanics of interactions between components, and consider the meaning of such interactions, otherwise known as semantic interoperability. A full consideration of this topic is (still) beyond the scope of this document, indeed of OWS 1.2 in general. Without becoming enmeshed in general computational models and knowledge representation, it is still useful to characterize most human users and virtually all software clients as creatures of habit best able to interact meaningfully with what is familiar. A useful strategy for playing to this strength involves making every interaction between ODP components (human or otherwise) as habitual and familiar as possible. Familiar roles and familiar protocols are one device , as is strong typing (e.g. unfamiliar service instances but representing familiar service types, including capabilities representation).

A more general approach to implementing this strategy involves classification – the art of describing unfamiliar objects such as service providers with familiar terms.

The OWS service taxonomy implements this approach by grouping services that are semantically similar in familiar categories, so as to facilitate browsing and discovery according to already understood (human) or pre-programmed (machine) functionality; this extensible classification scheme is extracted from ISO 19119, Clause 7 of which requires compliant systems to categorize services accordingly. Only the six top-level service domains are mandatory—these constitute a sort of upper-level ontology (Table 1).

Table 1. Top-level OWS1 service categories

Service category	Description
Human Interaction	Services for managing user interfaces, graphics, multimedia, and presenting compound documents.
Information Management	Services for managing the development, manipulation, and storage of metadata, conceptual schemas, and datasets.
Workflow	Services that support specific tasks or work-related activities.
Processing	Services that perform large-scale computations; a processing service does not include capabilities for providing persistent storage of data or transfer of data over networks.
Communication	Services that encode and transfer data across networks.
System Management	Services for managing system components, applications, and networks (including access control).

These categories illustrate that meaning doesn't (necessarily) exist outside of function, as they closely parallel distinct areas of functionality which may be common to many different services in varying degrees. The service model developed below as an aspect of the Computational Viewpoint contains, not coincidentally, top level packages quite similar to the above categories.

6.7 Architecture-driven Design Guidelines

No architecture is truly meaningful or valuable until it can be realized by the process of building something concrete (although not necessarily *of* concrete). A process framework is therefore needed to support and enforce conformant implementation. This clause presents elements of an OWS process framework, including design principles and high-level goals, constraints, assumptions and guidelines. Elements include:

1. Everything is a network resource including clients, services, data content, appliances, and computers.

2. Resources (especially services) have contracts (i.e., resources must have well-defined roles, responsibilities, interfaces, and semantics)

3. Interoperability of services over time is maintained by focusing on commitment to contracts not adherence to static protocols.

4. Design for availability through dynamic discovery of resources:

   a.) Assume networks are unreliable and will fail

   b.) Do not assume given resources are always available in the same location

   c.) Promote "self-healing" through dynamic discovery and fail-over to other resources

   d.) A dynamic distributed application relies on the availability of multiple instances of any given resource type; the more effective the typing framework, the more dynamic and therefore reliable the application can become.

5. Maximize stateless behavior of resources:

   a.) Resources are accessed on a transient basis.

   b.) Persistent state should be maintained solely on the tier that is interested in (responsible for) the specific computational transaction (this may also be termed separation of concerns)

   c.) Minimize dependencies of resources on the network; maximize loose coupling (i.e., minimize hard-coded dependencies between resources such as client to service instances).

   d.) Services should maintain state only within the context which requires it for a particular application (e.g. single request, single transaction of multiple requests with one client, single task sequence of transactions with multiple clients, etc.).

   e.) States which are to be maintained indefinitely (e.g. Web Map client contexts, user access profiles) should be managed as (metadata) resources in their own right rather than as stateful service behavior.

   f.) Clients and services should limit the duration over which they hold resources to minimize chances of losing the resource and maximize reuse.

6. To maximize reuse, assume resources will be deployed and used in different application and deployment contexts over time

   a.) Assume different deployment platforms and network communications protocols (e.g., transactional synchronous, point to point asynchronous messaging, broadcast asynchronous messaging, real time streams, low bandwidth formats, etc) will be required.

7 Enterprise Viewpoint

Note: The enterprise viewpoint into a service architecture concerns itself largely with the context of that architecture: the business models, policies, and organizational structures within which the architecture will be implemented. Another means of expressing the enterprise viewpoint is through use cases and derived requirements for the functionality to be supported through the architecture. In the case of OWS 1.2, those use cases and requirements were initially developed in an architectural requirements document and are now largely contained in individual IPRs for the principal activity areas of the initiative.

7.1 The OpenGIS Web Service Framework in the Enterprise

The OpenGIS Web Services Framework provides a common set of interfaces and encodings that span the functional parts of the enterprise shown in Figure 5 to provide enterprise-wide interoperability. The enterprise depicted is comprised of users and customers, networks and communities of geospatial services and data holdings representing real world features and phenomena. Network resources are published for broader use within and between communites. Applications discover, access and interact with network resources through simple request-response interactions.

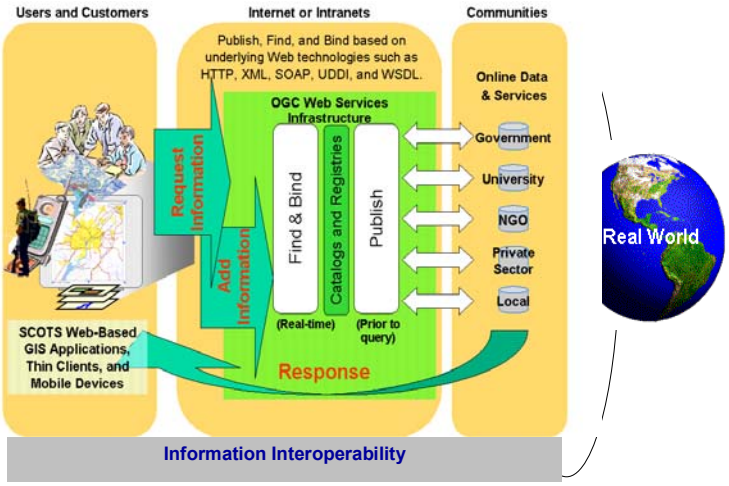


Figure 5. OpenGIS® Web Services in the Enterprise

The OpenGIS Web Services Framework is a platform for next-generation distributed geoprocessing and location systems that:

- Enables future applications to be assembled from multiple, network-enabled geoprocessing and location-services.
- Integrates standard web-services technologies
- Enables information interoperability across communities and resources
- Reduces barriers between real world, information about the real world, and distributed users

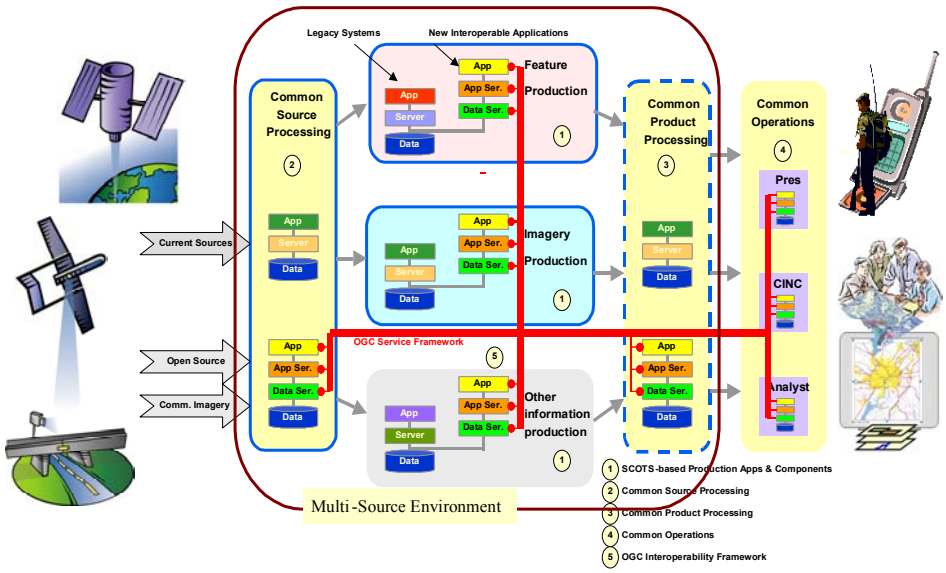


Figure 6 System Concept: Multi-source information operations

Figure 6 depicts the OWS1.2 system concept representing a typical information production environment where the OpenGIS Services Framework is the basis for enterprise-wide interoperability. There are six main functional parts to this production environment:

- Common Source Processing – where all source acquisition, assessment and processing occurs
- Feature Production – where feature production and management occurs
- Imagery Production – where imagery production and management occurs
- Other Information Production – where multi-source processing and specialized information production and management occurs
- Common Product Processing – the main product finishing and distribution capability

- Common Operations – common capabilities for customers to view and exploit deployed products

7.2 Components of the OpenGIS Web Services Framework

The OpenGIS Services Framework (Figure 7) provides the common set of interfaces that spans these functional parts of the enterprise and provides enterprise-wide interoperability.

