Planetary Data System

Search Service

Software Requirements and Design Document (SRD/SDD)

Sean Kelly
Sean Hardman

June 12, 2011
Version 0.4

Jet Propulsion Laboratory
Pasadena, California

CHANGE LOG

Revision	Date	Description	Author
0.1	2010-03-17	Initial draft.	S. Kelly
0.2	2010-07-12	Added content to the Analysis and Implementation sections.	S. Kelly
0.3	2011-01-20	Added PDS examples for the different query types. Reworked the Catalog Metadata use case and associated requirements from a push to a pull scenario. Added a requirement for metrics capture. Added a couple of high-level architecture diagrams. Added data model entity descriptions.	S. Hardman, S. Kelly
0.4	2011-06-12	Updated controlling document references, modified use cases and requirements based on SDWG comments and updated the architecture section to reflect the current design.	S. Hardman

TABLE OF CONTENTS

1.0 INTRODUCTION

The PDS 2010 effort will overhaul the PDS data architecture (including, but not limited to, the data model, data structures, data dictionary, data etc.) and deploy a software system (including, but not limited to, data services, distributed data catalog, data etc.) that fully embraces the PDS federation as an integrated system while taking advantage of modern innovations in information technology (including, but not limited to, networking capabilities, processing speeds, and software breakthroughs).

This service provides the function of searching cataloged metadata.

1.1 Document Scope and Purpose

This document addresses the use cases, architecture, analysis, design, and implementation of the Search service within the PDS 2010 data system. This document is intended for the reviewer of the service as well as the developer of the service (for implementation) and tester of the service (for quality assurance).

1.2 Method

This combined Software Requirements and Software Design Document (hereafter, SRD/SDD) represents the Search service software by defining use cases (a form of requirements capture), requirements list (a second form of requirements capture), and by using architecture diagrams, functional descriptions, context diagrams, and data flow diagrams for the high-level design. The detailed design will be illustrated using UML diagrams.

1.3 Notation

Although use cases are presented in this document, a separate list of requirements in the form of declarative statements is also present in this document. These declarative statements are identified with a paragraph encoding scheme "**L***i.n.j*", where:

- L is the literal letter L (ell) signifying "level".
- *i* is a level number from 1 to 5 inclusive.
- *n* is a three letter category name and may either be QRY for "query" or SCH for "search".
- *j* is a requirement number (a natural number).

Following the text of a requirement may be a reference to an overarching requirement or use case from which it was derived; these references will be listed in parentheses. Indented text (in a smaller font) represent comments for additional insight or explanation of the requirement.

1.4 Controlling Documents

The following documents exert a controlling influence on this document:

[1] Planetary Data System (PDS) Level 1, 2 and 3 Requirements, March 26, 2010.

[2] Planetary Data System (PDS) 2010 Project Plan, February 2010.

[3] Planetary Data System (PDS) 2010 System Architecture Specification, Version 1.2, May 25, 2011.

[4] Planetary Data System (PDS) 2010 Operations Concept, February 2010.

[5] Planetary Data System (PDS) General System Software Requirements Document (SRD), Version 1.0, June 11, 2011.

[6] Planetary Data System (PDS) Search Scenarios, Version 0.2, June 12, 2011.

1.5 Applicable Documents

The following documents are applicable or associated with this document, but do not exert any controlling influence on this document:

[7] Distributed Infrastructure Design Team. *Search Service Summary*, 2009-05-31.

[8] Distributed Infrastructure Design Team. *PDS Search Services*, 2009-06-03.

1.6 Document Maintenance

The PDS anticipates that the design of the Search service will evolve and mutate over time in order to accommodate ongoing updates to requirements and leverage pioneering technologies, and, similarly, this document should reflect such changes. This document will be limited to design content because the service specification content will be captured with the service documentation (including, but not limited to, an installation guide, an operations manual, etc.). This document will be stored within a configuration control system.

2.0 COMPONENT DESCRIPTION

The Search Service is a deployable server component that accepts queries for data and returns a set of matching results. Informally, queries pose the question of where certain data is, the Search service satisfies that question with the answer of where to find such data. Figure 1 depicts the Search service relative to the other components of PDS 2010.

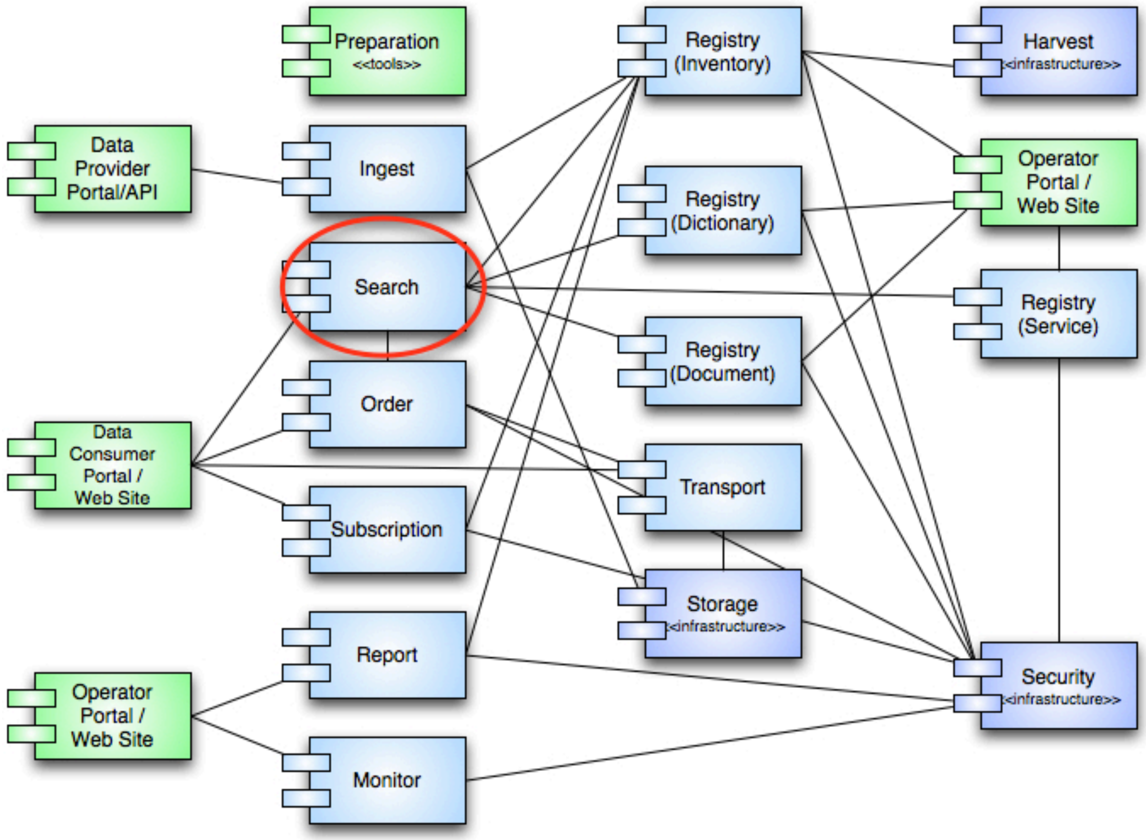


Figure 1. Context of the Search service within the overall PDS 2010 component architecture.

A query in the sense of the Search service means any type of declaration of desiderata, such as "Where are images of Mars taken by an infrared camera between 2009-01-01 and 2009-12-31 UTC?" or "What standards documents mentions the term 'interferometer'?" The desiderata necessarily require foreknowledge of what a particular Search Service curates in order for there to be any hope of an answer to a query.

An installation of the Search Service means a single running instance of the service, or—more specifically—a computing process that answers incoming TCP/IP connections on a set of network interfaces and on a single port. PDS nodes as well as other groups may deploy a Search Service on any compatible platform and use it to catalog particular sets of data and answer queries about that data. The PDS engineering node will run a "main" Search Service that provides answers to queries through the PDS public portal; the University of Maryland will run a backup service to perform the identical role should the engineering node's server fail. PDS nodes and other partners may run their own instances of the Search service for discipline-specific needs.

Data in this sense means anything that the Search Service is capable of cataloging. It includes but is not limited to images, PDS-formatted images, PDS labels, radar altimetry raw data, engineering data records, derived temperature emissions, derived and reduced magnetometer datasets, plain text documents, PDF documents, XML files, HTML documents, and empty files.

Results to a query consist primarily of two things:

- First, a set of *identifiers* that tell where matching data may be found. Identifiers in this sense are Uniform Resource Identifiers (URIs) that, depending on user capability, may be resolved to get actual data.
- Second, a set of *metadata* that annotate each location with descriptors indicating its relevance and context. Such metadata serves to help guide the ultimate end-user in selecting the best result from a potentially large set of matching results.

2.1 Querying

Queries to the Search Service express a user's desiderata PDS data. The Search Service supports several formats of queries in order to better facilitate certain use cases. This section describes, in a high level, those formats.

2.1.1 Open

The open query format is the easiest to use and most familiar to regular users of the Internet as it provides the identical capability that the intimately familiar Google search engine does. The open query format is merely a sequence of textual terms that the service matches against its catalog (subject to certain processing and rules as described in the Implementation section). Matching results depend on the resources containing the terms mentioned.

