Planetary Data System

Search Protocol

Sean Hardman

October 17, 2011
Version 0.2

Jet Propulsion Laboratory
Pasadena, California

CHANGE LOG

Revision	Date	Description	Author
0.1	2011-08-01	Initial draft.	S. Hardman
0.2	2011-10-17	Focused on the document on a PDS-specific search protocol instead of adapting PDAP for PDS.	S. Hardman

TABLE OF CONTENTS

1.0 INTRODUCTION

The PDS 2010 effort will overhaul the PDS data architecture (e.g., data model, data structures, data dictionary, etc.) and deploy a software system (online data services, distributed data catalog, etc.) that fully embraces the PDS federation as an integrated system while leveraging modern information technology.

1.1 Document Scope and Purpose

This document captures the search protocol for the PDS 2010 system. This protocol is targeted for implementation as the REST-based interface for the Search Service and is proposed as the protocol for other search services across the PDS. This document is intended for the reviewer of the protocol as well as the developer of the protocol (for implementation) and tester of the protocol (for quality assurance).

1.2 Method

This document captures the input and output structures specified by the protocol with no adherence to a particular method.

1.3 Notation

This document does not utilize special notations.

1.4 Controlling Documents

[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 Service Software Requirements and Design Document (SRD/SDD), Version 0.4, June 12, 2011.

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

1.5 Applicable Documents

[8] Planetary Data Access Protocol (PDAP), Version 1.1, September 9, 2011.

[9] VOTable Format Definition, Version 1.2, November 30, 2009.

[10] Key words for use in RFCs to Indicate Requirement Levels, March 1997.

1.6 Document Maintenance

The system design will evolve over time and this document should reflect that evolution. This document is under configuration control.

2.0 DESCRIPTION

The search protocol for the PDS 2010 system provides a single interface for discovery and access to data across the PDS. The protocol heavily leverages the query parser syntax from Apache Lucene as well as certain characteristics from the Planetary Data Access Protocol (PDAP) [8] developed by the International Planetary Data Alliance (IPDA).

As described in the Search Service SRD/SDD [6], this protocol will be implemented as a REST-based interface for the Search service. Although the Search Service is the main implementation of this protocol, it is anticipated that this protocol will find its way into other PDS services. For example, a discipline-specific search service developed at one of the PDS Nodes may want to implement this protocol directly into their service.

The protocol serves three different purposes within the system:

Data Discovery
The search protocol supports the discovery of content. In the case of PDS, content may be at the catalog or product level. The protocol supports this purpose by accepting search parameters as input and returning a structured result set containing metadata.

Data Access
The search protocol supports the retrieval of product data in the form of a file. Impacts on the search protocol include additional parameters for specifying desired transformations and packaging.

Service Linking
In order to support PDS's integrated search paradigm, passing of search parameters from one service to another must be supported. Integrated search consists of a catalog-level search interface facilitating discovery of missions, instruments, data sets, etc. This interface will also direct users to product-level search interfaces offered by the Nodes. Integration/linking is achieved by forwarding the user to the desired product-level search interface while passing their current set of search criteria, in the form of search parameters, to that interface. This alleviates the need for the user to enter that information again and enables the product-level search interface to display an appropriate result set based on that criteria. Although this can be achieved with any interface that supports parameter passing, the interface is streamlined when the target service supports the search protocol described in this document. In this case, the query request can simply be forwarded to the target service.

3.0 COMPLIANCE

This search protocol is being captured in a separate document from the Search Service design specification [6] in order to facilitate its implementation in other services deployed by the PDS Nodes as well as services deployed by the international community. This section details the compliance levels that an implementation may achieve.

The keywords “MUST” and “SHOULD” as used in this document are to be interpreted as described in RFC 2119 [10]. An implementation that satisfies all of the MUST level requirements is said to be “conditionally compliant”. An implementation that satisfies all of the MUST level requirements and all of the SHOULD level requirements is said to be “unconditionally compliant”. The following are key compliance items:

1. The Common Parameters MUST be supported as defined in section 4.1.
2. The Instance-Specific Parameters SHOULD be supported as defined in section 4.2.
3. The Control Parameters MUST be supported as defined in section 4.3.
4. The Simple Syntax MUST be supported as defined in section 5.1.
5. The Advanced Syntax SHOULD be supported as defined in section 5.2.