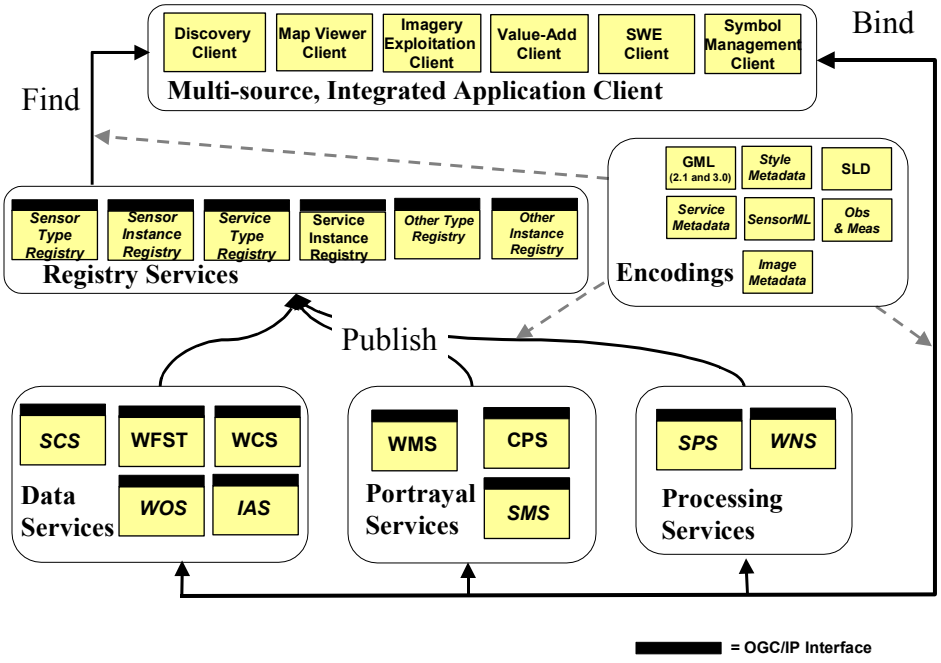


Figure 7. OpenGIS Web Service Framework Components

The elements of the architecture that are the focus of the OWS1.2 testbed are as follows:

- ***Client Services***[2] – the client-side components of client applications that interact with users, and on the server-side interact with *Server-side Client Applications, Application Servers* and *Data Servers*. (Client Services are services of type Geospatial Human Interaction Services as defined in OGC Abstract Specification Topic 12.)

[2] From ISO TC 211: The Client tier consists of the user environment. A client component (we refer to this as a Client Service, given a service perspective) contains the logic that presents information to an external source and obtains input from that source. In most cases the external source is a human end user working at their own computer, although the external source might also be process-oriented. The client logic generally provides menus of options to allow the user to navigate through the different parts of the application, and it manipulates the input and output fields on the display device. Frequently, the presentation component also performs a limited amount of input data validation.

- ***Registry Services*** – provides a common mechanism to classify, register, describe, search, maintain and access information about network resources (data and services). For OWS1.2, Registry Services include Web Registry Service (WRS).

- ***Processing-Workflow Services*** – the foundational application-building-block services that operate on geospatial data and metadata, providing value-add service. For OWS1.2, Processing-Workflow Services include Sensor Planning Service (SPS) and Web Notification Service (WNS).

- ***Portrayal Services*** – Portrayal Services provide specialized capabilities supporting visualization of geospatial information. Portrayal Services are components that, given one or more inputs, produce rendered outputs such as cartographically portrayed maps, perspective views of terrain, annotated images, views of dynamically changing features in space and time, etc.). For OWS1.2, Portrayal Services include Web Map Service (WMS), Coverage Portrayal Service (CPS) and Style Management Service (SMS).

- ***Data Services*** – the foundational service building blocks that serve data, specifically geospatial data. For OWS1.2, Data Services include Web Object Service (WOS), Web Feature Service (WFS), Sensor Collection Service (SCS), Image Archive Service (IAS) and Web Coverage Service (WCS).

7.3 High-level use cases

Annex A enumerates and references a set of high-level use cases for multi-source information operations developed during the OWS1 initiatives (OWS1.1 and OWS1.2). These use cases attempt to identify the general cases for client and service interactions within a scalable, deployable OpenGIS Web Services Architecture.

8 Information Viewpoint

The information viewpoint into a service architecture covers the information and data structures which are processed by applications conforming to that architecture. This information includes information which might be characterized as either data or as metadata. The information viewpoint, however, concerns itself mainly with metadata which describes the state, functioning, and payload flow of components within the architecture.

One of the most important parts of this metadata is the service information which describes the capabilities and content of service components in the architecture. A general model for geospatial services information is covered in detail in a separate document, OWS Service Information Model (OGC 03-026).

9 Computational Viewpoint

The Computational Viewpoint considers the functional structure of an ODP system, whether characterized as a functional decomposition, extraction of functional commonality. In order to focus on the concept of function, this viewpoint considers ODP to be carried out by interactions between interfaces without referring, if possible, to how such interfaces may physically be implemented. The functional structure defines all service compositions and service interactions necessary to provide required system functionality.

9.1 Elements of the Viewpoint

The basic entity or component of a service-oriented ODP system is the service. There are any number of definitions of "service" from the abstract to the concrete and practical. One definition is "useful/meaningful combination of operational interfaces and accessible content". This contains most of the important terms which are needed for construction of a service model. Topic 12 definitions are shown here for comparison:

- Service: distinct part of the functionality that is provided by an entity through interfaces.
- Interface: named set of operations that characterize the behaviour of an entity.
- Operation: specification of a transformation or query that an object may be called to execute. It has a name and a list of parameters.

The proposed definition adds "content", a service feature which may be unique to the OGC view of services.

9.2 Interfaces

An interface may be defined as any set of one or more operations, but it would be much more useful to define a set of interfaces which maximized commonality and reuse – in other words provide optimal familiarity and stability in order to promote interoperability on both syntactical and semantic levels. The service model proposed here contains an interface hierarchy that breaks down operations into small groups that can normally be implemented as a whole by any given service to which it applies. The reuse and specialization of more general operations is then characterized as interface inheritance.

Note: There is some question as to whether this is "legal UML" since the relationship between interface inheritance and service inheritance may be ambiguous. The approach taken here is to restrict inheritance to the interfaces themselves, and regard service types as flexible implementations of some one or more interfaces (along with associated content types).

One important goal of this approach is to emphasize elements common to more than one service. Another important goal is to ease both the construction and comprehension of

new services by providing the well-understood interface components with which to accomplish it.

In summary, the motivations for this design approach include the following:

1. Enable reuse and specialization of interfaces
2. Enable flexibile definition and composition of services
3. Focus on elements (interfaces, operations, messages, content) common to more than one service
4. Ease construction and comprehension of new services
5. Provide well-understood interfaces defining “contracts” for constructing and interacting with services

9.3 Interface Hierarchy

The proposed interface hierarchy is arranged in a small number of toplevel interfaces, as illustrated in Figure 8 below. The eight interfaces shown in the figure are described below, while the details of the interfaces and interface hierarchy deriving from each toplevel interface are presented in Annex B. Each toplevel interface also corresponds to a package for all interfaces deriving from that interface. All interfaces are shown to specialize an abstract *WSInterface* interface; they overload existing operations as well as add additional operations and content types. Only the operation and its return type are defined here. Documentation of the operation signatures themselves is left for now to the individual service implementation specifications.

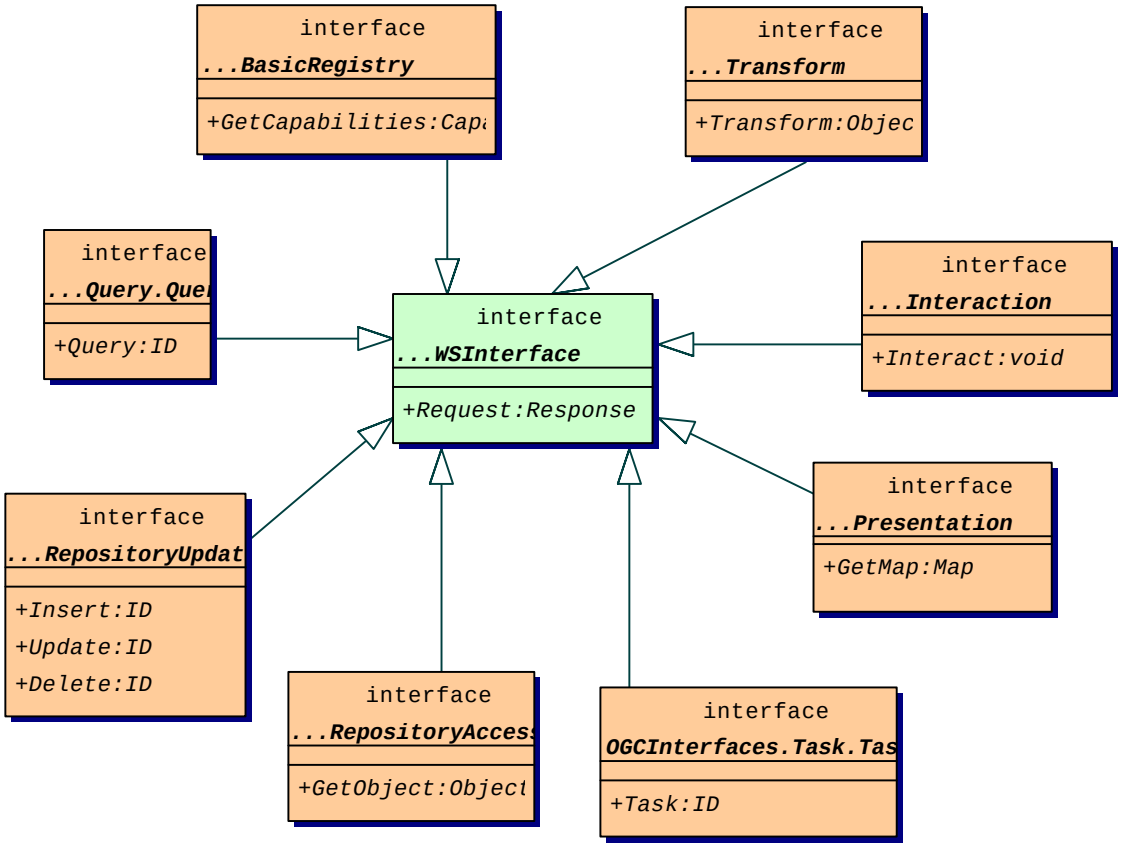


Figure 8. Top-Level Interfaces

9.3.1 BasicRegistry Package

The main operation in this interface is GetCapabilities which provides the basic registry functionality required for each self-describing service. This interface is implemented just once in each service and therefore provides a single GetCapabilities operation to describe any number of other operations being implemented by that service. Since GetCapabilities returns service information, any self-describing service also provides service information as its minimum accessible content.