Figure 2 shows an example of an existing “search engine” that allows the open format of query to be posed.

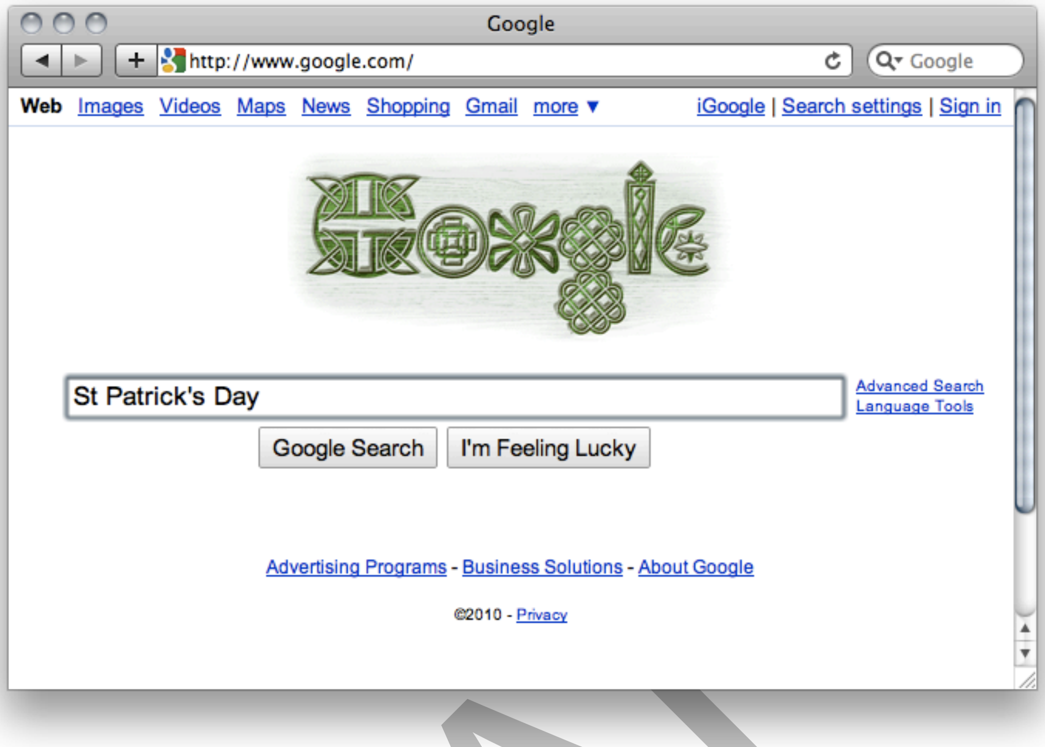
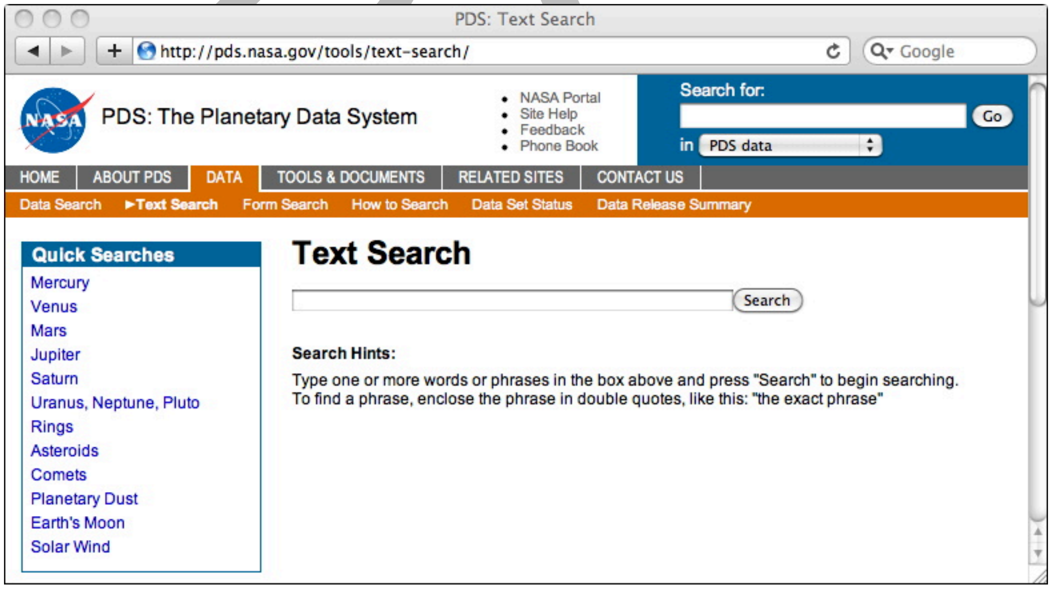

Google
http://www.google.com/
Google
Web Images Videos Maps News Shopping Gmail more ▼
iGoogle | Search settings | Sign in
St Patrick's Day
Advanced Search
Language Tools
Google Search
I'm Feeling Lucky
Advertising Programs - Business Solutions - About Google
©2010 - Privacy

Figure 2. An example of a web-based "search engine" that supports "open" queries.

Figure 3 shows a PDS example of a web-based search interface that supports "open" query requests.



Search Hints:

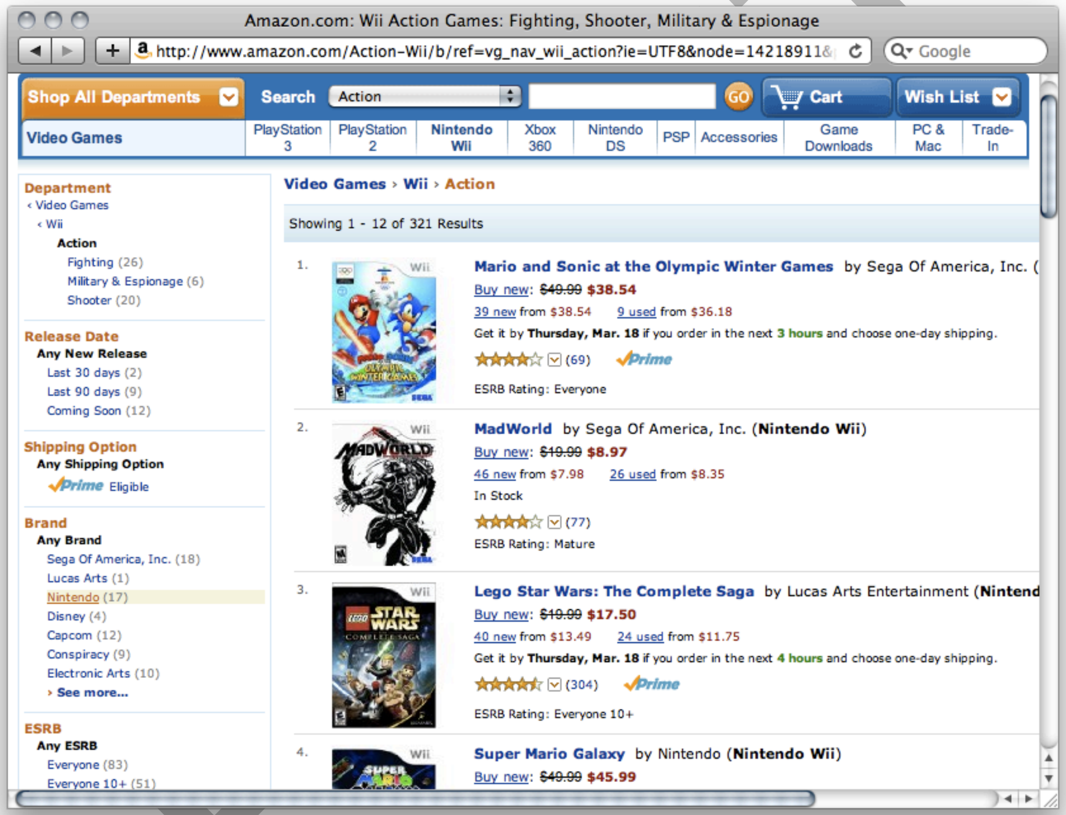
Type one or more words or phrases in the box above and press "Search" to begin searching.
To find a phrase, enclose the phrase in double quotes, like this: "the exact phrase"

Figure 3. A PDS example of a web-based "search engine" that supports "open" queries.

2.1.2 Guided

The guided query format (also known as "faceted query") is highly interactive and is familiar to Internet users who frequent online shopping sites. Guided query presents a series of high-level organizational categories along with a set of terms in each category. Selecting an item constrains what items appear in other categories or may open up entirely new categories. This method of searching allows a broad overview of available data while enabling customized narrowing of matching results through user-selected progressive disclosure.

Figure 4 shows an example of guided search from a web-based "online shopping" site.



Department
‹ Video Games
‹ Wii
Action
Fighting (26)
Military & Espionage (6)
Shooter (20)

Release Date
Any New Release
Last 30 days (2)
Last 90 days (9)
Coming Soon (12)

Shipping Option
Any Shipping Option
Prime Eligible

Brand
Any Brand
Sega Of America, Inc. (18)
Lucas Arts (1)
Nintendo (17)
Disney (4)
Capcom (12)
Conspiracy (9)
Electronic Arts (10)
› See more...

