Open GIS Consortium

35 Main Street, Suite 5
Wayland, MA 01778
Telephone: +1-508-655-5858
Facsimile: +1-508-655-2237

Editor:
Telephone: +1-703-830-6516
Facsimile: +1-703-830-7096
ckottman@opengis.org

The OpenGIS™ Abstract Specification
Topic 13: Catalog Services

Version 4

OpenGIS™ Project Document Number 99-113.doc

NOTICE

The information contained in this document is subject to change without notice.

The material in this document details an Open GIS Consortium (OGC) specification in accordance with the license and notice set forth on this page. This document does not represent a commitment to implement any portion of this specification in any companies' products.

While the information in this publication is beleived to be accurate, the Open GIS Consortium makes no warranty of any kind with regard to this material including but not limited to the implied warranties of merchantability and fitness for a particular purpose. The Open GIS Consortium shall not be liable for errors contained herein or for incidental or consequential damages in connection with the furnishing, performance or use of this material. The information contained in this document is subject to change without notice.

The Open GIS Consortium is and shall at all times be the sole entity that may authorize developers, suppliers and sellers of computer software to use certification marks, trademarks, or other special designations to indicate compliance with these materials.

Revision History

Date	Description
31 March 1999	Carry forward 98-113r2 and rename to 99-113; update copyright for 1999; use revised document template; move former Section 2.1 to new Section 1.2; update cross-references to references, sections, figures and tables.

This page is intentionally left blank.

Table of Contents

1. Introduction

1.1. The Abstract Specification

The purpose of the Abstract Specification is to create and document a conceptual model sufficient enough to allow for the creation of Implementation Specifications. The Abstract Specification consists of two models derived from the Syntropy object analysis and design methodology [1].

The first and simpler model is called the Essential Model and its purpose is to establish the conceptual linkage of the software or system design to the real world. The Essential Model is a description of how the world works (or should work).

The second model, the meat of the Abstract Specification, is the Abstract Model that defines the eventual software system in an implementation neutral manner. The Abstract Model is a description of how software should work. The Abstract Model represents a compromise between the paradigms of the intended target implementation environments.

The Abstract Specification is organized into separate topic volumes in order to manage the complexity of the subject matter and to assist parallel development of work items by different Working Groups of the OGC Technical Committee. The topics are, in reality, dependent upon one another— each one begging to be written first. *Each topic must be read in the context of the entire Abstract Specification*.

The topic volumes are not all written at the same level of detail. Some are mature, and are the basis for Requests For Proposal (RFP). Others are immature, and require additional specification before RFPs can be issued. The level of maturity of a topic reflects the level of understanding and discussion occurring within the Technical Committee. Refer to the OGC Technical Committee Policies and Procedures [2] and Technology Development Process [3] documents for more information on the OGC OpenGIS™ standards development process.

Refer to Topic Volume 0: Abstract Specification Overview [4] for an introduction to all of the topic volumes comprising the Abstract Specification and for editorial guidance, rules and etiquette for authors (and readers) of OGC specifications.

1.2. Introduction to Catalog Services

This topic, labeled OpenGIS Catalog Services, covers the Geospatial Information Access Services discussed in Section 3.1.1 of Topic 12: The OpenGIS Service Architecture. Those Geospatial Information Access Services include Geospatial Information Retrieval Services (3.1.1.1), Geospatial Product Information Services (3.1.1.2), and Geospatial Catalog Services (3.1.1.3). This topic thus covers OpenGIS services for both data discovery and data access.

Discovery and access services are complementary in nature and are expected to have somewhat similar interfaces. However, this topic discusses them independently since they could be implemented separately. That is, some implementations may provide only discovery services, while others may provide only access or both discovery and access services.

1.2.1. Overview

We use the term “Catalog” to describe the set of service interfaces which support organization, discovery, and access of geospatial information. Catalog services help users or application software to find information that exists anywhere in a distributed computing environment. A Catalog can be thought of as a specialized database of information about geospatial resources available to a group or community of users. These resources are assumed to have OpenGIS feature, feature collection, catalog and metadata interfaces, or they may be geoprocessing services.

Catalogs have three essential purposes:

- to assist in the organization and management of diverse geospatial data and services for discovery and access,
- to discover resource information from diverse sources and gather it into a single, searchable location, and
- to provide a means of locating, retrieving and storing the resources indexed by the catalog.

1.2.2. Motivation

Networks contain vast amounts of geospatial information resources that are distributed among multiple databases, stored in many formats and maintained by myriad individuals and organizations. Navigating even a carefully designed network can be confusing to those looking for specific information. A successful search can be nearly impossible, for example, when the domain of a search extends to an organization-wide collection of networks and data sources.

A user looking for information has no alternative but to browse known locations and navigate through links, hoping to find the desired resources. But unless the user already knows where to look, browsing for information is generally the least effective method of finding information.

One way of improving the process of searching the expanse of networks, servers, databases, documents and resources is by cataloging— grouping specific information about a specific set of resources in a form that users and applications can search in a number of ways. A catalog brings together enough information about a set of resources so that users can pinpoint and retrieve what they need.

The catalog helps reduce the number of locations the user (or application) visits while looking for a particular resource, thereby reducing the time required to find useful information and increasing the probability of actually locating the desired resources. In library parlance, catalogs improve the recall and precision of searches, where "recall" measures the ability to find all applicable information and "precision" measures the ability to weed out unwanted information. In addition, the catalog gives data producers, publishers and librarians control over what resources are available.

1.2.3. What can be cataloged?

Catalogs store and distribute information about geospatial resources. We envision geospatial resources, in this context, to have one of OpenGIS feature, feature collection, catalog or metadata interfaces. Other types of information (e.g., graphic images, plain text files, applications/applets, documents, etc) could of course be added to a catalog in support of resource retrieval but these would likely be accessed as properties of OpenGIS features, feature collections, catalog and metadata entities. In any case, the structure and content of the resources stored in or referenced by a catalog is determined by the catalog's schema— the set of queryable property name-type pairs used to describe geospatial resources.

1.2.4. Resource descriptions

The individual entries in a catalog are resource descriptions. A catalog is therefore a container of resource descriptions that can be arranged and subdivided in various ways. These resource descriptions, we call them catalog entries, are essentially summary information about a particular resource. Like a card catalog in a library, catalog entries can be organized and indexed in various ways— alphabetically, by author, title, subject, type, date, cost, fitness for use, etc.

1.2.5. Summary

Catalogs are used to:

- represent large volumes of geospatial resources
- present a common format (or software interface) for the information describing each resource
- arrange resource descriptions to facilitate easy access
- provide mechanisms for access and retrieval of the desired resource Catalogs enable:
- resources to be organized in various ways without changing the content or physical organization of the data (easier maintenance and organization of resources)
- a standard way of describing those resources
- users to find information about resources
- access to desired resources once they have been located
- multiple ways of viewing the same resources

- remote access to resources from multiples sources, formats and locations

1.3. References for Section 1

[1] Cook, Steve, and John Daniels, Designing Objects Systems: Object-Oriented Modeling with Syntropy, Prentice Hall, New York, 1994, xx + 389 pp.

[2] Open GIS Consortium, 1997. OGC Technical Committee Policies and Procedures, Wayland, Massachusetts. Available via the WWW as <http://www.opengis.org/techno/development.htm>.

[3] Open GIS Consortium, 1997. The OGC Technical Committee Technology Development Process, Wayland, Massachusetts. Available via the WWW as <http://www.opengis.org/techno/development.htm>.

[4] Open GIS Consortium, 1999. Topic 0, Abstract Specification Overview, Wayland, Massachusetts. Available via the WWW as <http://www.opengis.org/techno/specs.htm>.

2. The Essential Model for Catalog Services

2.1. Context

The purpose of this section is to explain the "essential" objects, interfaces, behaviors and parameters of Catalogs [12]. We begin by introducing three real-world perspectives that provide context and motivate the need for OpenGIS Catalogs. Next, we talk about the essential functions of Catalogs: discovery, access and librarian functions. Finally, we introduce a conceptual model of Geospatial Resources and discuss its components individually.

2.1.1. The Library View

Geographic Information Communities [1] need a technology that empowers them to announce the existence of themselves and their information, so that other individuals outside that GIC may discover them and assess whether there may be interest in sharing their information. Since these services are traditionally provided by libraries, we postulate a Library World specific for digital geospatial information.

The notion of Library World is decomposed into functional areas that support aspects of digital information ingestion, storage, access and dissemination.

The Catalog is the fundamental tool for digital geospatial information discovery. As with traditional libraries, we postulate that a catalog is a "gateway" through which users can match their information requirements to the most appropriate information set available. We use the traditional (albeit, digital) library as a model to describe the context and motivation for OpenGIS catalogs. It is just one of many possible contexts for catalogs.

In the abstract model shown in Figure 2-1, there are two "objects" and three "actors": the objects are catalogs and storage collections and the actors are providers, librarians and users. We use these to describe the Library World aspects of a Geographic Information Community.

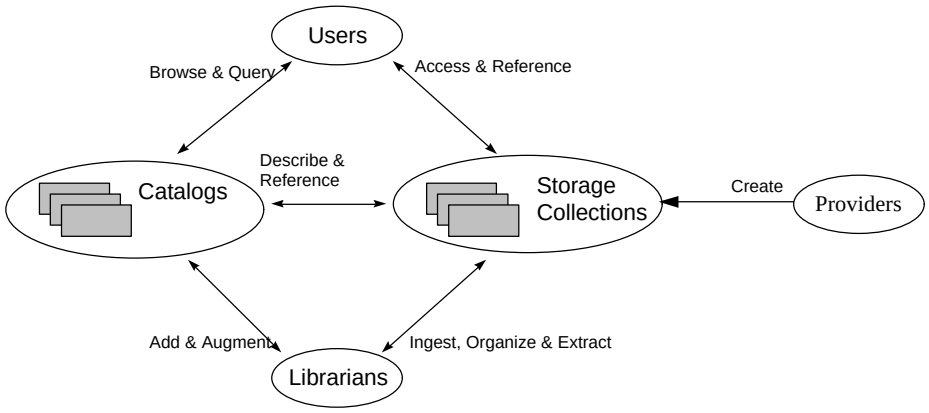

Figure 2-1. Catalogs in the Library World.