9.3.2 RepositoryAccess interface

The main operations in this interface group are GetObject and derivatives for specialized content. Note that the most general version of this operation belongs to the repositoryAccess interface itself and only fetches content by ID. Services providing further query functionality must also implement a query interface (which happens to get invoked almost all the time nested within an operation such as GetObject. The most specialized operation in this group, GetRecord, adds the capability of mapping from one content model to another, an important value for registries and catalogs.

9.3.3 Query Package

The interfaces in this package provide various forms of query capability from describing types to returning feature information to supporting OGC Filter operations. Some of these operations are “standalone” while others are most often invoked inside of other operations. Perhaps they should be termed “parasitic operations”.

9.3.4 Presentation Package

The interfaces in this package concentrate on the presentation of content visualizations for human consumption.

9.3.5 Repository Update Package

The interfaces in this package provide the transactional (insert-update-delete) operations for any service which allows its content to be modified. More than any other interfaces besides BasicRegistry, the named operations making up this interface seem now to be standardized across a number of service types.

9.3.6 Task Package

The interfaces in this package cover a variety of ways of combining multiple operation invocations into one task, including transaction, chain, plan, and distribution. They all presume some form of content which defines the set of operations to be carried out, but differ in the degree to which each operation or operation choice depends on the ones before it.

9.3.7 Transform Package

The interfaces in this package focus on transformation from one content type or instance to another. For example, the interpolation interface may turn a point feature collection into a gridded coverage (type conversion) or re-project a feature collection from one coordinate system to another (instance conversion). Their content consists generally of service information alone, although some other form of content might be appropriate (e.g. coordinate system definitions).

9.3.8 Interaction Package

The interfaces in this package apply particularly to human interfaces. They provide operations which involve human interaction with a user interface and would generally be matched with the client equivalents of the appropriate interfaces in order to connect the user interfaces to real services.s

9.4 Information Flows

TBD

9.5 Persistence and Statefulness

TBD

10 Service Definitions

UML diagrams for each service definition in the proposed model are presented in Annex C. The following subclauses discuss classes of service types in terms of the interface – content combinations which define them. Each service implements (in addition to the interfaces described) a basic registry interface and service information content.

10.1 Mapping

This class of services concentrate upon the presentation of composed and rendered maps, implementing at a minimum a rendering interface. They may in addition implement a query interface (GetFeatureInfo) for providing more information about the map layers which are being served. Additional processing interfaces may also be supported (e.g. map re-projection).

10.2 Object Handling

These services implement repository data access at a minimum and may also support query, update, and transaction interfaces where appropriate. As a class of services, they focus upon the storage and retrieval of resources without too much regard to the content or type of those resources.

10.3 Registry_Catalog

Registries or catalogs are services focused upon the management of metadata, either building on or working from metadata repositories to provide query, update, translation, tasking, or other operations on metadata resources. Although functionally similar to both object and feature handling services, registry or catalog services are really distinguished by the metadata content which they serve.

10.4 Feature Handling

Feature handling services implement object handling interfaces specialized for feature data.

10.5 Image Handling

Image handling services implement object handling interfaces specialized for image data, which may be treated as image objects, as imagery data, as coverages, or as rendered map layers. Given the variety of ways in which images may be accessed, there is a large number of interfaces which may potentially be involved in such services.

10.6 Sensor Web Enablement

Although sensor web enablement services focus on measurements and observations, which may be characterized as features, such content is also characterized by particularly intricate relationships between those observations and the metadata documenting its validity, such as sensor descriptions, measurand dictionaries, mission tasks, and processing algorithms. A combination of feature, object, and registry interfaces is appropriate for such services, but they may also involve other interfaces such as tasks and transformations.

10.7 Client

Clients or "user-facing services" provide user interfaces which are not yet well defined in the OGC realm, but usually involve some combination of such elements as:

- Form navigation
- Map navigation
- Map drill-down
- Table navigation
- Query construction

all of which have "familiar" if not yet well codified interface definitions. Another important aspect of clients (user-facing or not) is that they implement the mirror image of any one of the interfaces supported by services they connect to.

11 Service Combinations

This clause compares and contrasts three methods for combining services on some level, either by type, instance, or content.

11.1 Overview

In all three cases (described below), we are just talking about interfaces. Nothing is specified about how many platforms are involved behind the scenes, or where data actually lives.

In all cases, we would need to know how to formulate the capabilities of each service involved. We would also need to know how to express the connections between content and content metadata for each operation. For example, image A available through GetObject is the same as what is rendered in GetMap layer B and the same as what is georectified in Coverage C.

Note: This is complicated by the case of "millions of images", which is not the level of metadata which it would be useful to expose through GetCapabilities. It is well enough understood that the images available through a WOS interface would only show up in capabilities as some sort of object type collection (possibly a single collection). What then happens to the capabilities of a GetMap interface on that same archive? Clearly, an image catalog would need to maintain metadata on individual images, but we don't, for example, have other catalogs which maintain metadata on individual features right now. This case can also be addressed by the use of the RenderMap interface instead, which defers to the GetObject contents and has no map layers of its own to worry about returning from a GetCapabilities request.

Further discussion of the service information requirements arising out of service combination will be found in the SIM DIPR.

11.2 Service Aggregation

The most common way to go at present is the service aggregation, where two or more distinct services refer to the same or similar content. For example, some un-named company might have a WMS and a WFS. Separate services, separate capabilities, even separate endpoints perhaps, but serving in at least some cases the same "content" in different ways. Easy on development, but possibly awkward to synchronize. Present implementation depend on “private agreements” as to content similarity.

11.3 Service Composition

In the second case, composition, interfaces which may have previously been implemented by two or more existing services are combined to form a new service. The basicRegistry interface with its GetCapabilities would only be implemented once in the new service, but it would need to provide information about all of the capabilities and content of the new service. This is the most integrated approach, but involves the most new development. Some kinds of opaque chaining may look like this, while still preserving separate services. They would still represent aggregation in the UML sense, however, since the chained-to service would still have independent existence. In the case of service composition, the interfaces composing the new service would have no independent existence.

11.4 Service Chaining

The third case, chaining, involves being able to invoke a second service through a given service. An example might be submitting an image and metadata to an Image Archive Service, and having the Archive then submit the metadata to one or more Image Catalogs once an ImageID has been generated. This approach is a bit awkward unless operation sequences are particularly important, or unless the first service is a facade in a UML sense, which just chains to other services to offer real functionality.

Without delving into the engineering which may be involved in service chaining, there appear to be three chaining approaches in terms of service interface involvement:

- **Opaque** – service chaining occurs anonymous behind one service interface without any implications for the functionality of that interface (e.g. a Web Map Service fetches features to support a NamedLayer).

- **Translucent** – one service is invoked to chain to another service behind it. The service interface must support the chain invocation (e.g. UserLayer with WFS reference in SLD).

- **Transparent** – a specific service chain manager is employed to invoke each service in a chain, provide operation inputs and route operation outputs to the next service. Requires service interfaces to support a "pipe" approach to inputs and outputs. This variety of service chaining has not yet been fully realized in OGC specifications or initiatives, but relates to reseach into such areas as Grid Computing)

12 Future Areas of Work

There are a number of areas of discussion among OWS 1.2 participants which are not strictly within the scope of OWS 1.2, but have clearly been identified as having critical importance to the future of OGC Web Services. In some cases, there are even component implementations and proposed specifications, but these areas are being included under Future Work in the context of OWS 1.2, since that is the focus of this document.

12.1 Semantic Processing

The first step of interoperability is to use common formats and messages, but the next step is to agree on the meaning of those elements. In most cases, this meaning is currently communicated by a private understanding between parties, augmented by the familiar structural configurations in which they occur. For example, "car" is well understood in the statement "Drive the car from A to B", but less well understood in "Set car to 5"

Many efforts are underway to key message and format elements with metadata which places the meaning of those elements automatically into a well-known ontological framework, such that people and even, to some extent, machines, can interpret the meaning carried by unfamiliar combinations of those elements. Some of this approach has been explored in the OWS 1.2 Image Handling IPR, but preliminary efforts have shown that this is a large task in and of itself. Expression of such metadata must also be accompanied by development both of the ontological framework and of methods for creating and processing this metadata. The Spatial Semantic Web lies somewhere at the end of these (large) efforts.