ESRB
Any ESRB
Everyone (83)
Everyone 10+ (51)

Video Games › Wii › Action

Showing 1 - 12 of 321 Results

1. Mario and Sonic at the Olympic Winter Games by Sega Of America, Inc. (
Buy new: ~~$49.99~~ $38.54
39 new from $38.54 9 used from $36.18
Get it by Thursday, Mar. 18 if you order in the next 3 hours and choose one-day shipping.
(69) Prime
ESRB Rating: Everyone

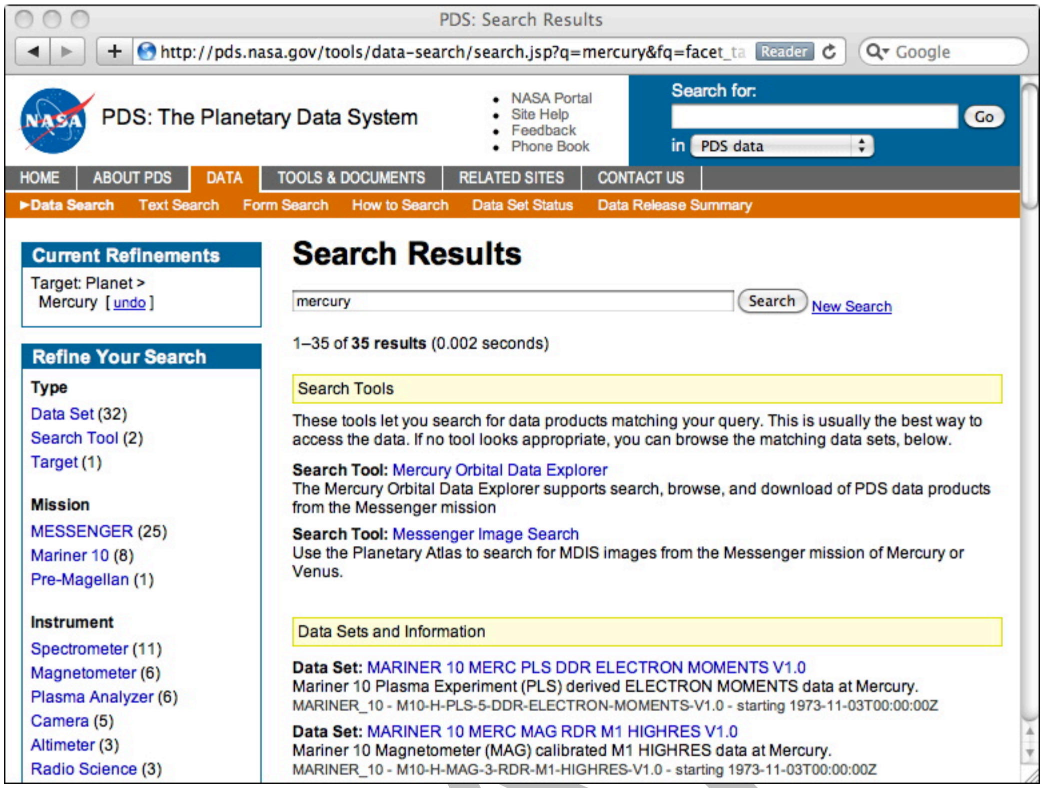
2. MadWorld by Sega Of America, Inc. (Nintendo Wii)
Buy new: ~~$49.99~~ $8.97
46 new from $7.98 26 used from $8.35
In Stock
(77)
ESRB Rating: Mature

3. Lego Star Wars: The Complete Saga by Lucas Arts Entertainment (Nintend
Buy new: ~~$49.99~~ $17.50
40 new from $13.49 24 used from $11.75
Get it by Thursday, Mar. 18 if you order in the next 4 hours and choose one-day shipping.
(304) Prime
ESRB Rating: Everyone 10+

4. Super Mario Galaxy by Nintendo (Nintendo Wii)
Buy new: ~~$49.99~~ $45.99

Figure 4. An example of a guided or "faceted" search in an online shopping context. Facets appear on the left of the window and change as categories are selected.

Figure 5 shows a PDS example of guided search from the Data Search application on the PDS home page.


PDS: Search Results

http://pds.nasa.gov/tools/data-search/search.jsp?q=mercury&fq=facet_ta

PDS: The Planetary Data System

- NASA Portal
- Site Help
- Feedback
- Phone Book

Search for: Go
in PDS data

HOME | ABOUT PDS | DATA | TOOLS & DOCUMENTS | RELATED SITES | CONTACT US

▸Data Search | Text Search | Form Search | How to Search | Data Set Status | Data Release Summary

Current Refinements

Target: Planet > Mercury [undo]

Refine Your Search

Type
Data Set (32)
Search Tool (2)
Target (1)

Mission
MESSENGER (25)
Mariner 10 (8)
Pre-Magellan (1)

Instrument
Spectrometer (11)
Magnetometer (6)
Plasma Analyzer (6)
Camera (5)
Altimeter (3)
Radio Science (3)

Search Results

mercury Search New Search

1–35 of **35 results** (0.002 seconds)

Search Tools

These tools let you search for data products matching your query. This is usually the best way to access the data. If no tool looks appropriate, you can browse the matching data sets, below.

Search Tool: Mercury Orbital Data Explorer
The Mercury Orbital Data Explorer supports search, browse, and download of PDS data products from the Messenger mission

Search Tool: Messenger Image Search
Use the Planetary Atlas to search for MDIS images from the Messenger mission of Mercury or Venus.

Data Sets and Information

Data Set: MARINER 10 MERC PLS DDR ELECTRON MOMENTS V1.0
Mariner 10 Plasma Experiment (PLS) derived ELECTRON MOMENTS data at Mercury.
MARINER_10 - M10-H-PLS-5-DDR-ELECTRON-MOMENTS-V1.0 - starting 1973-11-03T00:00:00Z

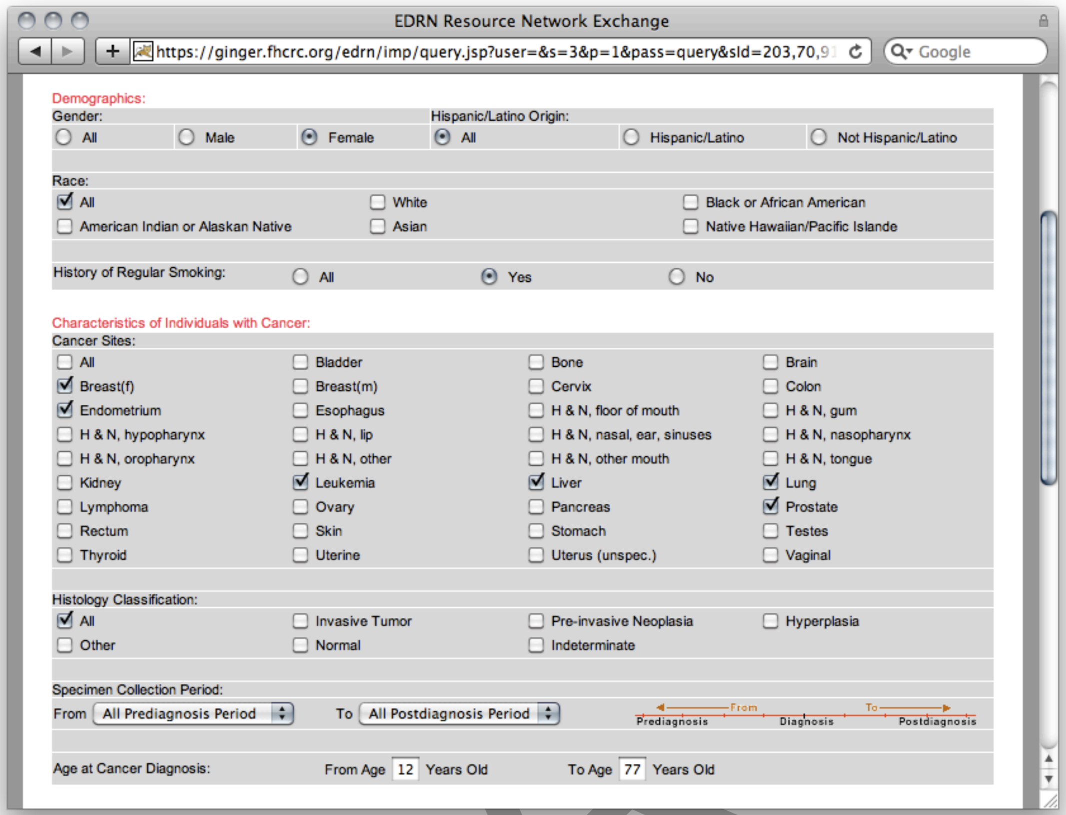
Data Set: MARINER 10 MERC MAG RDR M1 HIGHRES V1.0
Mariner 10 Magnetometer (MAG) calibrated M1 HIGHRES data at Mercury.
MARINER_10 - M10-H-MAG-3-RDR-M1-HIGHRES-V1.0 - starting 1973-11-03T00:00:00Z

Figure 5. A PDS example of guided or "faceted" search.