The *user* actor supports graphics and text based interactions by humans and automated "agents." Users are consumers of both information found in catalogs and storage collections. Users browse and query catalogs in order to access or reference information in storage collections of interest.

The *catalog* object includes metadata (information like who, what, why, when, where and how) and search engines that let users identify holdings of interest. Catalogs describe and reference content found in storage collections and in other catalogs.

The *storage collection* object contains data holdings comprised of feature collections, features and metadata of a Geographic Information Community. Storage collections are offered by data providers.

The *librarian* actor manages catalogs and collections. Librarians ingest, organize and extract information in storage collections and add and augment catalog entries.

The *provider* actor creates and publishes storage collections.

2.1.2. The Information Consumer View

Refer to Section 7.1 for a use-case description of an information consumer utilizing catalog services.

2.1.3. The Information Provider View

Refer to Section 7.2 for a use-case description of an information provider utilizing catalog services.

2.2. Essential Functions

2.2.1. Discovery

We want to use catalogs to organize, discover, navigate geodata of interest. In order to accomplish this we must leverage the information available to us as metadata. It is no coincidence, therefore, that the reasons for using a catalog closely parallel the intended uses of metadata [1]. We use catalogs:

- to search for feature collections using metadata key words and their associated values.
- to find information that may assist a process (e.g., a workflow or a particular geoprocessing operation) that is about to be attempted.
- to query information about geodata to determine the fitness of the data for the application at hand.
- to query information about geodata to determine whether data exists or not.
- to enable different “views” or “profiles” of the same geodata tailored, for example, for different disciplines, application domains, groups of users or natural languages.
- to refine queries, including querying on the results of a query

2.2.2. Access

- direct access to data distinguished from access through references or metadata entities.
- ability to access selected subsets of the information referenced in the catalog.

2.2.3. Librarian

- define, modify and destroy catalog instances
- to add, modify or delete catalog entries

2.3. Geospatial Resources

The previous sections provide motivations and functional requirements for the Open GIS Catalog services. This section discusses the resources and relationships needed to support these requirements.

OpenGIS Catalogs may be used for discovery of both geodata and geoservices. We use the term “georesource” to represent and encompass both concepts. A Geospatial Resource is the basic (but abstract) unit of geospatial information in a networked computer system environment. It is a unit of trade in a geospatial information sharing transaction— the primary geospatial item exchanged through geospatial commerce. A Geospatial Resource is also the primary object manipulated by geospatial software applications. But which type of Geospatial Resource is the basic unit of information sharing, commerce and exploitation?

The answer depends on the application and the role an actor plays in an information community. To a “producer,” the Feature Collection may be the primary unit of geodata management and commerce. To a “consumer,” a Service (“I need to know the fastest route to the nearest courier drop-box”) may be the unit of interest as well as the unit on which transactions are based. To “librarian” or “data broker” actors, Catalogs and Catalog Entries may be the primary units for organizing geodata resources and advertising their availability.

The following diagram depicts the basic relationships that exist between common types of georesources and the catalogs that reference them.

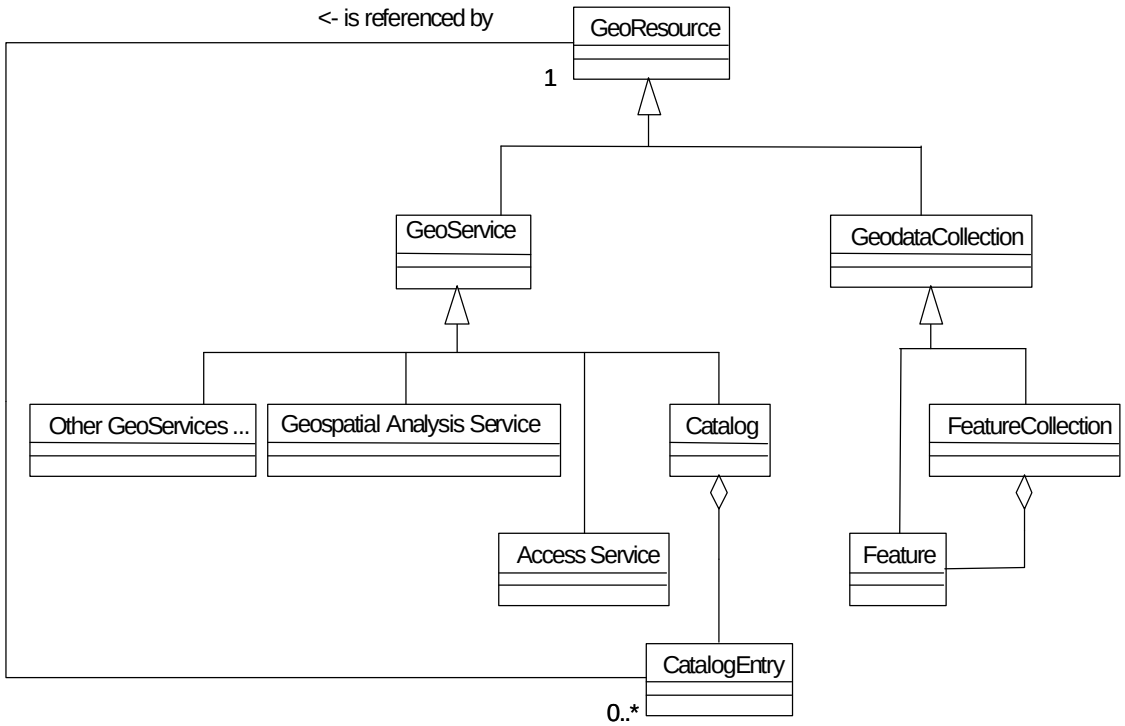

Figure 2-2. Geospatial Resources

In essence, Catalogs contain Catalog Entries and Catalog Entries reference GeoResources. Since Catalog is a type of GeoService, and GeoService is a type of GeoResource, Catalog Entries in one Catalog may reference other Catalogs. They may also reference other types of GeoServices, including geodata access services, shown in Figure 2-2 as Access Service.

More frequently, Catalog Entries will reference various flavors of geospatial data, including individual features and collections thereof.

2.3.1. Features

Features model or represent real or imaginary things or phenomena on the earth. In general, a Feature has:

1. A type,

2. One or more descriptive properties,

3. Zero or more geometries (which have a spatial reference system),

4. Metadata,

5. Feature-to-feature relations, and

6. General purpose and type-specific behavior.

All of these parts conform to the "language" of the Project World as described in Topic 5, The OpenGIS Feature [1].

2.3.2. Feature Collections

A Feature Collection may contain many features, or a single feature. An arbitrary level of granularity is not imposed on a Feature Collection, as a Feature Collection may contain arbitrary numbers and types of features. For example, a building, elevation measurement, road segment, and several terabytes of elevation data may all be members of separate (or the same) Feature Collections. A Feature Collection presents a standard interface through which its content can be accessed.

Depending on the policy under which they are managed, Feature Collections may be available on-line, where the dataset is available for direct access over the network. In this environment, it may even be possible to access features directly without accessing the Feature Collection they are contained within. Feature Collections may also be stored off-line in what we think of as a traditional archive and digital library management framework. In this case, the data is stored off-line and must be ordered for shipment as part of a separate transaction. Feature Collections that are stored off-line require additional information and/or processes in order to be discovered and retrieved.

Individual Feature Collections, for instance, may take the form of on-line databases with a front end to support on-the-fly map generation and retrieval. In this approach, a database of geospatial information is available on-line for users to access and request tailored subsets of the Feature Collection. The desired subset is specified by providing a set of parameters to retrieve the desired information from the database, symbolize the results for display, and package the results for transmission to the requester. Subsets of feature collections created by these user-defined parameters may be downloadable to user clients.

Topic 5, The OpenGIS Feature [1] provides a broader and more complete description of the intended uses, content, and interfaces of feature collections.

2.3.3. Metadata

Metadata is data about the content, quality, condition, and other characteristics of more basic data. For example, the FGDC [3] describes metadata as having these kinds of descriptive attributes:

- Identification Information
- Data Quality Information
- Spatial Data Organization Information
- Spatial Reference Information
- Entity and Attribute Information
- Distribution Information
- Metadata Reference Information

Depending on the context in which they are used, metadata may exist implicitly or explicitly as part of, or separate from, the geospatial datasets they describe. Topic 11: Metadata [1] provides a broader and more complete description of the content, interfaces and intended uses of metadata. Within the OpenGIS Essential Model, we consider metadata to be a conceptualization that in no way dictates a physical implementation, either in terms of its content or structure.

2.3.4. Catalog Entry

A Catalog Entry describes or summarizes the contents of a set of geospatial data, and is designed to be queried. A Catalog Entry is usually a subset of the complete metadata for the described geospatial dataset. However, a Catalog Entry can be the complete set or a superset of the corresponding metadata. To avoid confusion with general metadata, we abstract the metadata needed for data discovery into an object type and call it a Catalog Entry. A Catalog Entry object allows its content and structure to be queried, identified, described, and retrieved.

A Catalog Entry provides a high-level description of the referenced dataset, including:

1. Where, the region of the Earth covered by the dataset

2. What, the thematic key words, approximate scale, etc. of the dataset

3. Who, the agent responsible for the dataset

4. When, the date the dataset was created and the date of its sources

5. How, the instructions for access to acquire the dataset

6. Why, the intended use of the dataset

This Topic does not specify the metadata contents of a Catalog Entry, in terms of semantics, names, types, and values. For good interoperability and broad utility, we assume that these metadata element properties (the names, types, values, and semantics) will be compatible with metadata standards. Such metadata standards are defined by FGDC, ISO/TC211, CEN/TC287 and their profiles [2,3,4]. This compatibility is a practical consideration that must be defined and agreed on by groups of providers and consumers (and librarians) having common interests within Geographic Information Communities. This compatibility is thus outside the scope of the OpenGIS specification.