12.2 Security Frameworks

Real implementations of OGC Web Services require control of access, authentication of users, privacy of interaction, and many other aspects of security. Both the concepts of security and implementation standards for distributed computing (particularly Web Services) have been in rapid flux during this initiative and should reach an appropriate level of maturity for testbed activity in the near future.

12.3 Commerce Frameworks

The comments above on security apply to commerce frameworks as well. In addition, business models to which such frameworks can be applied, are in flux as well. In this case, some research is also required to prepare the ground for testbed activity.

12.4 Service Chaining

All of the components for a service chaining framework exist, except for strong motivation. In this initiative, requirements began to be developed for which various forms of service chaining are the best solution. The development of service chaining specifications will largely be driven in the future by further requirements of complex distributed applications which are now being designed on the top of the relatively simple foundation specifications implemented to date.

Annex A
(informative)

OWS1.2 Use Cases

A OWS1.2 Use Cases

A.1 Image Handling Use Cases

A.1.1 Crop Monitoring

A series of sub use cases in which analysts, administrators, contractors and farmers plan for acquisition of imagery and manage, identify, discover and analyze information related to a national crop-monitoring project.

Reference:
http://member.opengis.org/portal/twiki_ows12/pub/Main/OWSUseCaseRepository/UCIM1_EU_Crops.doc

A.1.2 Hurricane Evacuation

Analysts at command center need immediate, continuous input on approaching tropical storm in order to assess the potential danger, and determine the best route for escape. Data available include: Goes Satellite Data- visible, IR; Doppler Radar, Aerial Photography/Video, Dropsondes, Balloons, Station Data for various meteorological parameters; Flood stage data.

Reference:
http://member.opengis.org/portal/twiki_ows12/pub/Main/OWSUseCaseRepository/UCIM2_Hurricane.doc

A.1.3 Smart Farming

The farmer has the goal of making management decisions on the application of insecticide, fertilizer, and irrigation to his fields. To do this, the farmer and analysts

1) obtain a map of his fields

2) assess the need for fertilizer as a function of location in his fields

3) use historical data to find areas of crop damage due to insects, and determine appropriate corrective action

4) use decision support tools to minimize the environmental impact of applying fertilizer and pesticides

5) generate an irrigation schedule for today and tomorrow

6) review all the data to detect small areas (or points) where needed or useful data is missing or ambiguous.

Reference:
http://member.opengis.org/portal/twiki_ows12/pub/Main/OWSUseCaseRepository/UCIM3_Farmer.doc

A.1.4 Home Buyer

The buyer, in a real estate broker's office or from home, selects a neighborhood (from an Internet service provided to support home-buying in a region), and is provided an aerial view of it. Through a series of sub use cases, the buyer discovers, accesses relevant data and services to perform analysis to assess the age of the roof, the health of the trees, and the trends in the neighborhood.

Reference:
http://member.opengis.org/portal/twiki_ows12/pub/Main/OWSUseCaseRepository/UCIM4_Homebuyer.doc

A.1.5 Modern Soldier

The soldier is on a peacekeeping mission and uses information received from surveillance cameras in planes, drones, satellites, video day and night in a continuous stream from lightweight airborne platforms and still images to perform analysis of imagery (e.g., feature detection). Soldier shares a common view of the battle space and collaborates with colleagues. Soldier makes contingency mission plans: selects potential targets, finds optimum access and egress paths, prepares flight folders that pilots and other weapon control officers can use to train, navigate, execute, escape, etc.

Reference:
http://member.opengis.org/portal/twiki_ows12/pub/Main/OWSUseCaseRepository/UCIM5_Soldier.doc

A.2 Resource Publish, Find and Bind Use cases

A.2.1 Find Data

The analyst wishes to find pollution data in Athens.

Reference:
http://member.opengis.org/portal/twiki_ows12/pub/Main/OWSUseCaseRepository/UCArch1_FindData.doc

A.2.2 Find Services

The analyst wishes to locate and make use of services on to which she wishes to bind a client application.

Reference:
http://member.opengis.org/portal/twiki_ows12/pub/Main/OWSUseCaseRepository/UCArch2_FindServices.doc

A.2.3 Bind Data To Service

The analyst wants to bind discovered data instance to a service instance.

Reference:
http://member.opengis.org/portal/twiki_ows12/pub/Main/OWSUseCaseRepository/UCArch3_BindData2Service.doc

A.2.4 Bind Service To Service

The analyst wants to connect the server (producer) interface of one service to the client (consumer) interface of another service, then exploit the resulting connection.

Reference:
http://member.opengis.org/portal/twiki_ows12/pub/Main/OWSUseCaseRepository/UCArch4_BindService2Service.doc

A.2.5 Bind Map To Gazetteer

The analyst wants to get a base map of the area around a given place name.

Reference:
http://member.opengis.org/portal/twiki_ows12/pub/Main/OWSUseCaseRepository/UCArch5_BindMap2Gazetteer.doc

A.3 Feature Handling Use Cases

A.3.1 Bind Style to Feature Type Instance

The analyst issues a request, specifying a list of feature type names, styles, and a particular portrayal service, to a mapping system. An image is rendered and returned to the user.

Reference:
http://member.opengis.org/portal/twiki_ows12/pub/Main/OWSUseCaseRepository/UCSM1_BindMapSLD.doc

A.3.2 Bind Style to Coverage

The analyst wishes to use a Coverage Portrayal Service using SLD to style the coverages.

Reference:
http://member.opengis.org/portal/twiki_ows12/pub/Main/OWSUseCaseRepository/UCSM2_BindCoverSLD.doc

A.3.3 Browse Styles

The analyst wishes to browse the styles available from a given server.

Reference:
http://member.opengis.org/portal/twiki_ows12/pub/Main/OWSUseCaseRepository/UCSM3_BindStyles.doc

A.3.4 Find Styles

The analyst wishes to retrieve a style based on all (or some combination of) the following: style name, list of feature type names, symbols used in the style, style type (SLD, XSLT, etc.), SMS server-specific defaults.

Reference:
http://member.opengis.org/portal/twiki_ows12/pub/Main/OWSUseCaseRepository/UCSM4_FindStyles.doc

A.3.5 Publish Styles and Symbols

The analyst has defined a style and wishes to make it available for retrieval by others or for later retrieval himself. The user has created a symbol and wishes to make it available for retrieval by others or for later retrieval himself.

Reference:
http://member.opengis.org/portal/twiki_ows12/pub/Main/OWSUseCaseRepository/UCSM5_PublishStyles.doc

A.4 Source Processing (Sensor Web) Use Cases

A.4.1 Find Terrorist

Suspected insurgence of terrorists into a rural area. Analyst needs to verify cell phone and radio transmissions over a specified area.

Reference:
http://member.opengis.org/portal/twiki_ows12/pub/Main/OWSUseCaseRepository/UCSW1_FindTerrorist.doc

A.4.2 Radio Control eyes in the skies

Actors, including Data Requestor (information consumer), Unmanned Aerial Vehicle (UAV) ISR, UAV pilot, UAV and UAV ground station processing segment manager, process tasking request for collection and exploitation of UAV data for border patrol.

Reference:
http://member.opengis.org/portal/twiki_ows12/pub/Main/OWSUseCaseRepository/UCSW2_FindBorderEvent.doc

A.4.3 Sports Blimp

Blimps are used to provide overhead images of slow-moving sporting events and environments. The method is either 1) under control of the cameraman in the blimp, or 2) same, but with instructions provided by telephone from the ground coordinator. Addition sub use cases include:

1) Remote control of blimp camera, now with orchestrated multi-source scripting

2) Remote control of blimp camera, including stored scenes, now with specific camera tasking.

3) Remote control and tasking of blimp and its camera, now including competition for resources

4) Remote control and tasking of blimp and its camera with multiple customer tasking

5) All users have flexible tasking and current product acquisition; now including archives correlation and retrieval.

6) Interactive ground control of blimp camera (no change to blimp or camera performance). Organize multi-tasking so blimp is pull on command, staged and pre-reviewed from ground.

7) All users have total imagery utility; now including non-imagery and fusion information.

Reference:
http://member.opengis.org/portal/twiki_ows12/pub/Main/OWSUseCaseRepository/UCSW3_FindSports.doc

A.4.4 Request Sensor Collection – Dynamic Remote

The analyst requests imagery for mapping an airfield.

Reference:
http://member.opengis.org/portal/twiki_ows12/pub/Main/OWSUseCaseRepository/UCSW4_FindSensor.doc

A.4.5 Request Observation

Analyst requests sensor collection (observation).