2.1.3 Constrained

The constrained query format enables the end user or client application to specify constraints on any of the searchable indexes within the Search service. Such constraints may specify exact or relative values, such as latitude equal to 23 degrees or longitude greater than 32 degrees but less than 46 degrees, and so forth. This format requires that the client be intimately familiar with the searchable indexes, their data types, and the kinds of constraints that may be expressed on each. As a result, this query format is targeted towards expert users or client applications that can be programmed with such knowledge.

Figure 6 displays an example of a constrained search form with a (typical) plethora of controls.


EDRN Resource Network Exchange
https://ginger.fhcrc.org/edrn/imp/query.jsp?user=&s=3&p=1&pass=query&sId=203,70,9
Google

Demographics:

Gender: All Male [✓ Female]

Hispanic/Latino Origin: [✓ All] Hispanic/Latino Not Hispanic/Latino

Race:

[✓ All]	White	Black or African American
American Indian or Alaskan Native	Asian	Native Hawaiian/Pacific Islande

History of Regular Smoking: All [✓ Yes] No

Characteristics of Individuals with Cancer:

Cancer Sites:

All	Bladder	Bone	Brain
[✓ Breast(f)]	Breast(m)	Cervix	Colon
[✓ Endometrium]	Esophagus	H & N, floor of mouth	H & N, gum
H & N, hypopharynx	H & N, lip	H & N, nasal, ear, sinuses	H & N, nasopharynx
H & N, oropharynx	H & N, other	H & N, other mouth	H & N, tongue
Kidney	[✓ Leukemia]	[✓ Liver]	[✓ Lung]
Lymphoma	Ovary	Pancreas	[✓ Prostate]
Rectum	Skin	Stomach	Testes
Thyroid	Uterine	Uterus (unspec.)	Vaginal

Histology Classification:

[✓ All]	Invasive Tumor	Pre-invasive Neoplasia	Hyperplasia
Other	Normal	Indeterminate	

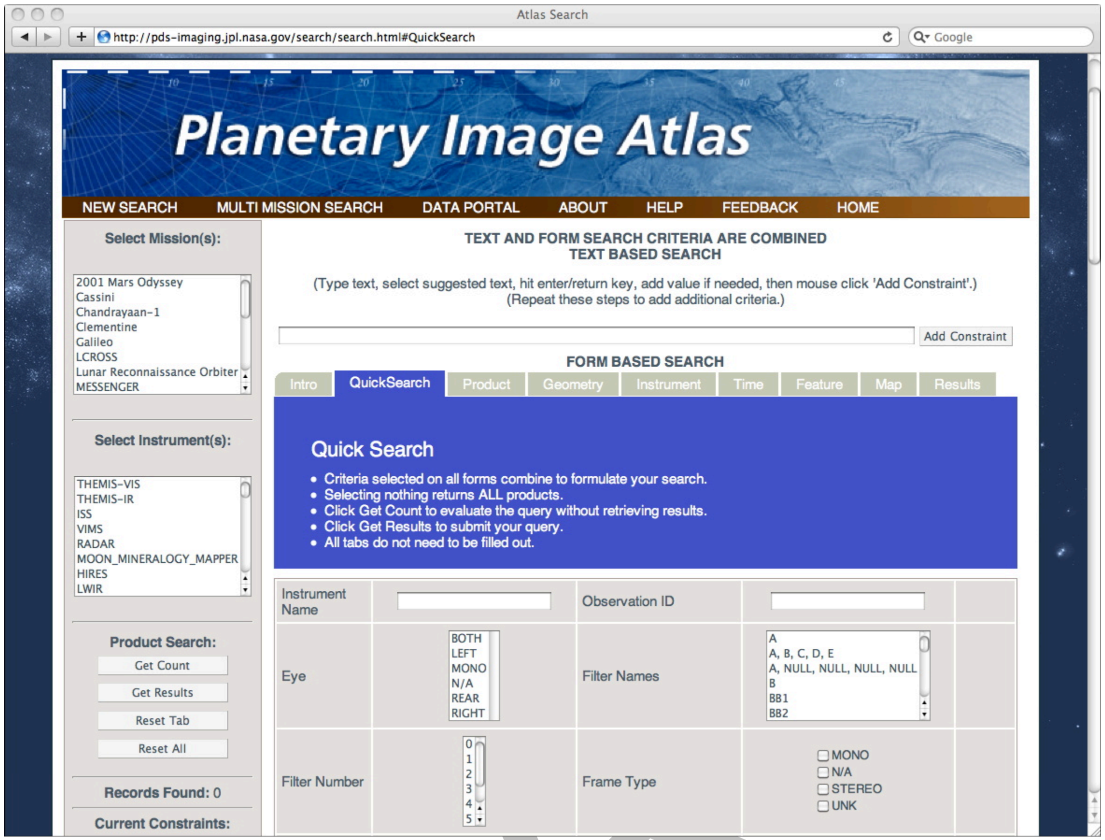
Specimen Collection Period:

From All Prediagnosis Period To All Postdiagnosis Period

Age at Cancer Diagnosis: From Age 12 Years Old To Age 77 Years Old

Figure 6. An example of a constrained search from a web-based medical application. Entry fields, check boxes, radio buttons, and drop-down menus are typical.

Figure 7 shows a PDS example of a constrained search from the Planetary Image Atlas application hosted by the Imaging Node.



TEXT AND FORM SEARCH CRITERIA ARE COMBINED
TEXT BASED SEARCH

(Type text, select suggested text, hit enter/return key, add value if needed, then mouse click 'Add Constraint'.)
(Repeat these steps to add additional criteria.)

- Criteria selected on all forms combine to formulate your search.
- Selecting nothing returns ALL products.
- Click Get Count to evaluate the query without retrieving results.
- Click Get Results to submit your query.
- All tabs do not need to be filled out.

Figure 7. A PDS example of a constrained search.

Although the Planetary Image Atlas application is an example of constrained search, the interface also combines the functionality of guided and open search into a single interface.

2.2 Results

Naturally, a search is nothing without results, which are what the end user is ultimately after. The Search service provides a single format of results regardless of the format of the query. The results are a sequence of Uniform Resource Identifiers (URIs) that name and/or locate data that the Search service has identified as being relevant to the query. The URIs are each reified with metadata that explain each match in order to provide the end user or client application with context that describes the data and why each URI was returned as a match.

The set of metadata is configurable for each installation of the Search service in order to provide necessary context of results for a particular scientific or other application.

2.2.1 Ordering of Results

The Search service offers a feature complementary to the acceptance of queries affecting the presentation of the sequence of matching URIs (and their metadata annotations). This feature enables the end user or client application to specify the sort order of the results.

Unless otherwise specified, results are returned by relevance. The computation of relevance depends on several factors:

- For open searches, keyword frequency ratios to document size provide relative scoring of relevance between hits.
- Guided and constrained searches compute weights assigned to terms and values and leverage multidimensional metrics for relevance scoring.

2.2.2 Segmentation

For certain searches, the number of matching URIs may be enormous and exceed what an end user or client application can comfortably process. As an example, a constrained query for datasets produced after 1900 could well include over a million hits from a Search service primed with every planetary, atmospheric, and geophysical dataset produced worldwide.

In order to overcome this problem the Search service provides the capability for segmentation of hits. Along with the expression of the query's desiderata, the end user or client application may also specify

- A limit on the total number of acceptable hits, such as "1000 or less"
- A cardinal number identifying which of all of the hits to return, such as "the 42nd hit"
- A cardinality on the sequence of hits that include the number, such as "the 10 hits following the 4th"

These features enable segmentation of results for display, for example, over a series of web browser pages (where segmentation takes on the more specialized term "pagination"), or for batch processing of data by a scientific analysis program.

Figure 8 gives an example of web-based pagination controls.

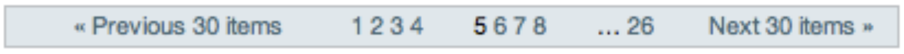


Figure 8. Navigation controls typical of pagination from a web-based content management system.

3.0 USE CASES

This section describes scenarios of the use of the Search service in order to capture the functional requirements. Use cases are a superior tool for capturing and describing requirements by explaining actual, specific scenarios of end user (or client application) goals within the system.

Figure 9 shows a UML diagram of the use cases that depict the requirements of the Search service.