A Catalog Entry may contain copies of the appropriate metadata of the referenced dataset. A Catalog Entry may also contain metadata properties whose values are derived, computed, or otherwise augmented (perhaps using automated mechanisms) from the actual metadata values of the dataset. For example, a Catalog Entry may contain a reduced resolution version of an image or other dataset.

A Catalog Entry can describe either an OpenGIS Feature Collection, or an individual Feature (or Coverage). A single Catalog can contain a mixture of different Catalog Entry types that are related only by their membership in the Catalog. However, the same interface should be provided by all Catalog Entries in one catalog.

Depending on the application and implementation, a Catalog Entry may use entirely different semantics than the metadata directly associated with the described dataset. For example, the metadata of a Feature Collection instance may conform to the US FGDC Content Standard for Digital Geospatial Metadata [3]. However, one of that Collection's many possible Catalog Entries may use metadata defined to meet the more specialized needs of a relatively small group of users.

One of the responsibilities of the librarian actor is to make storage collections most useful to a community of users. The interests of the community may be very broad or extremely specialized. In either case, the librarian may choose to extract and augment metadata about geospatial datasets in storage collections, saving the results as Catalog Entries that are managed within a Catalog. The Catalog is constructed to serve a specific community, but may be equally (and unintentionally) useful to other communities.

2.3.5. Catalogs

A Catalog is a collection of Catalog Entries that is organized to assist in the discovery and retrieval of Datasets which are of interest to the user, especially when the existence of a Dataset is not initially known to the user. Catalogs serve an intermediate role in the discovery of Datasets in a networked information environment. Catalogs are most useful in managing large volumes of Feature Collections that are related by some common property, such as off-line storage or product series. That is, a Catalog serves as an intermediary mechanism between the Geospatial Datasets (e.g., Feature Collections) and the querying software. The querying software could be a client application, a middleware component such as an agent or trader, or something else.

Catalogs may be static or dynamic. Static catalogs include those used within formal digital library systems that manage large holdings of archived digital geospatial data. Such catalogs are populated with explicit Catalog Entries containing metadata created through either manual or automated means. Dynamic catalogs are not created until the requester submits a query. The catalog is created on the fly using Catalog Entries extracted from the metadata of other datasets that are discovered as a result of the query.

Catalogs vary in the types of objects referenced. Catalogs contain Catalog Entries describing other Datasets (i.e., Feature, Feature Collection, or Catalog) instances. The decision of which Dataset types to reference, or whether to create a Catalog at all, is left up to implementers to optimize to meet user needs. A Catalog presents a standard interface by which its contents can be described.

Feature Collections (or their associated Catalog Entries), do not necessarily have to be part of any Catalog. Publishers of geospatial information may have a product that they want to make available to others via the network. They may not have a sufficient volume of geospatial information or available resources to justify the formal management overhead of a Catalog. Thus, Feature Collections may reveal their metadata directly to users via search mechanisms without an intermediary Catalog.

2.3.6. Services

Services are reusable, self-contained collections of executable software components. They may be pieces of software that can play in different operating systems, networks and application frameworks. A service is not bound to a particular program, computer language or implementation. They are the building blocks for creating highly integrated and distributed application systems.

Information system tool builders, integrators and users must be able to coherently mix and match the most suitable set of tools to meet their constantly changing requirements. They want to be able to construct “build to order” application systems by assembling, perhaps dynamically, off-the-shelf software components. Services, frequently called pluggable tools or distributed components, let us create whole applications from reusable software parts.

A service typically has these characteristics:

- **It is a self-contained entity.** A service is an “off-the-shelf,” often marketable, piece of binary software.
- **It is not a complete application.** A service is designed to perform a limited set of tasks within an application domain. It can be combined with other services to form a complete application.
- **It can be used in unpredictable combinations.** It can be used in ways that were totally unanticipated by its developer.
- **It has a well-specified interface.** A service can only be invoked through its interface. The interface exposes the service’s function to the outside world. A service’s interface is separate from its implementation.
- **It is interoperable.** A service can be invoked across address spaces, networks, languages, operating systems and frameworks. It is a system-independent software entity.

2.4. Known Uses

FGDC Clearinghouse [3], [11]

NASA EOS/DIS.

ESA/CEOS [5].

NIMA (CIIF/USIGS/GIAS/DAGS) [7], [8], [10].

ISO/OAIS [9].

2.5. Related Topics

Semantics and Geographic Information Communities [1].

Metadata [1], [3], [11].

Features [1].

Traders.

2.6. References for Section 2

[1] OpenGIS™ Abstract Specification, OpenGIS™ Project Documents 99-100 through 99-116, available through www as <http://www.opengis.org/techno/specs.htm>.

[2] CEN/TC287 Secretariat (1996), **CEN/TC 287 Geographic Information**, http://www.statkart.no/sk/standard/cen.

[3] Federal Geographic Data Committee (FGDC) (1994, rev 1997), **Content Standards for Digital Geospatial Metadata**, http://www.fgdc.gov/metadata.

[4] ISO/TC 211 Secretariat (1996), **ISO/TC211 Geographic Information/Geomatics**, http://www.statkart.no.isotc211.

[5] Committee on Earth Observation Satellites, CEOS/WGISS/PTT/CIB-B (1997), **Catalog Interoperability Protocol (CIP) Specification - Release B**.

[6] Gamma, Helm, Johnson, Vlissides, 1995. **Design Patterns: Elements of Reusable Object-Oriented Software,** Addison-Wesley, Reading, MA.

[7] National Imagery and Mapping Agency, 4 March 1995, **Global Geospatial Information and Services (GGIS) for the Warrior,** http://164.214.2.59/cgi-bin/waisgate?WAISdocID=087986451+0+0+0&WAISaction=retrieve.

[8] National Imagery and Mapping Agency (CIO-2061), 20 December 1996, **Common Imagery Interoperability Facilities CIIF) Reference Model, Version 2.0**. http://www-ismc.itsi.disa.mil/ciiwg/ciiwg.html

[9] International Organization of Standards (MUN/96/P2/N11), 25 October 1996, **Reference Model for an Open Archival Information Systems (OAIS), Draft Version 7.0**, http://www.gsfc.nasa.gov/nost/isoas/overview.html.

[10] **Geospatial and Imagery Access Services Specification**, Version 3.0, National Imagery and Mapping Agency, United States Imagery System, 22 July 1997.

[11] Federal Geographic Data Committee (FGDC) (April 1995), Development of a National Digital Geospatial Data Framework, Washington, DC.

[12] Cook, Steve, and John Daniels, **Designing Objects Systems: Object-Oriented Modeling with Syntropy**, Prentice Hall, New York, 1994, xx + 389 pp.

3. Abstract Model for Catalog Services

This section defines the abstract model of the OpenGIS Catalog Services. Figure 3-1 is a diagram of the primary classes used by the OpenGIS catalog services. It indicates the classes used by a geospatial discovery service, a geodata access service, and an access service for other data. In general, multiple specific services of each type will simultaneously exist, and can be inter-related by "describes" relationships as indicated.

The functions provided by each type of service are expected to include those listed in Table 3-1. The following subsections separately describe a geodata discovery service, geodata access service, and other data access service. Section 3.4 then describes some aspects common to all services and to multiple aspects of a service.

Note: This abstract model includes considerable detail, in order to adequately define and explain the service functions desired. The detail provided includes classes, class diagrams, class operations, and operation inputs and outputs. These details are not requirements on the implementation specifications. An implementation specification must provide the general service functions described, but is free to deviate from many details of the abstract model. Furthermore, proposed implementation specifications should deviate from the abstract model wherever a change will improve the service API and/or implementation. Such changes (may) include using interface classes and operations with significantly different functions (and names) than used in the abstract model. For example, the classes used in the abstract model might be either (1) visible in the interface, (2) used internally in the implementation but not visible in the interface, (3) virtual classes used to understand the interface but not directly visible, or (4) not used at all by the interface.