Reference:
http://member.opengis.org/portal/twiki_ows12/pub/Main/OWSUseCaseRepository/UCSW5_RequestObservation.doc

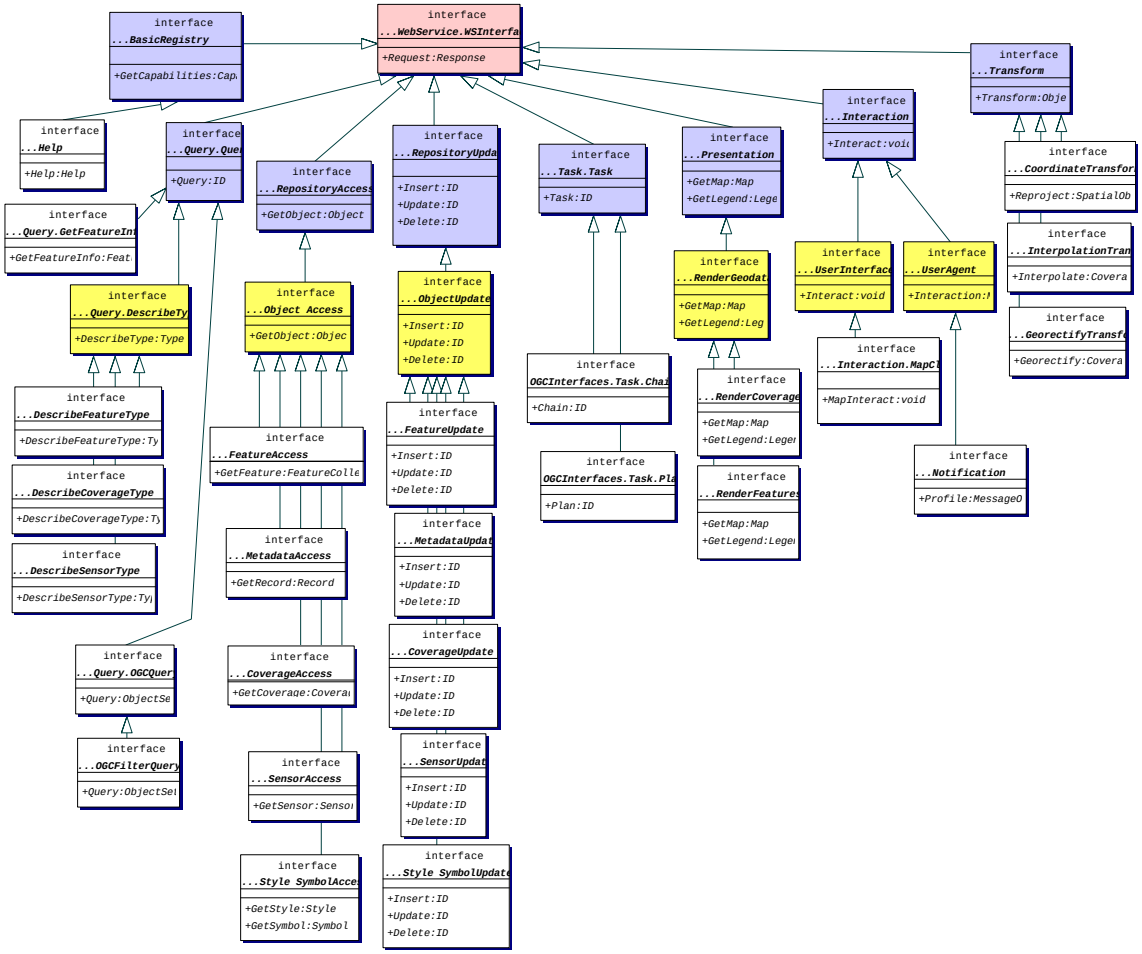
Annex B
(informative)

Interface Definitions

B Interface Definitions

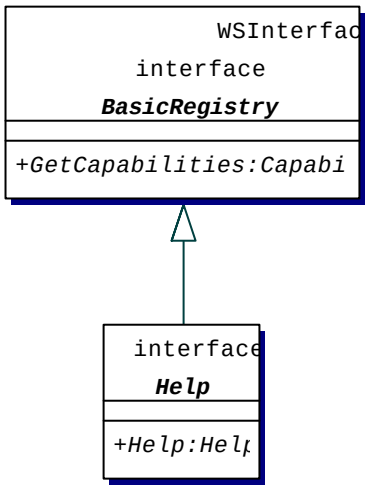
B.1 Interface hierarchy diagram

Figure A1 presents the complete OGC interface hierarch as so far developed. Individual packages are presented in the following sections of this annex. The full signatures of each presented operation are still to be documented within the Information Viewpoint context of the SIM DIPR.