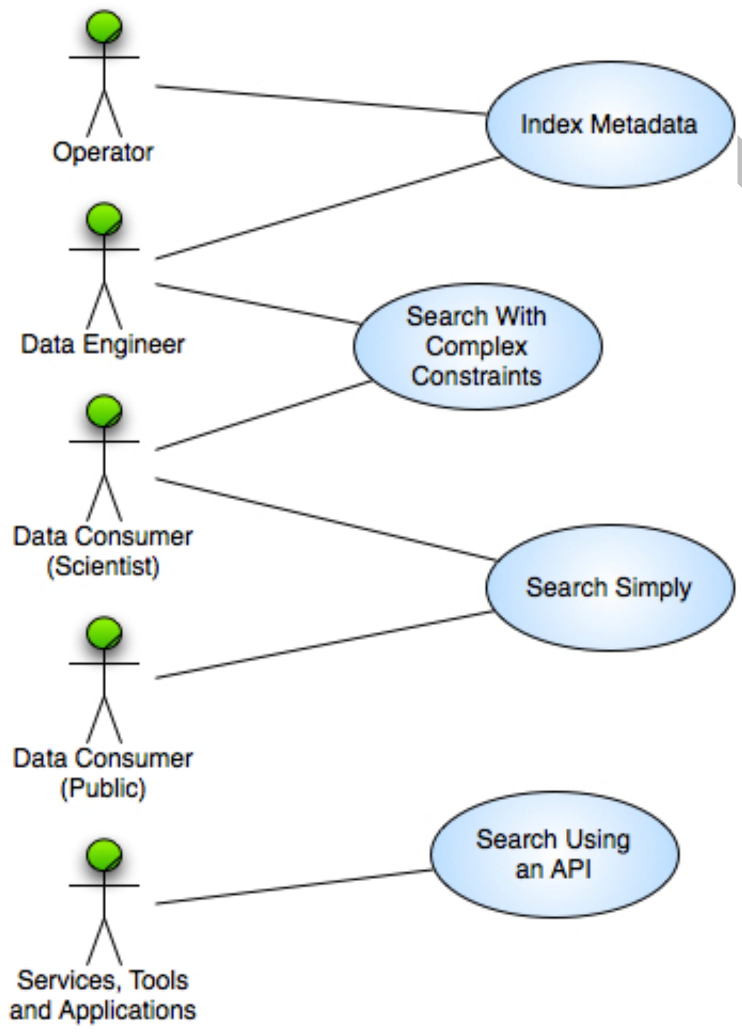


Figure 9. Use case diagram capturing requirements of the Search service.

The remainder of this section gives an overview of each of the actors in the use cases, followed by a description of each use case.

3.1 Actors

This section gives an overview of the actors involved in the use cases. Actors initiate interaction with the Search service in order to achieve a goal.

3.1.1 Data Consumer (Public)

The public's interest in scientific data may be driven by curiosity, a desire to learn, a need for new desktop background, or other reason. Regardless of the motivation, the public as an actor comes to the Search service without a comprehension of indexed data, its organization, or its internal structure.

3.1.2 Data Consumer (Scientist)

Planetary, astrophysical, atmospheric, and other scientists require data to complete their work. We may subdivide scientists by their experience with the kinds of data that may be discovered via the Search service. Novice scientists may not be intimate with the organization or structure of data, while advanced scientists (including those at PDS discipline nodes) could make "power queries"; the Search service accommodates both.

3.1.3 Data Engineer

This actor represents a portion of the PDS Technical group that curates the data before and after it enters the PDS system.

3.1.4 Operator

This actor represents a portion of the PDS Technical group that is responsible for configuring and monitoring the system.

3.1.5 Services, Tools and Applications

This actor represents software components within the PDS system and external to the system that require an Application Programmer’s Interface (API) to access the Search service.

3.2 Cases

The use cases in this section describe an actor's interaction with the Search service from start to finish in order to accomplish a goal.

3.2.1 Index Metadata

In this case, the Search service is configured to and extracts metadata to be indexed from specified sources, enabling it to be later searched for and discovered.

The basic course of events is as follows:

1. The operator or data engineer configures the Search service to generate indexes from specified metadata sources, i.e., Registry service instance(s).
2. The Search service periodically retrieves metadata, including identifiers (URIs) of the data, from the configured sources according to the configured schedule.
3. The Search service analyzes the metadata and updates its indexes based on the analysis.

3.2.1.1 Additional Information

Version	0.
Goal	Update the indexes maintained by the Search service with retrieved metadata.
Summary	The Search service retrieves metadata and indexes it.
Actors	Operator, Data Engineer.
Stakeholders	All.
Notes	This use case must occur before any searches can return results.

3.2.2 Search Simply

In this case, a user (any kind) visits the Search service's human-consumable web interface and enters an open search (as described in 2.1.1). Matching results are displayed 20 at a time in relevance order with result segmentation navigation controls, i.e., "pagination" controls.

The basic course of events is as follows:

1. User visits Search service's human-consumable web page.
2. User types terms into prominent open text search box.
3. User presses Search button.
4. User views results.

Optionally, if presented with more than 20 results, the user may press the "Next Page" link to see the next segment of matching results.

3.2.2.1 Additional Information

Version	0.
Goal	Support a simple search.
Summary	The user executes a search using free text keywords.
Actors	Any.
Stakeholders	Any.
Notes	None.

3.2.3 Search with Complex Constraints

In this case, a Scientist visits the Search service's human-consumable web interface and selects the Advanced Search feature. After s/he fills out the selected values for various constraints and results ordering and segmentation, s/he presses the Search button. Matching results are displayed a page at a time in the selected order with result segmentation navigation controls, i.e., "pagination" controls.

The basic course of events is as follows:

1. Scientist visits Search service's human-consumable web page.
2. Scientist clicks the Advanced Search link.
3. Scientist specifies search criteria by entering constraints on available indexes.
4. Scientist specifies results presentation by selected results sort ordering and number of results per page.
5. User views results.

3.2.3.1 Additional Information

Version	0
Goal	Support a complex constrained search with a potentially complex set of specifications.
Summary	The user executes a search using free text keywords.
Actors	Any.
Stakeholders	Any.

Notes	None.

3.2.4 Search Using an API

In this case, an internal or external application makes use of the web-based programmatic interface of the Search service. At a minimum, the API supports a PDS-specific query language and shall support other query languages such as the Planetary Data Access Protocol (PDAP) as appropriate.

The basic course of events is as follows:

1. The external application formulates a query.
2. The external application sends the query to the Search service.
3. The Search service returns a set of matching results.
4. The external application digests the result set in a way appropriate to it.

3.2.4.1 Additional Information

Version	0
Goal	Support queries via the PDS query language and other query languages as appropriate.
Summary	The Search service accepts these queries (keyword/value pairs) and returns appropriate responses.
Actors	Services, Tools and Applications.
Stakeholders	Any.
Notes	None.

4.0 REQUIREMENTS

During the architecture definition phase of PDS 2010, we decomposed the system into various elements that included the Search service, which was derived from requirement 3.1 of the Level 1, 2, and 3 Requirements document [1]. Although use cases are an excellent tool for capturing requirements, we re-state the requirements of the Search service in an alternative form in this section.

The following system requirements from [1] relate to the Search service and are quoted here for convenience:

3.1.1	PDS will develop and maintain online interfaces allowing users to search the archive.
3.1.2	PDS will develop and maintain online interfaces for discipline-specific searching
3.1.3	PDS will allow products identified within a search to be selected for retrieval.

In addition to the level 4 and 5 requirements specified below, the Search service must also comply with the general service-based requirements found in the General System SRD document [5].

4.1 Level 4 Requirements

The level 4 requirements in PDS represent subsystem, component, or tool requirements, but at a high level. The following list of level 4 requirements are relevant to the Search service:

L4.QRY.1	The system shall provide the capability to search for and identify artifacts registered with the PDS. (3.1.1).
L4.QRY.2	The system shall provide the capability to search for and identify artifacts within a defined scope (i.e., a single discipline). (3.1.2).

4.2 Level 5 Requirements

The level 5 requirements in PDS represent subsystem, component, or tool requirements as the level 4 requirements do, however they do so at a far more detailed level. The following list of level 5 requirements are relevant to the Search service:

L5.SCH.1	The service shall provide a user interface for entering of queries and display of search results accessible from a standards-compliant web browser. (L4.QRY.1, L4.QRY.2, UC 3.2.2, UC 3.2.3).
L5.SCH.2	The service shall degrade gracefully on browsers that lack modern features and not depend on them for operation. (L4.QRY.1, L4.QRY.2, UC 3.2.2, UC 3.2.3).
L5.SCH.3	The service's browser-based user interface shall be Section 508 compliant and adhere to WCAG level A (or better) standards for accessibility. (L4.GEN.5, UC 3.2.2, UC 3.2.3)
L5.SCH.4	The service shall provide a programmatic interface for entering of queries and return of search results that communicates over HTTP for use by client applications developed by PDS, PDS nodes, and others. (L4.QRY.1, L4.QRY.2, UC 3.2.4).
L5.SCH.5	The service shall provide the capability to retrieve metadata associated with registered artifacts for the purpose of generating search indexes. (L4.QRY.1, L4.QRY.2, UC 3.2.1).
L5.SCH.6	The service shall support searching by accepting criteria as a sequence of open text keywords. (L4.QRY.1, L4.QRY.2, UC 3.2.2, UC 3.2.4).
L5.SCH.7	The service shall support searching by accepting criteria as a series of values for constraints on specified indexes. (L4.QRY.1, L4.QRY.2, UC 3.2.3, UC 3.2.4).
L5.SCH.8	The service shall support narrowing of additional index results based on specifications of terms and/or values on indexes. (L4.QRY.1, L4.QRY.2, UC 3.2.2, UC 3.2.3, UC 3.2.4).
L5.SCH.9	The service shall support the ordering of results based on specified criteria including relevance and specified indexes. (L4.QRY.1, L4.QRY.2, UC 3.2.2, UC 3.2.3, UC 3.2.4).
L5.SCH.10	The service shall provide results to a search as a sequence of matching URIs to resources that contain search desiderata. (L4.QRY.1, L4.QRY.2, UC 3.2.2, UC 3.2.3, UC 3.2.4).
L5.SCH.11	The service shall annotate each URI of a result with metadata describing the URI. (L4.QRY.1, L4.QRY.2, UC 3.2.2, UC 3.2.3, UC 3.2.4).