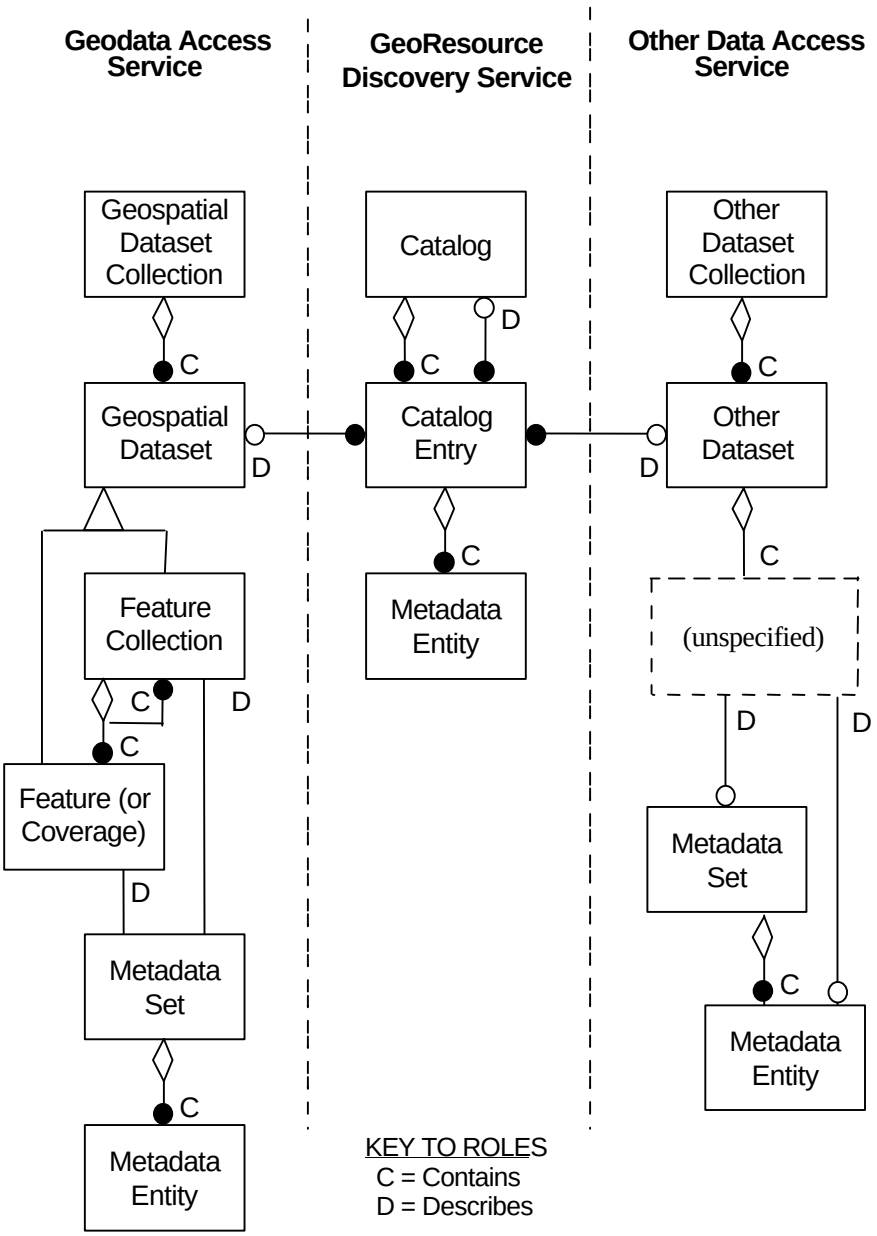

Figure 3-1. The Primary Data Structure Classes of OpenGIS Catalog Services.

Geodata Access Service	Geodata Discovery Service	Other Data Access Service
Copy complete dataset	Query catalog service	Copy complete dataset
Retrieve partial dataset	Add catalog entry	Retrieve partial dataset
Add dataset	Remove catalog entry	Add dataset
Remove dataset	Modify catalog entry	Remove dataset
Modify dataset	Copy selected catalog entry	Modify dataset
Create iterator through datasets	Create iterator through catalog entries	Create iterator through datasets
Query access service	Get catalog entry schema	Query access service
Get dataset schema	Get service properties	Get dataset schema
Get service properties	Set service properties	Get service properties
Get service property schema	Get service property schema	Get service property schema

Table 3-1. The Primary Functions Provided by Open GIS Catalog Services.

3.1. GeoResource Discovery Service

This section defines the abstract model of a geodata discovery service. This service corresponds to the Geospatial Catalog Services defined in Section 3.1.1.3 of Topic 12: The OpenGIS Service Architecture. A geodata discovery service uses repositories of metadata that describe and reference other geospatial resources. Such repositories of discovery metadata are what we refer to as catalogs. Discovery metadata is normally stored electronically in digital form, on-line to a computer system. The discovery metadata could be stored long term (e.g., for years) and/or short term (e.g., for minutes). For example, discovery metadata stored only short term might be the result of a query on another, larger geodata discovery service.

The following subsections describe the primary classes, relationships among those classes, and functions of a geodata discovery service. Possible extensions of a geodata discovery service are discussed later in Section 3.4 of this document.

The center section of Figure 3-1 indicates the primary classes used by a resource discovery service: Catalog, Catalog Entry, and Metadata Entity.

3.1.1. Catalog

A Catalog is simply a collection of Catalog Entries that is organized to assist in the discovery, access, and retrieval of geospatial resources that are of interest to the user, especially when the existence or whereabouts of the resource are not know to the user.

Being a collection of catalog entries, a catalog should be able to enumerate each of its entries, and should allow entries to be added or removed. It should also support queries that will enable the user to obtain entries of interest based on specified criteria.

3.1.1.1. Functions

3.1.1.1.1. Query Functions

The central function provided by a geodata discovery service is a query function. The query function(s) provided must find all the Catalog Entries in a catalog which meet the conditions desired by the user that issued the query. The primary input to the query function(s) is the query definition, and the primary output is the query result.

All the Catalog Entries that meet the specified conditions are returned in a query result, which is a collection of catalog entries (or a new Catalog). The query result need not contain the complete original Catalog Entries. The query should specify the subset of the relevant Catalog Entries that is to be returned in the query result. The desired subset might be specified by listing the specific types of Metadata Entities desired.

In general, a query definition specifies a Boolean function of multiple binary-valued conditions. This Boolean function of multiple conditions combines these conditions using AND, OR, and NOT operators. Each of these conditions has three parts:

1. Metadata Element name or identification, to be compared

2. Comparison value, to be compared to the value of the Metadata Element

3. Type of comparison to perform, between the comparison value and the value of the Metadata Element

The types of comparison that are allowed depend on the type of the value of the specified Metadata Element. The types of comparison allowed might include those listed in Table 3-2.

Data Type	Comparison Type (Examples)	Meaning
All	Equals	The metadata value equals the comparison value. For a floating point number, a small tolerance might be allowed.
Number	Greater than	The metadata value is greater than the comparison value.
	Less Than	The metadata value is less than the comparison value.
Text	Includes	The metadata text value includes the comparison value somewhere within it.
	Starts with	The metadata text value begins with the comparison value.
	Ends with	The metadata text value ends with the comparison value.
Geometry	Overlaps	The metadata polygon area overlaps the comparison polygon area.
	Contained in	The metadata polygon area is contained within the comparison polygon area.
	Contains	The metadata polygon area contains within it the comparison polygon area.
	Dimensionally Extended 9 Intersection Model (DE9IM) Operators	Boolean functions that are used to test for the existence of a specified topological relationship between two geometries. [1,2,3].

Table 3-2. Possible Types of Comparison in Query Conditions.

Each Metadata Element to be compared is normally contained within one of the Metadata Entity objects contained within each Catalog Entry object. The query function(s) should properly handle at least three special cases:

1. A Catalog Entry object contains multiple values of the specified Metadata Element to be compared. That is, the Catalog Entry object may contain multiple Metadata Entity classes of the (one) type which contains the specified Metadata Element. Alternately, the value of the specified Metadata Element may be a list of values. In both these cases, a Catalog Entry object should be considered to satisfy the specified condition if any contained metadata value meets that condition. That is, the multiple values should be treated like multiple similar conditions connected by OR operators.

2. A Catalog Entry object contains a null value for the specified Metadata Element to be compared. That is, the name of this Metadata Element is stored, but the associated value is null. This case assumes that some Metadata Elements can have one or more null values which are suitably specified. In this case, the proper action to take in response to a query condition should be specified with the null value(s) for that Metadata Element.

3. A Catalog Entry object does not contain a value of the specified Metadata Element to be compared. That is, the name of this Metadata Element is not stored in any Metadata Entity object contained in a Catalog Item object. In this case, the proper action to take in response to a query condition should be specified in the query. The proper action might be specified as part of the type of comparison used in the query. Alternately, the proper action to take in response to a query condition might be specified in the catalog, for a known set of Metadata Element names.

In addition, a comparison value could be allowed to be a list of values (of the same type). The multiple comparison values should be treated like multiple similar conditions connected by OR operators.

A query function required in some geodata discovery services is recording a stored query, within that geodata discovery service. Such a stored query is to be evaluated against each Catalog Entry that is subsequently added to that Catalog. A stored query might also be evaluated against each Catalog Entry that is subsequently removed from that Geodata Catalog. Whenever a new (or removed) Catalog Entry satisfies the conditions of the stored query, the user specified in that stored query must be automatically notified, in the manner specified in the inputs to this query function.

3.1.1.1.2. Other Functions on Catalog

The other needed functions that primarily deal with a Catalog are generally the functions appropriate to a collection, including:

1. Add catalog entry

2. Remove catalog entry

3. Copy selected entry from catalog

4. Modify catalog entry (optional)

5. Create iterator through catalog entries (optional)

6. Get catalog entry schema

7. Get catalog service properties (optional)

8. Set catalog service properties (optional)

9. Copy entire catalog object (optional)

10. Create catalog object

11. Destroy catalog object

These collection functions are discussed in more detail in Section 3.4.1.

3.1.1.2. Relationships

1. A Catalog object contains (in its collection) multiple Catalog Entry objects

2. A Catalog is optionally described by one or more Catalog Entries contained in other Catalogs.

3.1.2. Catalog Entry

The Catalog Entry contains all the Metadata Entities used to describe one resource. More specifically, an object of this class contains a collection of Metadata Entities which together describe the associated resource for the purpose of discovery. This collection of Metadata Entities is usually only a subset of all the metadata that exists for the associated resource.

One or more of the Metadata Entities usually directly reference the associated resource, and provide the basic information needed to locate and subsequently access that resource, either directly or through an access service. When applicable, one or more of the included Metadata Entities may specify the spatial location of the resource. The spatial location could be defined by a (minimum bounding) rectangle, and/or by a polygon bounding the ground area covered. Perhaps the spatial location could be defined by any OpenGIS geometry. For example, the spatial location of a feature describing a single point could be defined by that point.

If the resource is a dataset, several of the included Metadata Entities may define the types of data included in the dataset, plus the quality of that data. Some Metadata Entities that describe data quality are specified in Topic 9: Quality.

The entries of a Catalog can have different contents, depending on the type of dataset described. For example, some specific Metadata Entities may be optional, to be included only when applicable. Some specific Metadata Entities may be allowed to be repeated multiple times, with different data contents, when multiple values are needed or appropriate.

Being a collection of Metadata Entities, a Catalog Entry should be able to enumerate each of the components, and should allow components to be added or removed.

3.1.2.1. Functions

The needed functions that deal with individual Catalog Entries are generally the functions appropriate to a collection, including:

1. Copy catalog entry object

2. Copy selected Metadata Entity from catalog entry

3. Add Metadata Entity to catalog entry (optional)

4. Remove Metadata Entity from catalog entry (optional)

5. Create iterator through Metadata Entities(optional)

6. Get catalog entry schema

7. Create catalog entry object

8. Destroy catalog entry object

These collection functions are discussed in more detail in Section 3.4.1.