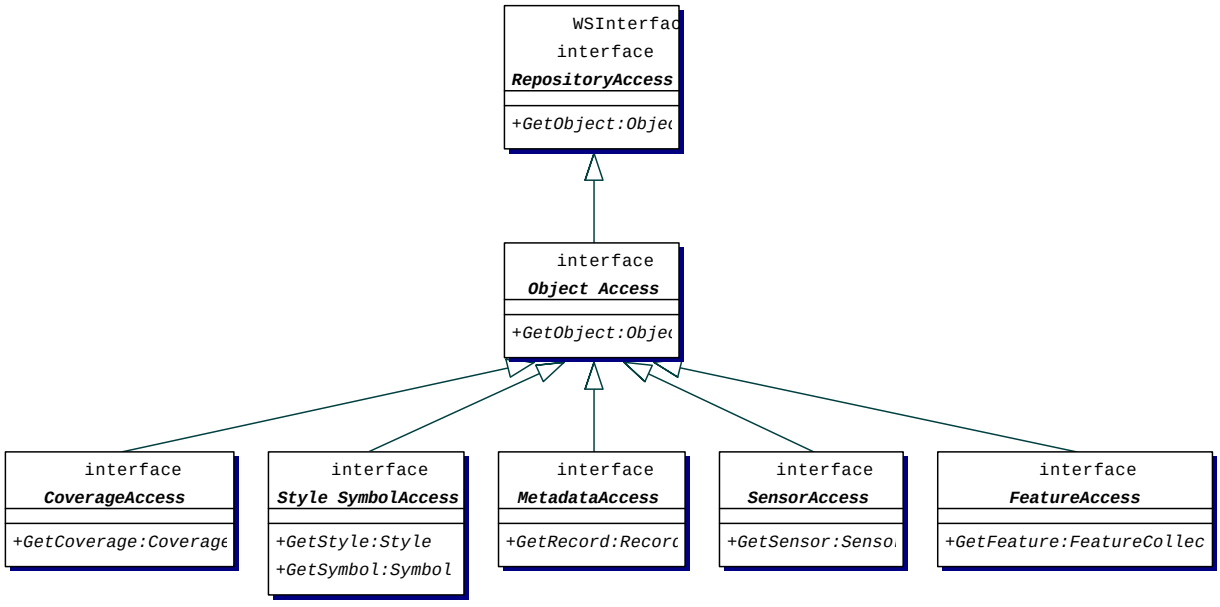
B.2 BasicRegistry

B.2.1 Class Diagram BasicRegistry



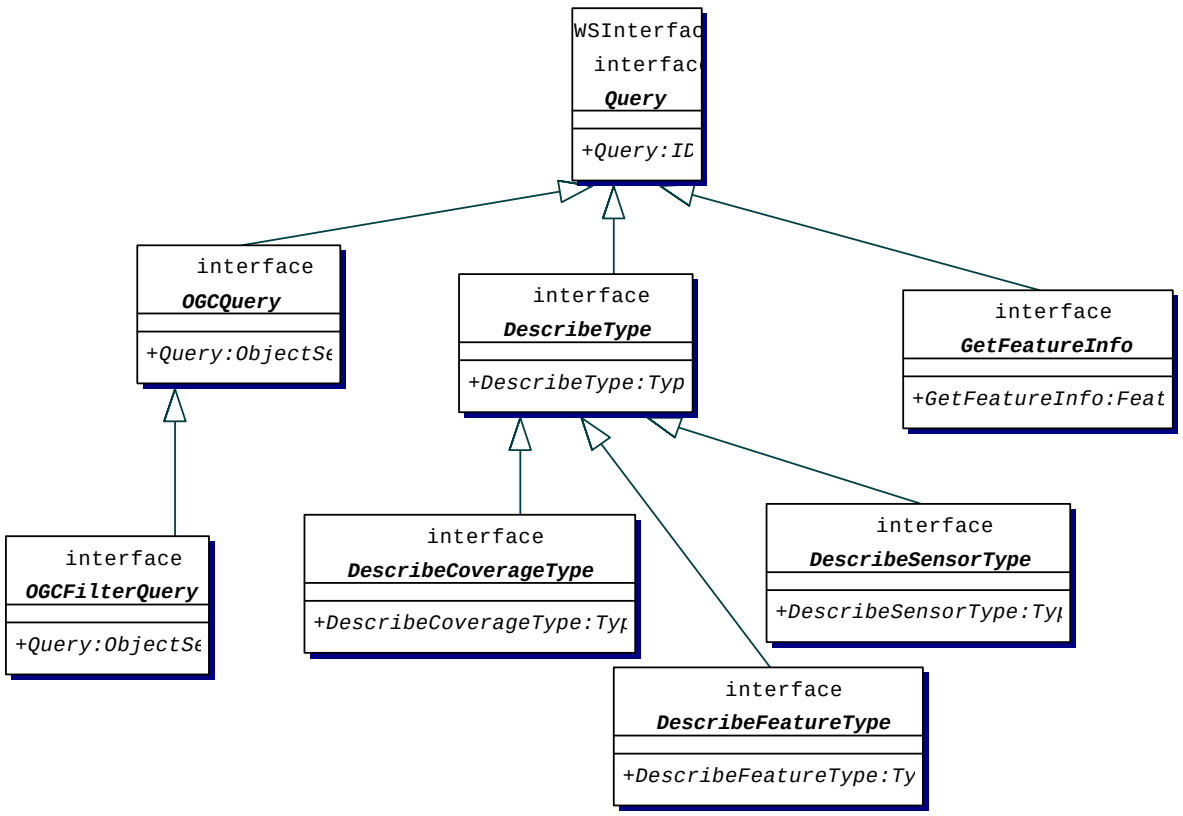
B.3 Repository Access

B.3.1 Class Diagram *RepositoryAccess*



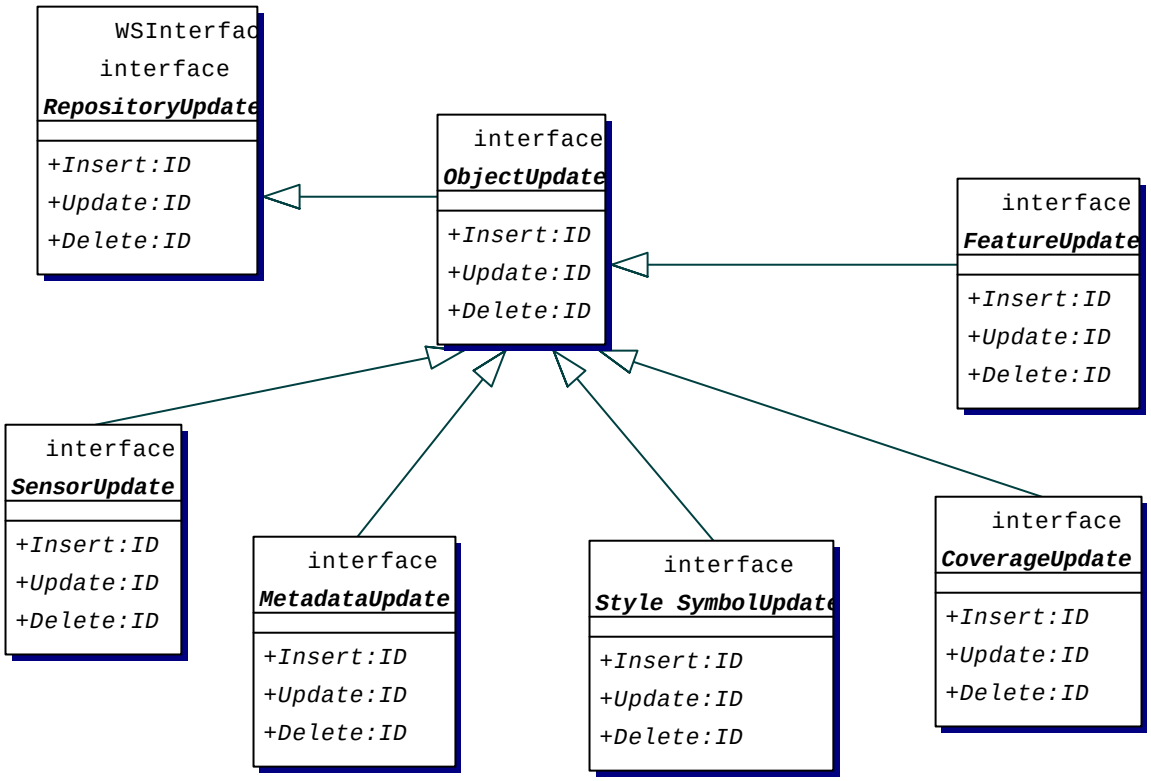
B.4 Query

B.4.1 Class Diagram Query



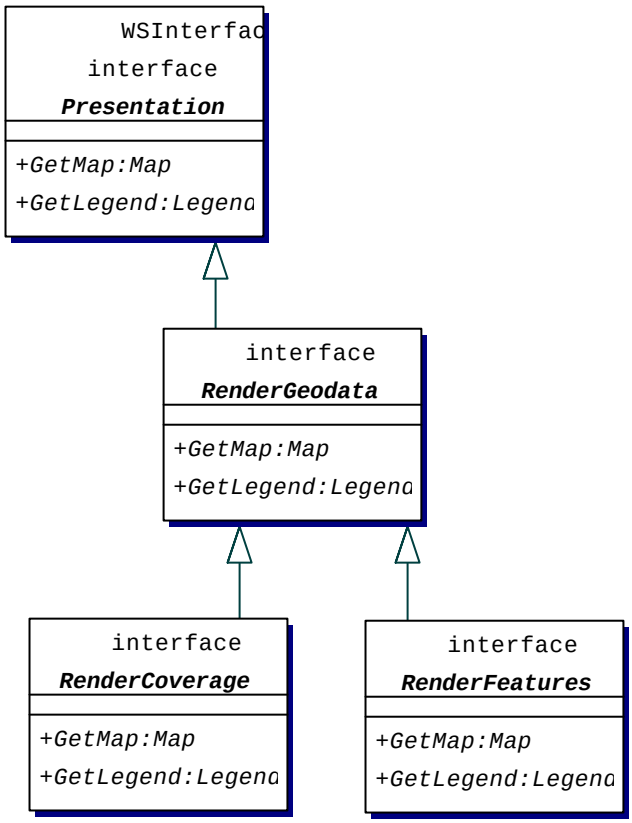
B.5 Repository Update

B.5.1 Class Diagram *RepositoryUpdate*



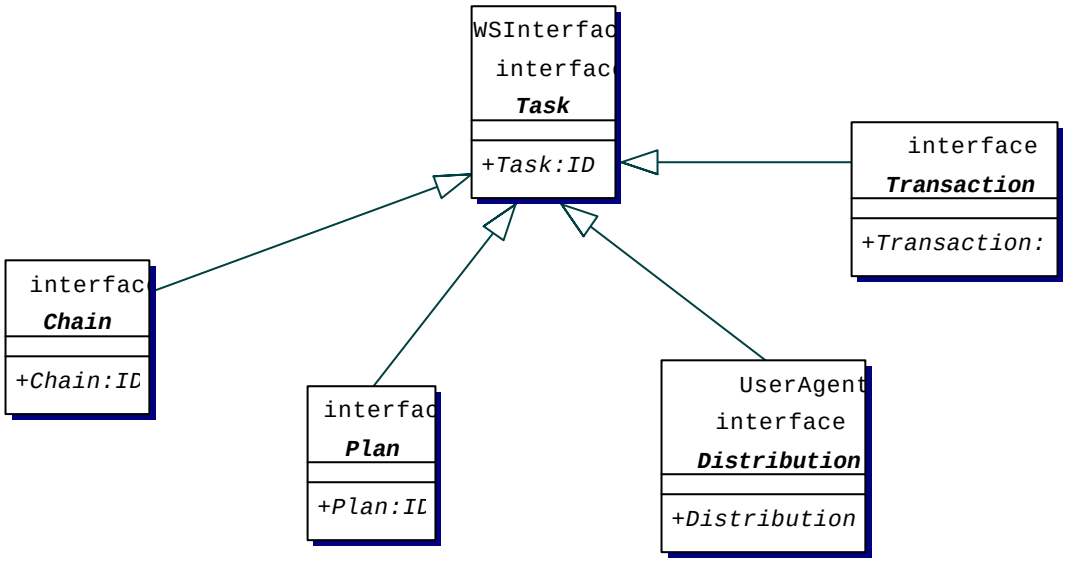
B.6 Presentation

B.6.1  Class Diagram *Presentation*