L5.SCH.12 The service shall support configuration on the kinds of indexes maintained on indexed data, including indexes that differ by data type, by data conversion, by index generation methodology, and by metadata maintenance for result annotation. (L4.QRY.1, L4.QRY.2, UC 3.2.1).

L5.SCH.13 The service shall capture metrics pertaining to its search indexes usage and contents. (L4.QRY.1, L4.QRY.2, UC 3.2.1).

5.0 DESIGN PHILOSOPHY, ASSUMPTIONS, AND CONSTRAINTS

The intent of the Search service is to make it easy to find PDS data. The Search service itself does not contain PDS data, but instead *indexes* them by being presented with a set of PDS data to process. Processing examines the data and adds to the Search service's indexes. Queries scan the Search service's indexes of previously indexed data to find potential matches. The service returns a sequence of URIs to the originally processed data that match, as well as any additional metadata kept in the indexes.

Due to the variability of discipline-specific data, the Search service does **not** enforce a specific set of catalog indexes. Rather, it enables each installation to configure the kinds of indexes needed to support searches. Indexes may:

- Be of various data types (such as date & time or floating point or HTML text).
- Require conversion or processing (such as removal of HTML tags, or ignoring of "stop words" like definite articles, prepositions, and the like).
- Display in search results as metadata annotations.

The Search service does not *ingest* or *store* data, but instead maintains indexes *about* the data and *identifiers* (in the form of URIs) to them. Results of a search are the identifiers, plus metadata annotations about the identifiers. It is then up to the client program or end user to examine the metadata annotations and choose whether to retrieve the identified data. We expect that in most if not all cases the URIs will take the form of URLs that enable retrieval of data directly by virtually any browser or with some rudimentary data movement software.

6.0 ARCHITECTURAL DESIGN

The architectural design includes the components that comprise the Search service as well as both internal and external interfaces with the service and the data model of which it takes advantage.

6.1 Component Architecture

Figure 10 details the high-level architecture and interactions of the Search service.

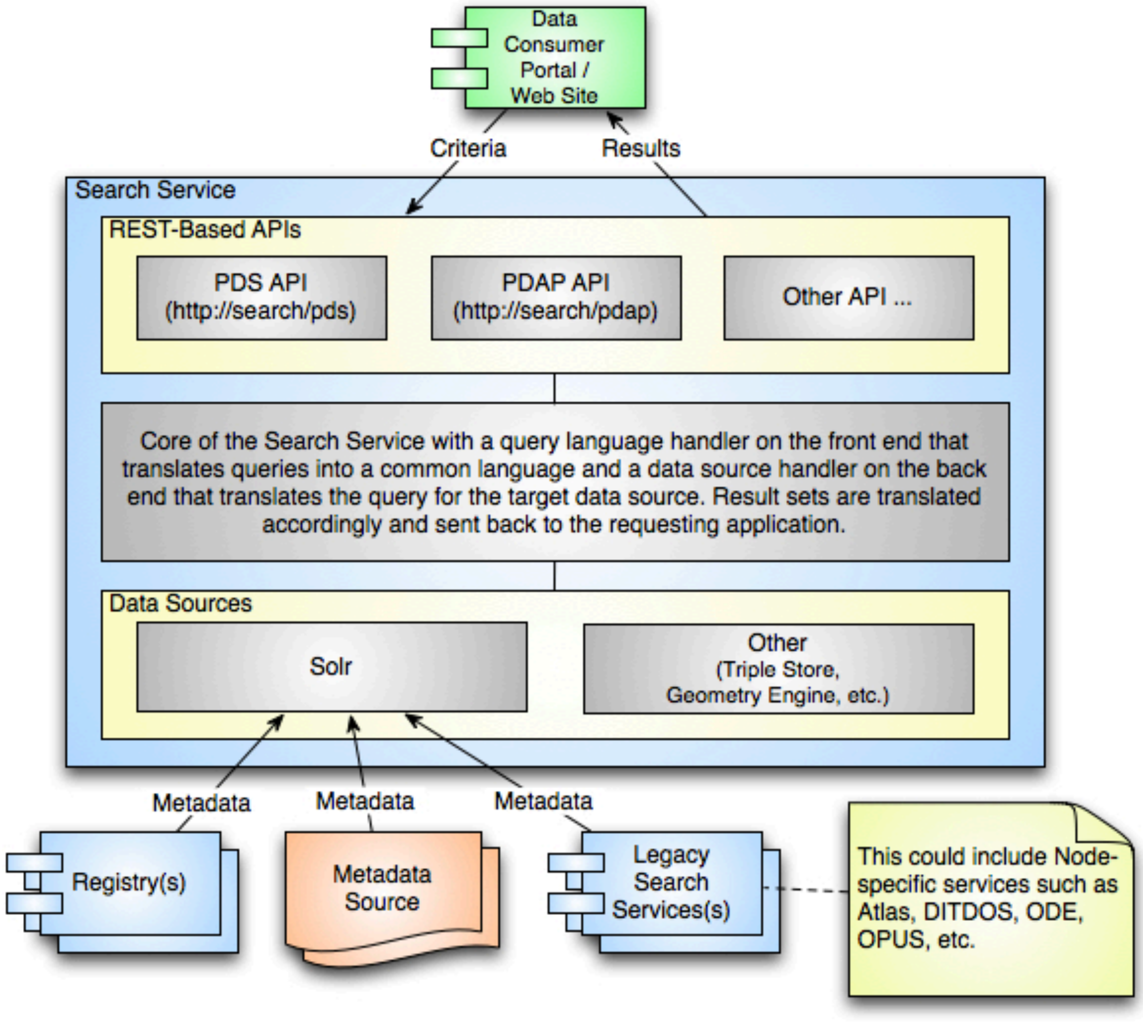

Data Consumer Portal / Web Site
Criteria
Results
Search Service
REST-Based APIs
PDS API (http://search/pds)
PDAP API (http://search/pdap)
Other API ...
Core of the Search Service with a query language handler on the front end that translates queries into a common language and a data source handler on the back end that translates the query for the target data source. Result sets are translated accordingly and sent back to the requesting application.
Data Sources
Solr
Other (Triple Store, Geometry Engine, etc.)
Metadata
Metadata
Metadata
Registry(s)
Metadata Source
Legacy Search Services(s)
This could include Node-specific services such as Atlas, DITDOS, ODE, OPUS, etc.

Figure 10. High-level architecture and interactions of the Search service.

The Data Consumer Portal component in the above diagram represents the user interface to the Search service including Node-developed browser-based implementations and other applications that may access the service directly via the REST-based HTTP interface. The REST-based interface allows for more than one query language to be utilized. A PDS-specific query language will be implemented first. Other possible query languages include the Planetary Data Access Protocol (PDAP).

The architecture also provides for multiple data sources. The initial and main data source is an indexed metadata search engine. Due to its prevalent use with PDS already, Apache's Solr will be utilized. Although Solr will satisfy the great majority of PDS search requirements, the architecture will allow for other data sources such as geometry engines and triple store solutions. The search engine will pull from three types of metadata sources for index generation:

Registry(s)
This component represents the PDS Registry service and will be the main source of metadata for each of the Search service deployments. Depending on the scope of a given Search service instance, the service may need to retrieve metadata from more than one Registry service.

Metadata Source
This source of metadata is somewhat undefined. One example of a metadata source is the PDS3 index table that accompanies a data set. This table contains additional or updated information for each product in the data set that cannot be found in the product labels. Other possible sources would serve the similar purpose of augmenting the metadata for a product registered in a Registry service instance.

Legacy Search Services(s)
In order to accommodate existing search services available from the Nodes, the metadata contained within these services will be extracted and indexed in the same manner as the metadata contained within the Registry service instances.

Figure 11 details the two main deployment scenarios for the Search service along with depicting its place within the bigger picture of the PDS 2010 system.