3.1.2.2. Relationships

1. A Catalog Entry object contains (in its collection) multiple Metadata Entity objects

2. A Catalog Entry object describes (and references) a geospatial resource such as a feature collection or another catalog. Multiple Catalog Entry objects can describe the same resource. These multiple Catalog Entries may be in the same and/or different Catalog instances.

3.1.3. Metadata Entity

The Metadata Entity is a logical collection of Metadata Elements. More specifically, an object of this class contains a set of logically related Metadata Elements , as specified in Topic 11: Metadata. Each Metadata Element is recorded as a name and value pair of object attributes..

Since the set of Metadata Elements depends on the metadata to be recorded, the Metadata Entity class is abstract. A concrete subclass is used for each different logical group of Metadata Elements recorded. The Metadata Entity abstract class defines one attribute included in all subclasses, a string giving the name of the specific Metadata Entity subclass. Each such name must include a version identification number. Each Metadata Entity object may be included in one or more Catalog Entry objects.

Being a collection of Metadata Elements, a Metadata Entity should be able to enumerate each of the items, and should allow components to be added, removed, or modified. For additional functional requirements see Section 3.2.5.

3.1.3.1. Functions

The needed functions that deal with individual Metadata Entities are generally the basic functions for any object, including:

1. Copy Metadata Entity

2. Modify Metadata Entity(optional)

3. Create Metadata Entity

4. Destroy Metadata Entity

3.2. Geodata Access Service

This section defines the abstract model of a geodata access service. Such services correspond to the Geospatial Information Retrieval Services plus the Geospatial Product Information Services, defined in Sections 3.1.1.1 and 3.1.1.2 of Topic 12: The OpenGIS Service Architecture. The geospatial data can be accessed or retrieved by a specific application for a wide variety of purposes, including data display, modification, deletion, and extraction of derived data.

A geodata access service provides access to geospatial data stored in an associated data repository. Such a repository could also be called a library, archive, data warehouse, or collection. The geospatial data might be stored electronically and/or in hardcopy, in digital and/or analog forms. Electronically stored data could be stored on-line or off-line to a computer system. Similarly, the geospatial data could be stored long term (e.g., for years or months) and/or short term (e.g., for hours or minutes).

The following subsections describe the primary classes, relationships among those classes, and functions of a geodata access service. Possible extensions of a geodata access service are discussed later in Section 3.4 of this document.

The left section of Figure 3-1 shows the primary classes used by a geodata access service: Geospatial Dataset Collection, Geospatial Dataset, Feature Collection, Feature, Metadata Set, and Metadata Entity. Each of these six classes is described in the following subsections.

3.2.1. Geospatial Dataset Collection

The Geospatial Dataset Collection contains all the data accessed by a geodata access service. This collection could also be called a library, archive, or data warehouse. An object of this class contains a collection of Geospatial Datasets. This collection of Geospatial Datasets is stored in a manner that permits accessing or retrieving any specified dataset when requested.

3.2.1.1. Functions

3.2.1.1.1. Retrieve Dataset

The central functions provided by a geodata access service are data retrieval functions. A specified geospatial dataset will be retrieved, as selected by the (person or machine) user that issues the retrieval request. Either a complete dataset or a specified subset can be requested. If separate functions are provided for retrieval of partial and complete datasets, these functions might be called:

1. Copy complete dataset – The output from this function is a copy of a complete dataset, as stored by this geodata access service. The input to this function is the identification of the desired dataset.

2. Retrieve partial dataset – The output from this function is a copy of the selected portions of a dataset, as stored by this geodata access service. The inputs to this function are the identification of the desired dataset, plus identifications of the dataset parts to be copied.

For the properties of a Feature or Feature Collection, the desired subset might be specified by listing the names of the desired properties. For the metadata included in a dataset, the desired subset might be specified by listing the specific types of Metadata EntityMetadata Entities desired. For a Feature Collection, the desired subset could be specified by listing the specific features desired.

3.2.1.1.2. Other Functions on Geospatial Dataset Collection

Most other needed functions that primarily deal with a Geospatial Dataset Collection are the functions appropriate to a collection, including:

1. Add dataset to collection

2. Remove dataset from collection

3. Modify dataset in collection (optional)

4. Create iterator through datasets in collection (optional)

5. Get dataset schema

6. Get dataset collection properties (optional)

7. Set dataset collection properties (optional)

8. Create dataset collection object

9. Destroy dataset collection object

These collection functions are discussed in more detail in Section 3.4.1.

3.2.1.1.3. Query Functions

In addition to all the functions defined above on a Geospatial Dataset Collection, query functions could be provided. These query function(s) would be similar to those defined in Section 3.1.1.1 for a Catalog. However, these function(s) would operate on the Geospatial Dataset Collection. More specifically, these query functions would operate on the Metadata Set objects associated with the contained Geospatial Datasets.

A desired subset of Geospatial Datasets can be specified by defining the ground area of interest. The ground area of interest should be specified by a polygon, either a rectangle or a more general polygon. All features in a Feature Collection that overlap the specified area of interest should be

retrieved. Only that portion of a Coverage (type of Feature) which covers the specified area of interest should be retrieved.

The query function(s) provided would find all the Geospatial Datasets in a collection which meet the conditions desired by the (person or machine) user that issued the query. Selected metadata for all the datasets that meet the specified conditions is returned in a query result, which is a collection of Metadata Sets (or a Catalog). The query result need not contain the complete metadata for each relevant dataset. The query should specify the subset of the metadata of the datasets that is to be returned in the query result. The desired subset might be specified by listing the specific types of Metadata Entities desired.

Most other aspects of a query function on a Geospatial Dataset Collection would be analogous to the corresponding aspects of a Query on a Catalog, as previously discussed in Section 3.1.1.1.1.

3.2.1.2. Relationships

1. A Geospatial Dataset Collection object contains (in its collection) multiple Geospatial Dataset objects, in the same geodata access service.

3.2.2. Geospatial Dataset

The Geospatial Dataset contains all the data in one set of geospatial data, including the associated metadata. More specifically, an object of this class is usually an OpenGIS Feature Collection, but can be an individual Feature (or Coverage).

3.2.2.1. Functions

In addition to the functions discussed above for Feature Collections and for Features, it will sometimes be necessary to control and test the current accessibility status of a dataset. For example, modifying or creating a dataset may take some time, and during that time it is likely that users other than the one modifying the dataset must be excluded from accessing that dataset. The functions provided on the accessibility status of a selected dataset might include the functions listed and described in Table 3-3.

Function	Function Description	Function Inputs	Function Outputs
Check dataset status	Check if the current dataset status is desired status	dataset identification, desired dataset status (might be a list)	correct status flag (Boolean)
Modify dataset status	Modify current dataset status in specified manner, if specified change is permitted	dataset identification, desired status change	modify successful flag (Boolean), new dataset status (optional)
Get dataset status (optional)	Retrieve current dataset status	dataset identification	dataset status

Table 3-3. Expected and Optional Functions on Dataset Accessibility.

The function outputs listed in the above table exclude the exception condition outputs that are may be needed.

3.2.2.2. Relationships

1. A Geospatial Dataset object is either a Feature Collection object or a Feature object. A Coverage object is a special type of Feature object.

2. A Geospatial Dataset object is often described (and referenced) by one or more Catalog Entry objects, in one or more Catalog objects.

3.2.3. Feature Collection

The Feature Collection contains all the data in one OpenGIS feature collection, as specified in Topic 10: Feature Collections and Topic 5: The OpenGIS Feature. This Feature Collection is a specific subclass of the Geospatial Dataset class. The data recorded for a Feature Collection includes a list of the properties of that collection. More specifically, an object of this class contains

a set of object attributes that store the values of a group of collection properties. Each property is recorded as a name and value pair of object attributes.

Each object of this Feature Collection class is related to one Metadata Set object. This relationship is required, and is recorded by a mandatory property of a Feature Collection with the property name Metadata. The value of the metadata property is the ID of the associated Metadata Set object. If this property value is null, there is no related Metadata Set object, and all metadata is assumed to be unknown or unspecified, except for any metadata associated with the individual features contained in that collection.

3.2.3.1. Functions

The needed functions that deal with a Feature Collection are generally the functions appropriate to a collection, including:

1. Copy selected feature from collection

2. Create iterator through features in collection

3. Get feature collection schema

4. Get feature collection properties

5. Set feature collection properties (optional)

6. Add feature to collection

7. Remove feature from collection

8. Modify feature in collection (optional)

9. Copy feature collection object (optional)

10. Create feature collection object

11. Destroy feature collection object

These collection functions are discussed in more detail in Section 3.4.1. Implementation specifications for most of these functions on feature collections are already defined in the responses to OpenGIS RFP 1.

3.2.3.2. Relationships

1. A Feature Collection object contains (in its collection) multiple Feature objects and/or other Feature Collection objects, in the same geodata access service.

2. A Feature Collection object is normally described by a Metadata Set object, in the same Geospatial Dataset object.

3.2.4. Feature

The Feature contains all the data in one OpenGIS feature or coverage, as specified in Topic 5: The OpenGIS Feature and Topic 6: The Coverage Type and its Subtypes. (An OpenGIS coverage is considered to be a particular type of feature.) This Feature class is a specific subclass of the Geospatial Dataset class. The data recorded for a Feature is a list of properties of that feature. More specifically, an object of this class contains a set of object attributes that store the values of a group of feature properties. Each property is recorded as a name and value pair of object attributes. One or more of these feature properties can be an OpenGIS geometry or coverage.

Each object of this Feature class can be related to one Metadata Set object. When a single feature is a Geospatial Dataset, this relationship would be normal and might be required. (When a Feature Collection is a Geospatial Dataset, this relationship is optional, included only when metadata is recorded for that individual Feature.) This relationship is recorded by a mandatory property of a Feature with the property name Metadata. The value of the metadata property is normally the ID of the associated Metadata Set object. If this property value is null, there is no related Metadata Set object, and all metadata not associated with the Feature Collection that contains this Feature is assumed to be unknown or unspecified.