4.0 PARAMETERS

This section details the parameters supported by the search protocol.

4.1 Common Parameters

The parameters defined in this section are considered common for all PDS products and must be supported by the underlying implementation. Unless otherwise stated, each of these parameters may be specified multiple times in a single request. An example of specifying a parameter multiple times can be found in the Syntax section.

4.1.1 identifier

In PDS4, every artifact is considered a product, and every product is assigned an identifier and a version. This parameter represents that identifier and is used to uniquely identify a product or multiple versions of a product. This parameter maps to the *logical_indentifier* attribute specified in a product label or the combination of the *logical_indentifier* and *version_id* attributes. The former case may return multiple versions of a product where the latter case should return one product. Both of these attributes are found in the *Identification_Area* class in PDS4 product labels.

This parameter also maps to the *alternate_id* attribute specified in a product label. That is a user-specified attribute that will likely be the home for the *DATA_SET_ID* and *PRODUCT_ID* keywords for products migrated from PDS3 to PDS4.

4.1.2 instrument

This parameter is used to identify products associated with or captured from a specific instrument. This parameter maps to the *instrument_name* attribute that is found in the *Subject_Area* class in PDS4 product labels. In the case where *product_class* is equal to “Product_Instrument” or “Product_Instrument_PDS3”, this parameter also maps to the *title* and *alternate_title* attributes.

In PDS3, instruments are identified by both the *INSTRUMENT_ID* and *INSTRUMENT_NAME* keywords. This parameter is intended to encompass the values for both of these keywords. It is not clear at this time whether the *INSTRUMENT_ID* value, for instrument products migrated from PDS3 to PDS4, will be captured in the *alternate_id* or the *alternate_title* attribute in the PDS4 label. The *INSTRUMENT_NAME* keyword is captured in the *title* attribute.

4.1.3 instrument-host

This parameter is used to identify products associated with or captured from an instrument on a specific instrument host (e.g., spacecraft). This parameter maps

to the *instrument_host_name* attribute that is found in the *Subject_Area* class in PDS4 product labels. In the case where *product_class* is equal to "Product_Instrument_Host" or "Product_Instrument_Host_PDS3", this parameter also maps to the *title* and *alternate_title* attributes.

In PDS3, instrument hosts are identified by both the *INSTRUMENT_HOST_ID* and *INSTRUMENT_HOST_NAME* keywords. This parameter is intended to encompass the values for both of these keywords. It is not clear at this time whether the *INSTRUMENT_HOST_ID* value, for instrument host products migrated from PDS3 to PDS4, will be captured in the *alternate_id* or the *alternate_title* attribute in the PDS4 label. The *INSTRUMENT_HOST_NAME* keyword is captured in the *title* attribute.

4.1.4 instrument-type

This parameter is used to identify products associated with or captured by a specific type of instrument. This parameter maps to the *instrument_type* attribute that is found in the *Instrument_PDS3* class in PDS4 product labels where *product_class* is equal to "Product_Instrument_PDS3".

Although not prevalent in PDS4 product labels, this parameter is common in PDS search scenarios and is facilitated with a taxonomy that relates instrument types with instruments.

4.1.5 investigation

This parameter is used to identify products associated with a specific investigation. This parameter maps to the *investigation_name* attribute that is found in the *Subject_Area* class in PDS4 product labels. In the case where *product_class* is equal to "Product_Investigation" or "Product_Mission_PDS3", this parameter also maps to the *title* and *alternate_title* attributes.

In PDS3, this parameter is equivalent to the concept of a mission and the *MISSION_NAME* keyword.

4.1.6 observing-system

This parameter is used to identify products associated with a specific observing system. This parameter maps to the *observing_system_name* attribute that is found in the *Subject_Area* class in PDS4 product labels.