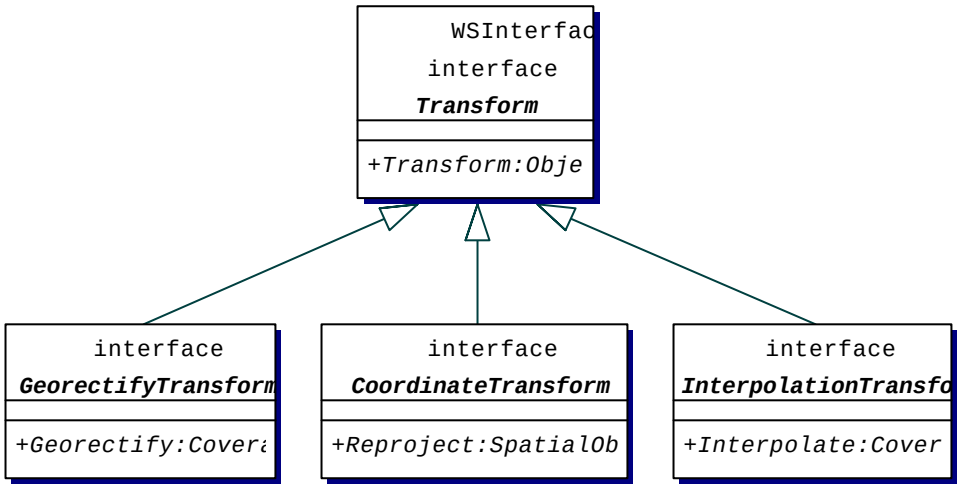
B.7 Task

B.7.1 Class Diagram Task



B.8 Transform

B.8.1 Class Diagram Transform



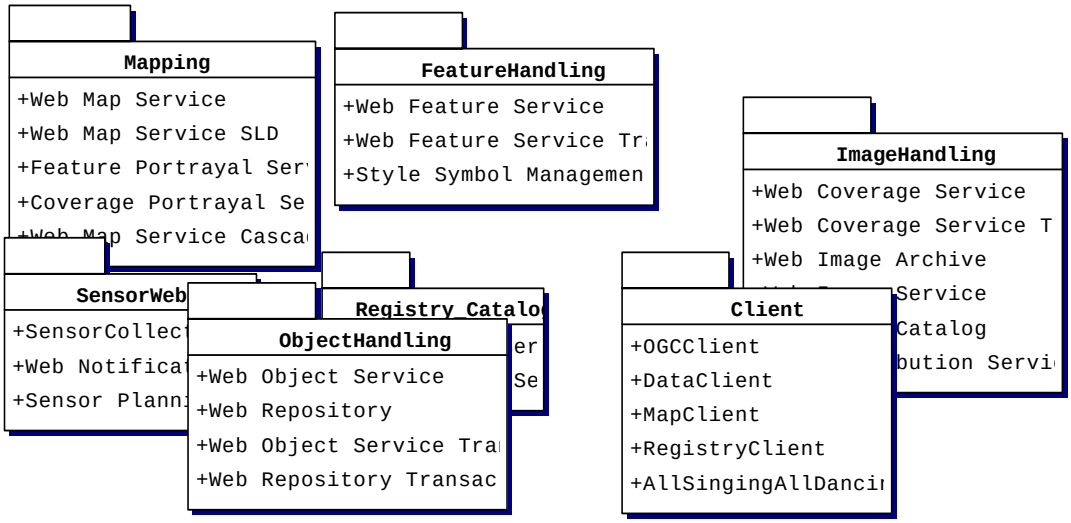
Annex C
(informative)

Service Type Descriptions

C Service Type Descriptions

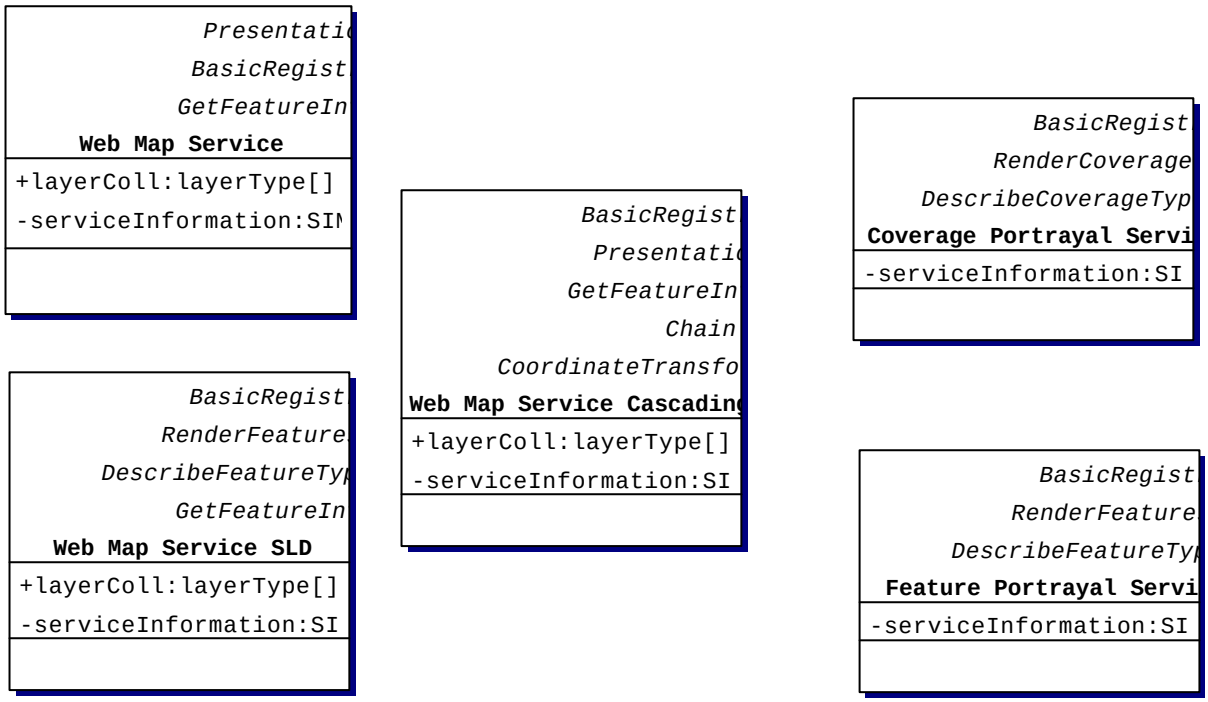
C.1 General
Figure A1 presents the toplevel packages which group particular service definitions by functionality and use. Which interfaces and content types are particular to each service type are defined in sections below.