3.2.4.1. Functions

The needed functions that deal with individual Features include the basic functions for any object, including:

1. Copy feature

2. Modify feature (optional)

3. Create feature

4. Destroy feature

Implementation specifications for most of these functions on features are already defined in the to OpenGIS Simple Features implementation specifications.

3.2.4.2. Relationships

1. A Feature object can be described by a Metadata Set object, in the same Geospatial Dataset object. An individual Feature object that is a Geospatial Dataset object is normally described by a Metadata Set object.

3.2.5. Metadata Set

The Metadata Set contains all the metadata associated with a Feature Collection or an individual Feature, as specified in Topic 11: Metadata. This metadata can include quality metadata, as specified in Topic 9: Quality. A Metadata Set object contains, or is related to, a collection of Metadata Entity objects. These relationships are recorded in the Metadata Set object, by using an object attribute that contains a sequence of the IDs of all the contained Metadata Entity objects.

3.2.5.1. Functions

The needed functions that deal with individual Metadata Sets are generally the functions appropriate to a collection, including:

1. Copy Metadata Set object

2. Copy selected Metadata Entityfrom Metadata Set

3. Add Metadata Entity to Metadata Set

4. Remove Metadata Entityfrom Metadata Set (optional)

5. Create iterator through Metadata Entities(optional)

6. Get Metadata Set schema

7. Create Metadata Set object

8. Destroy Metadata Set object

These collection functions are discussed in more detail in Section 3.4.1.

3.2.5.2. Relationships

1. A Metadata Set object contains (in its collection) multiple Metadata Entity objects, in the same Geospatial Dataset object.

3.2.6. Metadata Entity

The Metadata Entity contains the values of a logical group of Metadata Elements, as specified in Topic 11: Metadata. More specifically, an object of this class contains a set of object attributes that store the values of a logical group of Metadata Elements. Each Metadata Element is recorded as a name and value pair of object attributes.

Since the set of Metadata Elements depends on the metadata to be recorded, the Metadata Entity class is abstract. A concrete subclass is used for each different logical group of Metadata Elements recorded. The Metadata Entity abstract class defines one attribute included in all subclasses, a string giving the name of the specific Metadata Entity subclass. Each such name must include a

version identification number. Each Metadata Entity object is contained in one or more Metadata Set objects, but those relationships need not be recorded within a Metadata Entity object.

3.2.6.1. Functions

The needed functions that deal with individual Metadata Entities are generally the basic functions for any object, including:

1. Copy Metadata Entity

2. Modify Metadata Entity(optional)

3. Create Metadata Entity

4. Destroy Metadata Entity

3.3. Other Data Access Service

The “Other Data Access Service” is included on the right side of Figure 3-1 to indicate that a geodata discovery service can include references to datasets which do not contain geospatial data (in the usual OpenGIS sense). That is, a geodata discovery service can contain Catalog Entries that describe and reference non-geospatial datasets, which can be accessed from another type of data access service.

Many of the classes and service functions of an Other Data Access Service are expected to be directly analogous to the corresponding items discussed above for a Geodata Access Service. These classes and service functions are thus not specifically discussed herein. Such an Other Data Access Service may include additional classes, within the dashed box shown in Figure 3-1. This service could also provide additional service functions, appropriate to the type(s) of data accessed.

3.4. Other Functions

This section describes some aspects common to all services and to multiple aspects of a service, namely:

1. Functions on Collections

2. Functions for Large Query Results

3. Functions for Delayed Responses

4. Functions for User Collaboration

5. Functions for Services Collaboration

6. Data Form Transformation

3.4.1. Functions on Collections

This section provides more information about the functions provided by the collections used in the OpenGIS Catalog Services, including Catalog, Catalog Entry, Geospatial Dataset Collection, Feature Collection, and Metadata Set.

In general, a collection contains many items, of the type contained by that collection. A collection can contain only one member item, although this is expected to be rare. A collection might contain no items while it is in the process of creation or modification.

The functions provided to manipulate a collection object are expected to include those listed and described in Table 3-4. Some of the functions listed are optional, as indicated in the left column. For example, the optional “Get collection properties“ and “Set collection properties” functions are optional. One or both of these functions would be needed if this collection object contains any properties of the collection, separate from the properties of the individual items contained in the collection. Such collection properties could include the number of items currently contained in the collection. Such collection properties should include definitions of all the externally visible service functions supported by the collection object, and all the data elements used as inputs and outputs of those functions.

The function outputs listed in the figure exclude the status and exception condition outputs that are usually needed. Some of the listed inputs and outputs are assumed to be:

1. References to existing objects

2. New objects, or references to such objects

3. Data in the form of name and value lists

Function	Function Description	Function Inputs	Function Outputs
Add item to collection	Add new item to collection, at end of current collection list	item object	(none)
Remove item from collection	Remove one specified item from current collection list, and destroy that item object	item object reference, or partial item properties name and value list	(none)
Modify item in collection (optional)	Modify contents of one specified item in current collection list	item object reference, modified item properties name and value list	(none)
Copy selected item from collection	Create and return exact copy of one specified item object from current collection list	item object reference, or partial item properties name and value list	item object
Create iterator through collection (optional)	Create and return iterator object for all items in current collection list	(none)	new iterator object
Copy collection object (optional)	Create and return exact copy of this entire collection object, including all contained item objects	(none)	new collection object
Get item schema (or Get item data definitions)	Get schema of specified item in current collection list, including definitions of all data elements in that item and all functions supported by that item object	item object reference, data element names list (optional)	data dictionary name and value list
Get collection properties (optional)	Retrieve specified properties of this collection object	collection properties names list	collection properties name and value list
Set collection properties (optional)	Modify specified properties of this collection object	collection properties name and value list	(none)
Create collection object	Create new collection object of selected type. A separate factory object can be used if needed.	initial collection items list (optional), collection properties name and value list (optional)	new collection object
Destroy collection object	Delete this collection object	(none)	(none)

Table 3-4. Expected and Optional Functions on a Collection Object.

The functions provided to manipulate a collection object are expected to often include the ability to sequentially access all items in that collection. The “Create iterator through collection” function listed in Table 3-4 would provide this capability. This function would return a new iterator object, that could initially point to the first item in the associated collection list. This iterator object might provide the functions listed and described in Table 3-5.

Function	Function Description	Function Inputs	Function Outputs
Get next item in collection	Retrieve next item in collection list, after last item previously retrieved	(none)	item object
End of collection? (optional)	Check if there no more items in collection list, after last item previously retrieved	(none)	end of collection flag (Boolean)
Get current item in collection (optional)	Retrieve again last item previously retrieved	(none)	item object
Get first item in collection (optional)	Retrieve first item in collection list	(none)	item object
Destroy iterator object	Delete this iterator object	(none)	(none)

Table 3-5. Expected and Optional Functions on an Iterator Object.

3.4.2. Functions for Large Query Results

The preceding abstract model discussions assume a query result is sufficiently small that the entire query result is always sent to the user that submitted the query. However, a Catalog can contain a very large number of catalog entries that satisfy the specified query conditions. It can be impractical for the:

1. Discovery or access service to produce such a large output,

2. Communication system to transfer such a large data set, and/or

3. User system to receive such a large output.

One possible way to handle large query results is as follows. The queried service initially outputs only a limited number of the collection items which satisfy the query, with an indication that more result items are available. The maximum number of collection items returned initially could be fixed in the service. (For example, the maximum number could be specified by a collection property.) Alternately, the maximum number of collection items returned initially could be specified in the query. (That is, the maximum number would be specified by an input to the query function.)

When the user is ready to receive more query result items, a separate query function would be called to return more of the collection items that satisfy the previous query. Again, only a limited number of the collection items that satisfy the initial query would be output. The maximum number of collection items returned could be the same as for the initial output, or could be specified in the same manner.

If and when the user decides not to retrieve any more query response items, a separate query function should be called to discard any remaining query result items. This function would free any computer storage used by the query result items, including any storage of information about the original query.

3.4.3. Functions for Delayed Responses

The preceding abstract model discussions generally assume that a geodata service is able to respond quickly, while the (human or machine) user waits for the response to a function request. However, Catalogs, Geospatial Dataset Collections, and Geospatial Datasets will often be so large that quick responses are not possible. In this case, provision must be made for the user to proceed with other activities while the request is being processed. The user will later receive the results of the request.

The service functions needing delayed response capabilities may include most of the functions listed in Table 3-1 and described in Sections 3.1, 3.2 and 3.3. The functions most likely to need delayed responses include those listed in Table 3-6.

Geodata Access Service	Geodata Discovery Service	Other Data Access Service
Retrieve partial dataset Copy complete dataset Add dataset Remove dataset Modify dataset Query access service	Query catalog service Add catalog entry Remove catalog entry Modify catalog entry	Retrieve partial dataset Copy complete dataset Add dataset Remove dataset Modify dataset Query access service

Table 3-6. Service Functions Likely to Need Delayed Response Capabilities.

One possible way to handle delayed responses to service requests is for the output from the function request to be only a reference to a request object. A request object is maintained by the service that received the request. That is, the data normally produced by the request is not included as a direct output from the request function. The request object provides functions that allow the user to check the status of the request and to receive the request results (if any) when they are available. Such request objects might provide the functions listed and described in Table 3-7.

Function	Function Description	Function Inputs	Function Outputs
Get request status	Retrieve current status of this request	(none)	request status
Cancel request	Cancel this request, and destroy this request object	(none)	(none)
Get request results	Retrieve results of this request, waiting if needed for request processing to be completed. When done, destroy this request object.	maximum waiting time (optional)	request results (depending on request type)
Check request status (optional)	Check if request results are now available	(none)	results available flag (Boolean)
Modify request (optional)	Change existing request in specified manner, if these changes are allowed	request changes name and value list	(none)

Table 3-7. Expected and Optional Functions on a Request Object.

The function outputs listed in the above table exclude the exception condition outputs that are may be needed.