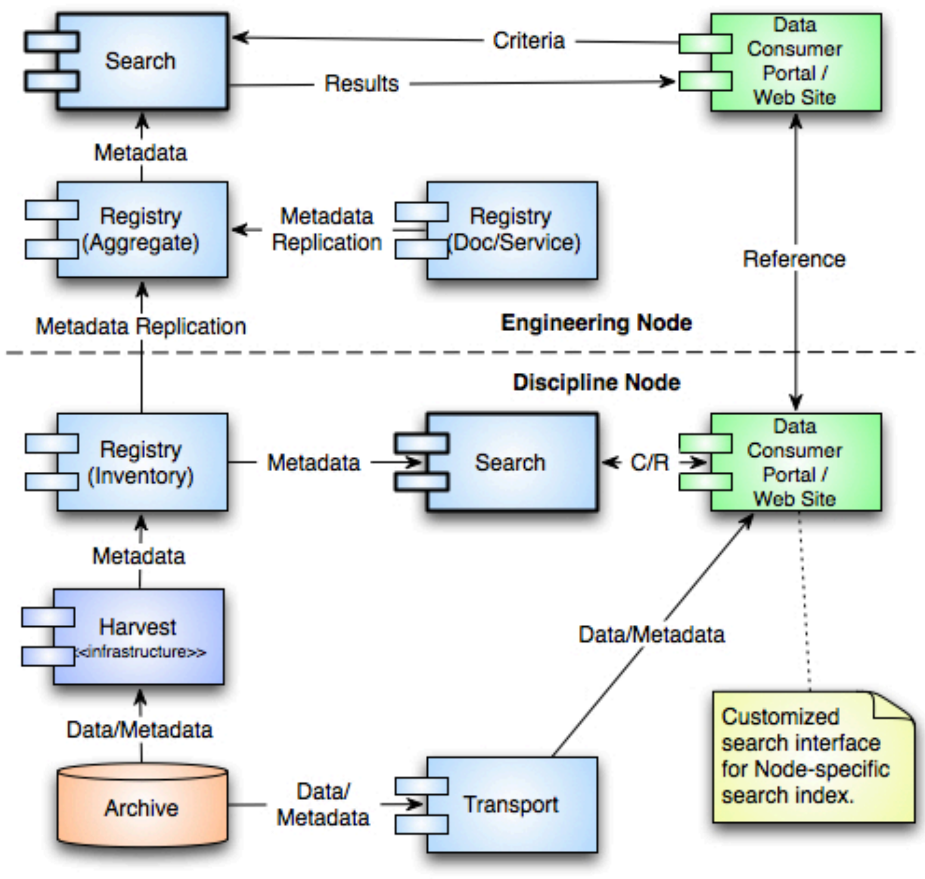


Figure 11. Deployment scenarios for the Search service.

The diagram above depicts two instances of the Search service:

Hosted at the Engineering Node
As depicted in the diagram above, metadata from the various Registry service instances throughout PDS are replicated and stored in an aggregate instance of the Registry service hosted at the Engineering Node (EN). This Registry service instance is the source of catalog metadata for the Search service instance also hosted at the EN. This instance of the Search service will support the high-level search interfaces found on the PDS home page.

Hosted at the Discipline Node(s)
The plan is to have a local instance of the Search service installed at each Node to support discipline-specific search needs. This instance would only be necessary if the Node chose to generate and host a discipline-specific search index to support a custom-built search interface for their users.

6.2 External Interface Design

The Search service offers a REST-based or REST-like external interface that is accessible via the Hypertext Transfer Protocol (HTTP). A REST-based interface exhibits the following characteristics:

- A URL assigned to every resource
- Formulate URLs in a predictable manner
- Use HTTP methods for actions on a resource (GET, POST, PUT and DELETE)
- Leverage HTTP protocol headers and response codes where applicable

The goals for the interface are as follows:

- Keep the service simple and refrain from adding too much functionality
- Allow messaging in the form of XML or JavaScript Object Notation (JSON)
- Allow for extensibility as new artifact types are defined

In addition, each interface should adhere to the following:

- Be self documenting
- Have a defined standard response including passed parameters
- Provide a schema for the defined response
- Provide a command-line method of execution

More information on the query language is forthcoming.

6.3 Internal Interface Design

The internal interfaces of the Search service comprise the objects that are the major architectural “players” in its design. Figure 12**Error! Reference source not found.** shows a UML class diagram that depicts the architecture of the Search service.

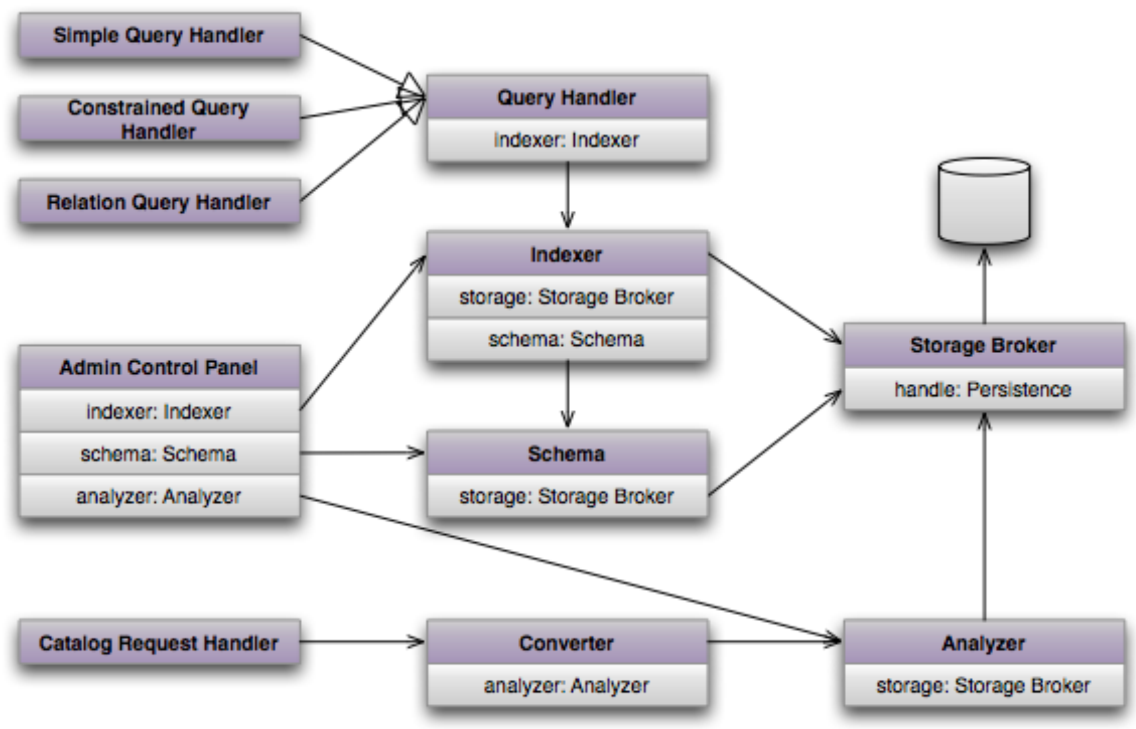


Figure 12. Overall architecture of the Search service.

This section describes each of these parts.

- **Query Handler**. The query handler is the bridge that connects external queries to the internal indexer for resolution. It accepts queries from external applications in REST-esque formats, parses them for correctness and consistency, hands them off to the indexer, gathers results, and returns them. Concrete query handlers exist to handle simple queries (free text), constrained queries (single, sequenced, or ranges of values of metadata), and relational queries.
- **Indexer**. The indexer is the main workhorse for query resolution and the interface to the persistence backends. It uses queries to scan for matches in the indexed content including both the inverted index and numeric trie. It annotates results from the schema component and returns them to the query handler.
- **Schema**. The schema's job is to maintain persistent metadata about indexed items. It has an interface to the backend persistence and is responsible for annotations on results, as well as accepting administrative commands about schema updates, i.e., what metadata to track, in what formats and datatypes, and so forth.
- **Catalog Request Handler**. The catalog request handler takes content from external sources (such as instances of the registry service), performs first-level error and consistency checks, and—if passing—passes the content to be cataloged by the converter.

- **Converter**. The converter's responsibility is to take new or updated content that users wish searchable and prepare it for cataloging. For tabular data, this may be locating specific columns for indexing and other columns for both indexing and schema storage. For textual data, this may include stripping of markup such as rich text or HTML indicators, as well as stripping of stop words and the like.
- **Analyzer**. The analyzer takes prepared content from the converter and does the actual heavy lifting of cataloging it. This updates the inverted index and numeric trie and places the updated catalogs into the storage broker's persistence.
- **Storage Broker**. The storage broker is the interface to the long-term (hard disk) persistence for all cataloged data and their schema annotations. Its duties entail handling efficient disk file representations for the inverted index, numeric trie, and schema stores.
- **Administrative Control Panel**. The administrative control panel provides a browser-based facility for maintaining the Search service, enabling such functions as schema/metadata updates, index pruning, re-indexing, compression, statistics logging, and so forth.

6.4 Data Model

Figure 13 details the data model of the Search service.

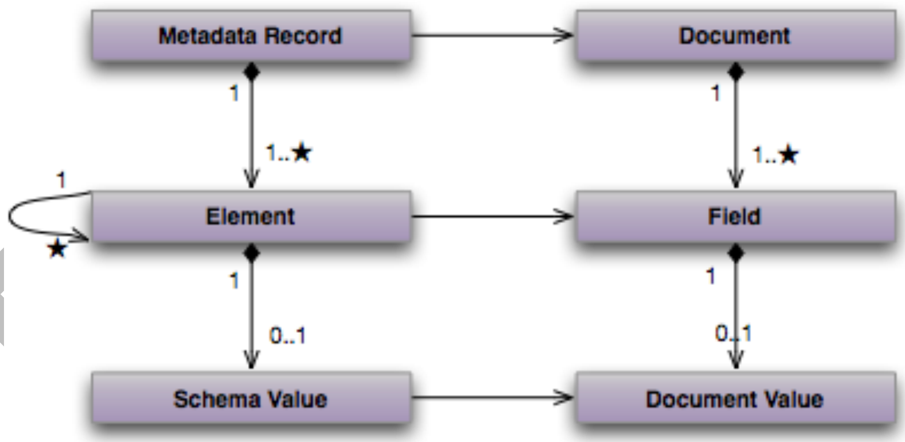


Figure 13. UML diagram depicting the internal data model of the Search service.