C.1.1 Class Diagram OGCServices



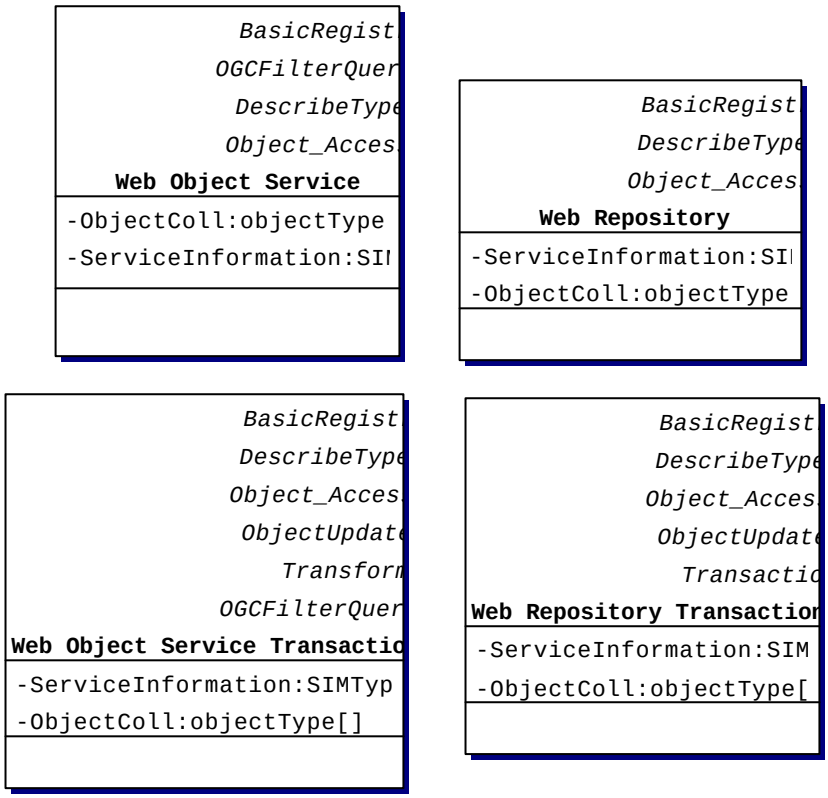
C.2 Mapping Services

C.2.1 Class Diagram *Mapping*



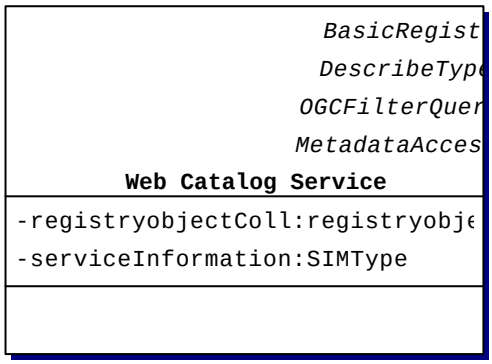
C.3 Object Handling Services

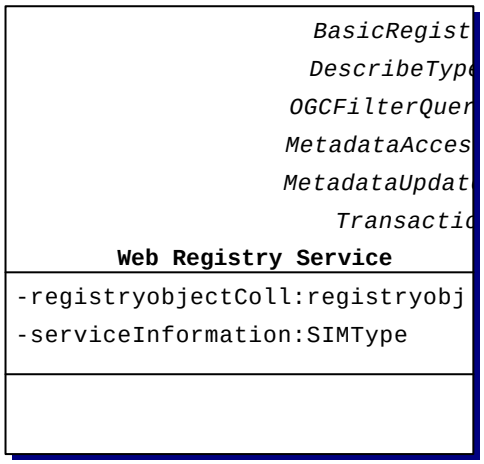
C.3.1 Class Diagram ObjectHandling



C.4 Registry/Catalog Services

C.4.1 Class Diagram Registry_Catalog

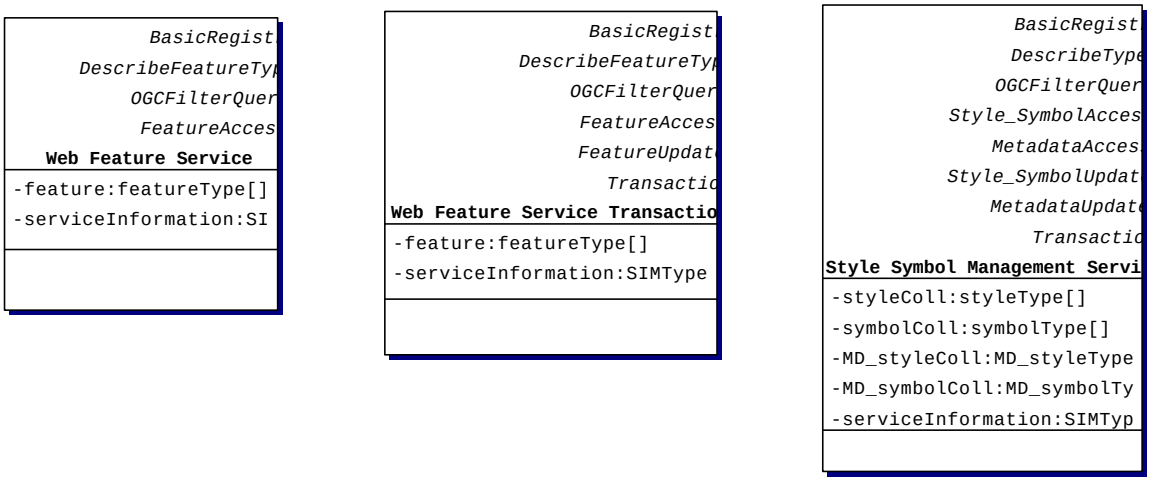




C.5 Feature Handling Services

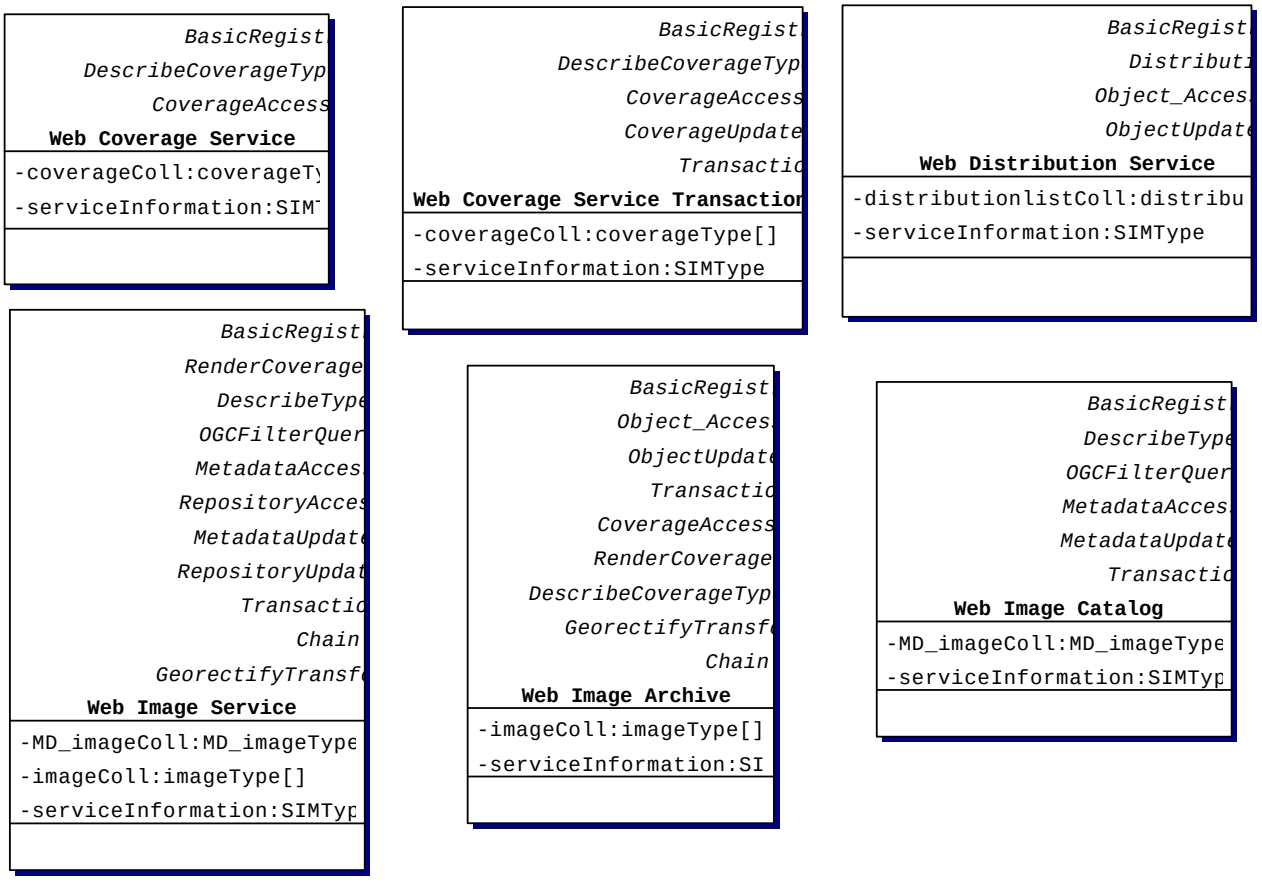
(SMS should be moved to registry services)

C.5.1 Class Diagram FeatureHandling



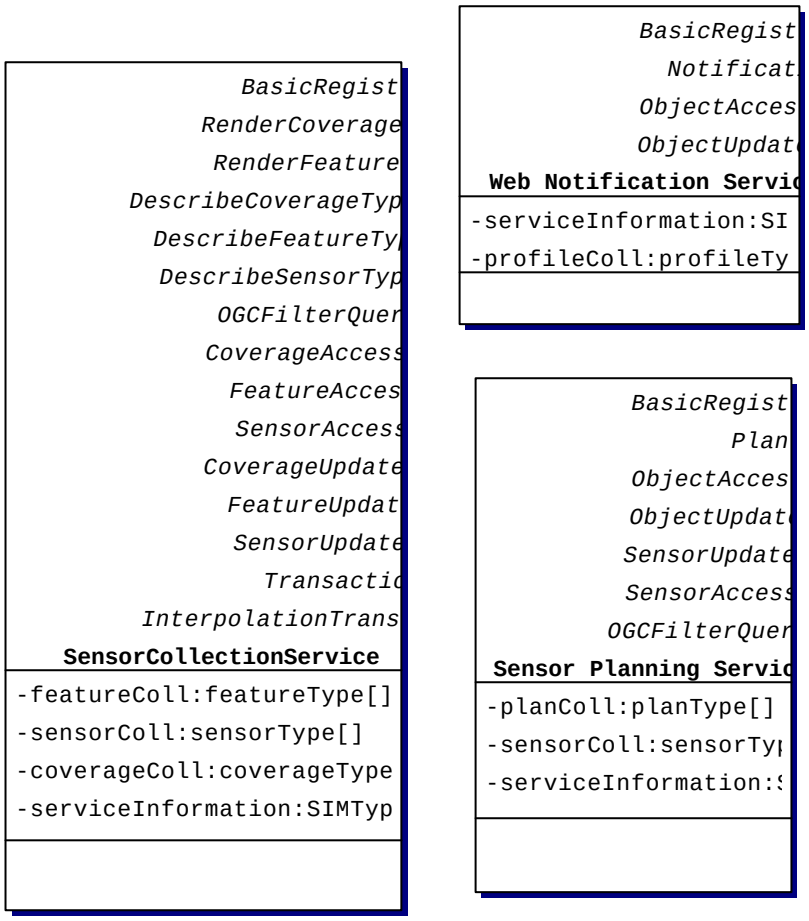
C.6 Image Handling Services

C.6.1 Class Diagram ImageHandling



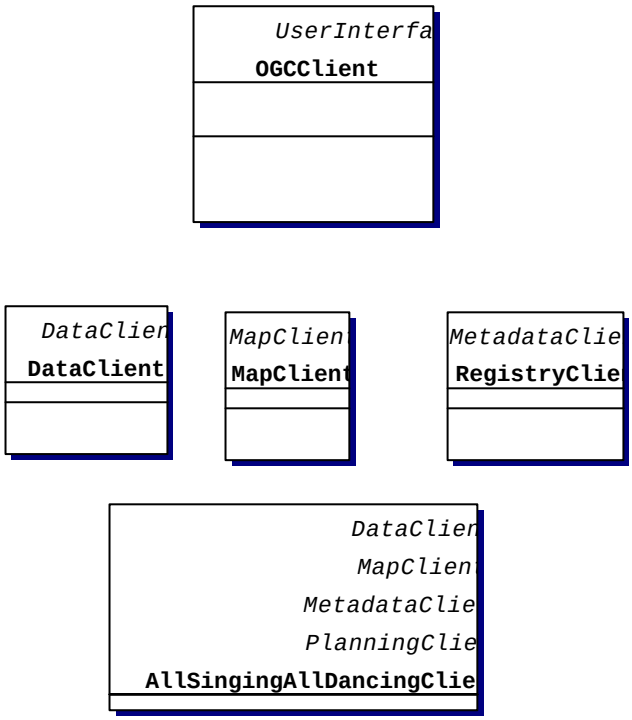
C.7 Sensor Web Enablement Services

C.7.1 Class Diagram *SensorWeb*



C.8 Client Services

C.8.1  Class Diagram *Client*



Bibliography

[1] ISO 31 (all parts), *Quantities and units*.

[2] IEC 60027 (all parts), *Letter symbols to be used in electrical technology*.

[3] ISO 1000, *SI units and recommendations for the use of their multiples and of certain other units*.

[4] Guidelines for Successful OGC Interface Specifications, OGC document 00-014r1