3.4.4. Functions for User Collaboration

The preceding abstract model discussions generally assume that users operate independently. That is, the result of a service request is returned to the one user who made that request. However, a group of users sometimes need to collaborate and share work. For example, the result from a request may be needed by a user other than the one who submitted the request. Also, the result from a request may be needed by multiple users, not just by the one who submitted the request.

The OpenGIS discovery and access services should facilitate user collaboration by allowing most function requests to optionally include a list of one or more users to which the request results are to be sent. Such a list of users need not include the user who issued the request. In the absence of this user list in a request, the request result is sent to the one user who issued that request.

3.4.5. Functions for Services Collaboration

The preceding abstract model discussions generally assume that similar services operate independently. That is, a user must send a request to all accessible services that might be able to satisfy a request. To simplify use, a set of similar services should be able to collaborate, so that a user could directly send a request to only one service in the set.

One possible way for geodata discovery services to collaborate is for a Catalog to include one or more Catalog Entries that describe other Catalogs (and the corresponding geodata access services). A catalog of catalogs could contain only Catalog Entries that describe other Catalogs. Alternately, one catalog could contain Catalog Entries that describe both Geospatial Datasets and other Catalogs.

When a catalog contains Catalog Entries that describe other Catalogs, query functions might work somewhat differently on such Catalog Entries. For example, query conditions may need to be

implemented differently when a Catalog Entry describes another Catalog, that contains many Catalog Entries. A Catalog Entry that describes another Catalog will usually have multiple values of multiple Metadata Elements. That is, the Catalog Entry collection is likely to contain many Metadata Entities of the same type, recording multiple values of the same Metadata Elements.

In addition, when a Catalog Entry that describes another Catalog matches the conditions of a query, that query function should also be performed on the referenced Catalog. The (human or machine) user who issued this query could query each other Catalog referenced in the query response. However, the burden on the user would be reduced if the first geodata discovery service would automatically query each other Catalog referenced in a query response. The first geodata discovery service should receive the query responses from the other Catalogs queried, and automatically consolidate all the query responses. For example, this consolidation should eliminate or combine Catalog Entries that reference the same dataset.

Another possible form of collaboration between geodata discovery services is to copy appropriate Catalog Entries. For example, one geodata discovery service (or its librarian) could query another such service to obtain copies of all its Catalog Entries that would help fulfill the purpose of the first geodata discovery service. Such a query might be automatically issued periodically by the first discovery service. Alternately, the first discovery service could issue a stored query to each collaborating discovery service. In this case, a query response may be desirable when a Catalog Entry is removed that meets the conditions of the stored query.

Collaboration between separate geodata access services may also be needed. If these access services support query functions, those services may need to collaborate as discussed above for geodata discovery services. A Geospatial Dataset contained in one access service might be primarily a reference to the actual dataset that is stored by one or more other access services. In this case, requests for all (or most) functions which reference that dataset should be forwarded to one of the access services which actually store that dataset.

3.4.6. Discovery and Access of Services

Although most of this document discusses discovery and access of datasets, discovery and access of other services are also of use. That is, services for discovery of and access to other geospatial services are also needed by many users. For ease of use, these other-service access services should operate similar to the dataset discovery and access services discussed above. Even better, other-service access services should be combined with the dataset access services to offer general resource discovery and access services.

Compared to datasets, other-services have some differences that affect resource discovery and access. For example, other-service access functions, schema, and metadata replace dataset access functions, schema, and metadata. The other-service access functions, schema, and metadata will often use significantly different properties and input/output arguments.

3.4.7. Data Form Transformation

The preceding abstract model discussions generally assume that each user wants the data in the same form as it is stored by a geodata discovery service or geodata access service. However in many cases, the user will not be directly able to request a service and/or to use a response in the forms that are implemented by the service. The types of data transformation that may be needed include:

1. Data format conversions, of data item values and names

2. Coordinate transformations, of spatial position data

3. Semantic translation, of data item meanings

Many of the needed data format conversions are expected to be relatively easy. For example, different names for the same data (with the same meaning) could be handled by using tables of the corresponding names, as used in different data formats or by different geographic information communities. The needed coordinate transformations can be more complex, but still fairly straightforward. Coordinate transformations could be performed by coordinate transformation services being separately defined, as specified in Section 3.4 of Topic 12: The OpenGIS Service Architecture. Semantic translation is much more difficult, and the OGC is not expected to soon deal with this issue.

One possible way to handle data transformation is to use interface software that operates between the OpenGIS Catalog Services software and the user application software, as indicated in Figure 3-2. In this diagram, the arrows represent data transfers between software components. (This is not a class or object diagram.) Multiple variations and/or copies of all the software elements shown as boxes are expected to exist simultaneously.

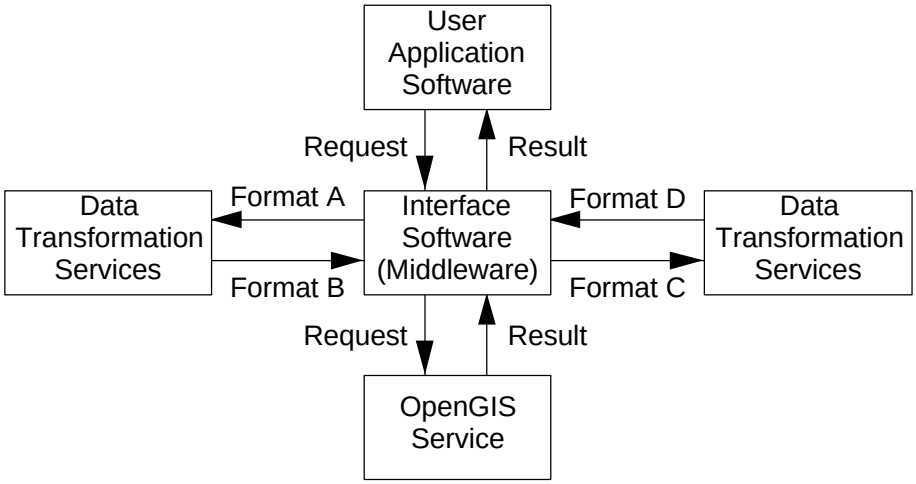

Figure 3- . Interface Software Connects User Applications to Services.

The interface software, or middleware, performs any needed data transformation on each request

calls any data transformation services software needed to transform request data from one format (or form) to another. The needed data transformation services are shown on the left side of the

Similarly, the middleware also performs any needed data transformation on each result going from the service software to the user application software. That is, this middleware calls any data

another. The needed data transformation services are shown on the right side of the diagram.

The middleware could be thought about in several different ways. The middleware could be treated

shown in the above diagram. Alternately, the middleware could be treated as part of, or an extension of, any of the other three types of software. For some purposes, the middleware would

middleware should provide to the user application software almost the same Application Programming Interfaces (APIs), as are provided by the underlying OpenGIS services.

References for Section 3

OpenGIS Features for ODBC (SQL) Implementation Specification, Open GIS Consortium,

[2]

Massachusetts, 1997.

[3] OpenGIS Features for CORBA Implementation Specification, Open GIS Consortium, Wayland, Massachusetts, 1997.

4. Future Work

Future work is dependent on the completion of OGC Request 6, a request for proposal for OpenGIS™ Catalog Interfaces available at < http://www.opengis.org/techno/request.htm>.

5. Well Known Structures

No Well Known Structures for the geodata discovery and access services have been defined.

Appendix A. Glossary

Access The ability to access stored data, usually a Geospatial Dataset in the context of

Catalog A service used for discovering geospatial resources. The catalog indexes

specified characteristics. A catalog contains a collection of multiple Catalog Entries, that each describes and references one resource. This collection of

all the stored catalog entries that meet specified conditions.

Catalog entry A object stored in a catalog that describes and references a resource, usually not stored in that catalog. A Catalog Entry is usually a subset of the complete

Collection A set of related objects that is assembled for some purpose.

A recorded descriptor or attribute of a collection as a whole, not

Dataset.

context of this topic volume.

Discovery The ability to search for (or query) and find data that has desired

topic volume.

Feature property

(considered to be a type of feature).

Geodata al data.

Geospatial dataset

geospatial data, including the associated metadata.

Iterator A data structure (class or object) that supports sequential access to all the items contained in a collection.

Data describing the content, quality, condition, format, and/or other

Metadata Set A data structure (class or object) that contains all the metadata associated with

15.

Metadata Element

Metadata Entity A data structure (class or object) that contains the values of a logical group of

classified. See ISO TC211 WG3 15046-Part 15.

Property

or an individual Feature.

Query dual elements, where all those elements are checked to find those elements which satisfy

Geospatial Resource See Section 2.3.6

Service property A recorded descriptor or attribute of a ser
the individual items contained in the multiple collections used by that service.

7. Appendix B. Use Cases

7.1. Information Consumer Use Case: Catalog Federation

7.1.1. Proposed Application Domain

Use of cartographic and topographic models in a geographic data base to improve vegetation/land cover classification derived from satellite imagery for park management is presented as a scenario. The objective is to discover, evaluate and retrieve Landsat multispectral imagery together with USGS 1 degree Digital Elevation Models and USGS 1:100K Transportation Digital Line Graphs to build a land use/land cover classification for Morris County, New Jersey.

7.1.2. Background

Computer-aided classification of digital satellite data has been shown to be an effective tool for conducting land resource inventories. Acceptable accuracies for lower levels in land use/land cover classification hierarchies have been obtained using satellite imagery acquired during different seasons. However, in some situations such as in mountainous terrain, seasonally variable sun angles produce different intensities of shadowing that can obscure otherwise clear distinctions between land cover classes.

Previous studies have used a GIS approach to land use/land cover classification in which satellite imagery and topographic data in digital formats are geographically referenced to a common base so that a digital elevation model and derivative layers (e.g., slope and aspect) can be used to refine classification results. Moreover, digital line graphs yield other useful information about cultural features, e.g., transportation network, and political boundaries, e.g., for masking, not easily derived from imagery or digital elevation models.