The entities depicted are described as follows:

- A **metadata record** represents a single indexed item in the catalog (termed a document). Searching the catalog involves searching the metadata records. It refers to a single document (which can be a document in the traditional sense, a collection of data sets, a data set, a product, a granule, and so forth). Metadata records contain one or more elements.

- A **document** is indexed into the catalog. Documents take the form of text files, structured text, imagery, data sets, archives, and so forth. Documents contain one or more fields, which contain valuable data.
- An **element** is an entity within a catalog that captures the state of a document's field. As the search service searches through metadata records, it uses the information captured in the elements to determine a match.
- A **field** defines the structure of the valuable data of a document. An unstructured document, like a plain text file, may consist of a single field whose content is the text. Structured documents, such as tabular data files, may have multiple fields (one for each column).
- A **document value** is the actual data in a field of a document. Document values represent the concrete information, data, and knowledge PDS seeks to preserve and make available for research. In a plain text file, a single document value represents the text in the file. An image file may have document values for three fields, each representing the color contributions to each pixel.
- A **schema value** is a duplication of a document value within the catalog. Recall that a successful match of a search does not return the matched entity but rather the URI for the document. Users must then resolve and retrieve the identified document to truly determine if it matched their desiderata. By duplicating some of the document values as schema values, a subset of the actual matched entity may be presented to users. This enables software (and humans) to decide whether to actually retrieve the identified matching documents.

7.0 ANALYSIS

Search systems have a history as long as computer science itself. While there may be important discoveries to be made in the field of search, PDS must take a pragmatic approach in analyzing existing algorithms and implementations and adopting such for the Search service.

Searching a corpus like the PDS is different from "search engine" scenarios in that the data comprises a disparate mix of highly regular scientific observations (instrument readings, engineering data records, and the like), imagery (derived from more plain instrument readings or directly imaged from various wavelengths), as well as unstructured information (mission documentation, data set calibration notes and the like). The Search service must be capable of transparently searching over all these entities.

Search systems can be characterized by the approach they take to performing a search. Direct searches are horribly inefficient and merely map desiderata over content corpus in the hopes of a match. Larger desiderata and larger content result in longer search times. Therefore, pre-processed searches are definitely the way to go.

Pre-processing can amortize the time required to perform a search over either the desiderata or the content. Pre-processed search terms (such as in the Boyer-Moore algorithm) are helpful with smaller content corpora. However, the PDS has anything but small corpora. Therefore, pre-processing the data to search is appropriate. In this form, pre-processing is termed *indexing*.

Indexing strategies themselves bring to bear several considerations: how indexes are made persistent, how they're updated, how they utilized in queries, what kinds of information may be returned, what metadata is yielded, whether results can be ranked, whether results include the originally indexed resources, whether there is concurrency in updates and queries, what policies for fault tolerance they support, and what types of data may be saved.

Rather than reproduce a full analysis of varying index systems, PDS will select a *numeric trie* combined with an *inverted index* as the appropriate technologies that meets the majority of the points outlined above.

Range-constrained and range-specific queries for numeric values will generally access the trie for extremely fast searches, with minimal pre-processing for certain queries (such as geographic longitudinal searches that cross a planetary prime meridian). Text queries utilize the inverted index. Both support relevance ranking as well as configurable metadata/schema retrieval.

In addition to standard index generation techniques described above, the Search service will also take advantage of structures within the PDS product label, or more specifically their representation within the Registry service, to determine relevance ranking and relationships to other products. The PDS product label design includes a cross-reference section where data providers can specify references to other products within PDS including the reference type. An example of a cross-reference entry found in a product label is as follows:

```
…
<Cross_Reference_Area_Product>
  <Product_Reference_Entry>
    <lidvid_reference>
      URN:NASA:PDS:instrument_host.MRO::1.0
    </lidvid_reference>
    <reference_association_type>
      has_instrument_host
    </reference_association_type>
  </Product_Reference_Entry>
</Cross_Reference_Area_Product>
…
```

The above reference specifies that an instrument on the MRO spacecraft produced the current PDS product. MRO is classified as an instrument host in PDS and is also represented by a product that is registered with an instance of the Registry service.

The Harvest tool recognizes references, like the one above, and registers a corresponding association with the Registry service. The association simply captures the source, target and type of the reference. The reference above is considered a primary reference. A referenced product may in turn specify references to additional products. These references are considered secondary references and so on. Although prioritization of these references has yet to be determined, it is assumed that index generation will take advantage of these references and their priorities in order to enhance the result set for a search request.

8.0 IMPLEMENTATION

The PDS engineering node will develop and deploy the components of PDS 2010 in a series of phases, with additional capabilities appearing over time. The product of each phase is termed a build. The builds are as follows:

- **Build 1** — This build comprises the ingestion subsystem, including the security, harvest, registry (inventory, dictionary, document, service), and report components. It also delivers the data provider tool suite.
- **Build 2** — This build completes the distribution subsystem, including the search component (the topic of this document) and the monitor component. It also deploys a revised web site and general portal applications.
- **Build 3** — This build consists of enhanced user capabilities, including the order and subscription components. It also integrates discipline node applications and science services.

The implementation platform for the search component will leverage a portable software-based "virtual machine" that will enable the Search service to be deployed to a wide variety of popular operating system platforms. At a minimum, the service will be deployable to Mac OS X, Linux, and certain other Unix-based systems, as well as Windows systems. Other systems may or may not be supported depending on availability of an underlying virtual machine for each platform.

In order to answer REST-based enterprise-level service inquiries and to facilitate implementation, the Search service shall be implemented within the context of an application server container. Such a container abstracts lower level networking and operating system services into a common API and manages routine functional "housekeeping". However, rather than require sites that adopt the Search service the overhead of installing a compatible application server, the Search service installer will include deploying its own application server. The following diagram depicts the deployment scenario:

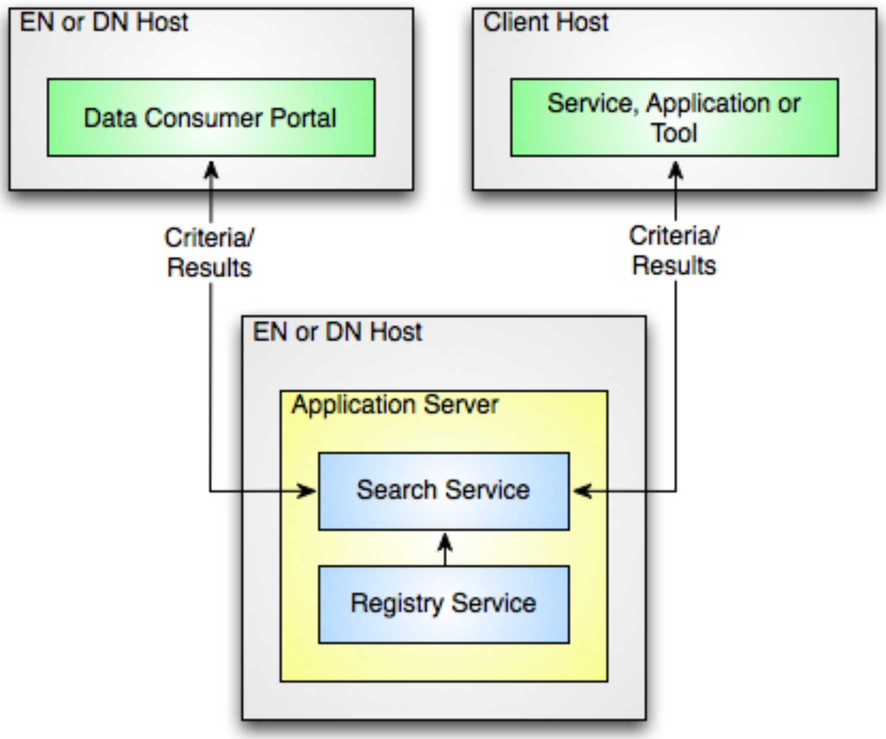


Figure 14. Diagram depicting Search service deployment.

The Search service accesses the Registry service via its REST-based interface over HTTP while the search client applications access the Search service via its REST-based interface over HTTP.

APPENDIX A ACRONYMS & ABBREVIATIONS

The following acronyms made an appearance in this document:

API	Application Programmer's Interface
ESA	European Space Agency
HTML	Hypertext Markup Language
IP	Internet Protocol
ISRO	Indian Space Research Organization
JAXA	Japan Aerospace Exploration Agency
JPL	Jet Propulsion Laboratory
MRO	Mars Reconnaissance Orbiter
NASA	National Aeronautics and Space Administration
OODT	Object Oriented Data Technology
PDAP	Planetary Data Access Protocol
PDF	Portable Document Format
PDS	Planetary Data System
QRY	Query component
REST	Representational State Transfer
SCH	Search component
SDD	Software Design Document
SRD	Software Requirements Document
TBD	To Be Determined
TCP	Transport Control Protocol
UML	Unified Modeling Language
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
URN	Uniform Resource Name
UTC	Coordinated Universal Time
WCAG	Web Content Accessibility Guidelines
XML	Extensible Markup Language