7.1.3. Problem

The GIS approach above requires satellite imagery, topographic information and other cartographic data. New Landsat 7 Enhanced Thematic Mapper (ETM+) data will soon become available from the USGS EROS Data Center, as are USGS 1 degree and 7.5 minute Digital Elevation Models (DEMs), and USGS 1:100K Transportation DLGs. However, the catalogued metadata for these data types implement different standardized metadata schema. The ETM+ collection is managed by the EOSDIS Core System (ECS) which organizes its metadata compliant with the ECS Metadata Standard; while both the USGS DEM and DLG metadata are catalogued compliant with the FGDC Content Standard for Digital Geospatial Metadata. Moreover, the ETM+ imagery is of type raster stored in HDF-EOS containers, the DEM data are type grid stored in DEM format (as flat ASCII files), and the DLG data are of type vector stored in SDTF-VP (Spatial Data Transfer Standard-Vector Profile) distributions. In addition, the DLG data are attributed with the DLG-3 feature schema.

Many county-level planning applications require only a subset of the Landsat imagery to provide coverage for an area of interest. So there is also the need to efficiently discover all of the current image, topographic and cartographic data available for an area, browse the metadata for these data to determine their usefulness, and retrieve only the highest quality data required to cover the study area but in a form, and with sufficiently accurate georegistration, so that they can be used together by a classifier.

7.1.4. Data

This scenario involves *Landsat 7 ETM+ imagery*, *USGS 1 degree DEM*, and *USGS 1:100K Transportation DLG* data. These data repositories, and the catalogs containing metadata that reference them, are located on different network servers (but this is hidden from the planner). That is to say, these catalogs and data repositories are extremely heterogeneous and distributed on the Internet, but linked logically in an earth science information federation.

7.1.5. Actors

Two actors are identified for this scenario. The primary actor is the *Planning Engineer* who wants to use the data described above to generate a current land use/land cover thematic map for Morris County, NJ. The Engineer must locate these data, determine their suitability for classification purposes, and retrieve a useful subset of these ready for ingest by local geoprocessing software applications. Secondary actors in this scenario are the *Data Providers* who are responsible for

archiving Landsat ETM+, DEM and DLG data on network servers, and for updating digital catalogs with metadata that reference these new data as they are added to their archives.

7.1.6. Use Case

A Planning Engineer in the Morris County Park Commission wants to create a USGS Level I land use/land cover classification for Morris County, NJ [1]. Currently, the Park Commission has only historical black-and-white aerial photography for the county, so this engineer wishes to locate all available multispectral satellite imagery that covers Morris County. Since the county contains areas of relatively steep relief in the northwest and southeast, this engineer believes that digital elevation data for the same area could be used to improve the classification of image pixels obtained from an unsupervised classification. This engineer has ISDN access to the Internet and a workstation with a WWW browser, and so wants to use the Web to obtain the satellite imagery and topographic data. Since Morris County forms part of the New York Metropolitan region and is heavily trafficked, he also wants to acquire county boundary and transportation features to overlay on his land use/land cover layer. Once these data are obtained, the engineer can better determine the amount of the forested park land managed by the MCPC that is located in mountainous areas.

Because the engineer doesn't have handy a map from which to determine coordinates, he begins his search for useful data by drawing a bounding box around a boundary line graph of Morris County presented in his client. Since he is also interested in identifying any data objects whose content relates to "New Jersey" and the purpose of "land cover classification," he enters these character strings in a form provided by his client. The bounding box is used to query the footprint metadata attributes, the placename is used to query placename keyword metadata attributes, and the purpose character string is for querying full-text indices, of all geospatial catalogs in the earth science data federation.

An Object ID is returned for each geospatial data object whose footprint intersects the one drawn by the engineer or whose content is related to both New Jersey and land cover classification. In this scenario, four OIDs are listed: two ETM+ objects, a DEM object, and a DLG object.

The engineer browses metadata for one of the ETM+ objects and discovers that it contains 90% cloud cover. While browsing metadata for the other, he learns that only 20% of it is covered by clouds. He inspects a browse image for this second ETM+ object and finds that most of the clouds are located outside the area containing Morris County.

Satisfied that the second ETM+ image is useful for making his classification, he creates an order to extract bands 3, 4 5 and PAN but only those rasters needed to fill his bounding box. This ETM+ data access order is stored in a "shopping cart" object.

In a similar manner, the engineer browses metadata for the DEM and DLG objects returned from his catalog query to confirm that they also cover his area of interest. In the case of DLG objects, the user is also presented with a list of DLG-3 features that are supported by data in the object, and he selects those features for which he wants data. Then he creates orders to extract data from these, again storing with each order his bounding box coordinates so that he obtains only the data required for his application. The DEM and DLG data access orders are also added to the shopping cart.

The ETM+, DEM and DLG data access information stored in the shopping cart is sent to networked Data Access

Servers. The three extracts are retrieved and stored locally for use by the engineer's image classifier and GIS.

7.1.7. References

[1] Anderson, J.R., E. Hardy, J. Roach, and R. Witmer. 1976. A Land Use and Land Cover Classification System for Use with Remote Sensor Data. U.S. Geological Survey Professional Paper 964.

7.2. Data Provider Use Case : Catalog Population and Usage

For over 25 years, the earth science community has attempted to take raw satellite sensor data collected for the earth's environs and turn these data into useful information (i.e., geophysical products), such as the radiance properties of the earth's surface. Since the middle part of the 1970s, most satellite data products that have been available from different information communities (e.g., NOAA and NASA, certain research departments of universities, and private vendors) have been raw or slightly processed sensor data. With the advent of more powerful computers and the

maturation of the science of how to generate geophysical products from raw sensor data, different

make these available on a routine basis to almost any user.

During the same 25 year period, new satellite sensors with higher spectral and spatial resolution have become operational in an attempt to provide more information about the earth's surface. In almost all cases, this has resulted in a large of volume of raw data being generated from these newer sensors. Different data providers will handle their product generation of the sensor data in different ways. Some will process all of the raw data they ingest into various types of geophysical products, others will process the raw data into geophysical products only when requested by their user community. How a particular data provider handles this situation will depend on many factors, and it usually results in a tradeoff study being done to determine which alternative to implement.

For this use case, production of any geophysical products is only undertaken by the data provider after a user has made a formal request for product generation. The use case is constructed from the point after the request has been accepted by the data provider and is acting on creating the geophysical products to fulfill the request. The use case details the sequence of events that would occur after the raw process sensor data has been ingested into the data provider's system and the newly created are "advertised" in the provider's product holdings database (i.e., catalog).

7.2.1. Actors and System Components

The primary actor for this use case is the human operator of the data provider's production system (referred to here as the production system operator). Secondary actors are external systems that provide the following inputs; 1) the raw sensor data to the data provider's ingest system, and 2) ancillary data from an external system that the data provider's system needs as an input into the geophysical product being generated. Another secondary actor is a data product specialist who assists the production system operator (POS) by doing the initial quality assurance (QA) of the newly created products.

The major subsystems of the data provider's overall system that the POS interacts with are the archive subsystem, and the data production subsystem. The archive subsystem, (referred to here as the data server[DS]), has several major functions which are the following: 1) maintain the catalog of the data provider's product holdings, 2) archive both the raw data ingested and the products generated for a certain time period, 3) software tools to "crack open" the ingested raw data and product data to extract the necessary metadata to populate its product's catalog, 4) stage data to a working storage area so that other subsystems can "pull and push" data to and from this storage area, and 5) distribute requested products to the appropriate user(s).

The data production subsystem (DPS) , has the following functions: 1) provides the POS and the data specialist with the software tools to plan and execute a production run, and do the initial QA once the products are generated, 2) interacts with the other subsystems such as the DS , to push and pull data from these subsystems(e.g., DS) to its own local storage disks, and 3) generate the geophysical products.

7.2.2. Use Case

The POS receives notification from the ingest subsystem that a new data tape containing up to 12 Advanced Spaceborne Thermal Emission and Reflection Radiometer (ASTER) scenes have been ingested into the DS. He queries the DPS planning data base (PDB) to see if any product requests are pending against the new scenes. He sees that there are several request for surface radiance products and he submits a query to the DS product holdings catalog to retrieve the following information about the newly arrived data sets:

- the scene's cloud coverage rating,
- overall quality rating, and
- geographical extent of the scene.

The production operator also begins examining the PDB to see what ancillary data are needed to generate the surface radiance products. In reviewing the PDB, he sees that several ancillary data sets are needed and he constructs another query and submits it to the DS to see the availability of these data sets.

The DS responds to the operator's initial query by returning the metadata attributes from its catalog that inform the POS about the information he initially queried on. An examination of those query

radiance products. Two of the 12 did not pass the criteria of less then 20 percent cloud cover for

then interacts with the DPS to have it retrieve the 10 scenes to the DPS's local storage disks.

The POS receives the results of his second query concerning the availability of ancillary data sets needed for the production process. The DS has many of the data sets in its archive, but has submitted it own query to an external system's catalog to see if they have an aerosol climatology data set needed to process one of the 10 ASTER scenes. It has received notification back from the

shipped to it. The POS issues a request to the DPS for it to go out and pull back to its local disks, the ancillary data needed for processing. The DPS initiates a ftp session with the DS and retrieves

By this time, the DS has received the aerosol climatology into its working storage area. The DS

climatology data set as being part of its holdings. The DS then notifies the POS that the requested ancillary data product is now available on its working storage. The POS again issues a request to

Satisfied that all the necessary "ingredients" are available for generating the surface radiance products from the raw ASTER data sets, the POS initiates a production run on the DPS. Once the production run is complete, the data specialist is notified via email, and she begins the initial QA

for obvious bad processing results such as incomplete products or no product generation at all.

earth scientist whose algorithm was used to generate the radiance products in case he wants to do a

Upon the receipt of the radiance products on its working storage, the DS again uses part of suite of software tools to extract from the products, the necessary metadata to update its catalog holdings

requester of the surface radiance products that they are available to him for a period of 10 days on