4.1.7 person

This parameter is used to identify products associated with a specific person, usually the principal investigator. This parameter maps to the *full_name* attribute that is found in the *Subject_Area* class in PDS4 product labels.

4.1.8 product-class

This parameter is used to constrain the types of products returned in the result set. This parameter maps to the *product_class* attribute that is found in the *Identification_Area* class in PDS4 product labels. This parameter specifies the class of products to include in the result set subject to the other search criteria.

This is prevalent in PDS4 product labels, but the value set follows the class hierarchy of the data model. This parameter is facilitated by a taxonomy that represents that hierarchy.

4.1.9 start-time

This parameter is used to identify products within a specific data/time range. This parameter maps to the *start_date_time* attribute that is found in the *Observation_Area* class in PDS4 product labels. In the case where *product_class* is equal to “Product_Investigation” or “Product_Mission”, this parameter also maps to the *start_date* attribute found in the *Investigation* or *Mission* class, respectively. In the case where *product_class* is equal to “Product_Mission_PDS3”, this parameter also maps to the *mission_start_date* attribute found in the *Mission_PDS3* class.

4.1.10 stop-time

This parameter is used to identify products within a specific data/time range. This parameter maps to the *stop_date_time* attribute that is found in the *Observation_Area* class in PDS4 product labels. In the case where *product_class* is equal to “Product_Investigation” or “Product_Mission”, this parameter also maps to the *stop_date* attribute found in the *Investigation* or *Mission* class, respectively. In the case where *product_class* is equal to “Product_Mission_PDS3”, this parameter also maps to the *mission_stop_date* attribute found in the *Mission_PDS3* class.

4.1.11 target

This parameter is used to identify products associated with or containing data for a specific target. This parameter maps to the *target_name* attribute that is found in the *Subject_Area* class in PDS4 product labels. In the case where *product_class* is equal to “Product_Target” or “Product_Target_PDS3”, this parameter also maps to the *title* and *alternate_title* attributes.

4.1.12 target-type

This parameter is used to identify products associated with or containing data for a specific type of target. This parameter maps to the *target_type* attribute that is

found in the *Target_PDS3* class in PDS4 product labels where *product_class* is equal to “Product_Target_PDS3”.

Although not prevalent in PDS4 product labels, this parameter is common in PDS search scenarios and is facilitated with a taxonomy that relates target types with targets.

4.1.13 term

This parameter does not directly map to a specific attribute found in PDS4 product labels but can be used to query against several attributes based on the configuration of the underlying implementation. Candidate attributes include any description-related attributes and specifically the *keywords* attribute found in the *Subject_Area* class.

This parameter may not be specified multiple times in a single request.

4.1.14 title

This parameter is used to identify products by their name or alternate name. This parameter maps to the *title* and *alternate_title* attributes that are found in the *Identification_Area* class in PDS4 product labels.

4.2 Instance-Specific Parameters

A specific implementation should support additional search parameters beyond the common parameters. The parameters in this section should be supported by the underlying implementation. Unless otherwise stated, each of these parameters may be specified multiple times in a single request. An example of specifying a parameter multiple times can be found in the Syntax section.

Discussions of additional search-related parameters will be covered in more detail in future versions of this document.

4.2.1 query

This parameter enables advanced query criteria specification supporting all of the parameters detailed in the Common Parameters section above, but allowing them to be passed as a single parameter to the underlying implementation. See the Advanced Syntax section below for more details.

This parameter may not be specified multiple times in a single request.

4.3 Control Parameters

The control parameters allow for managing content returned from a request. These parameters may only be specified once in a single request.

4.3.1 Related to Data Discovery

These parameters allow for managing the result set returned from a successful search. An underlying implementation must support pagination of the result set allowing the client to process query results in manageable chunks.

4.3.1.1 start

This parameter allows for specification of the offset in the complete result set where the set of returned products should begin. By default, the value is “0” indicating the first product in the result set.

4.3.1.2 rows

This parameter allows for specification of the size of the result set. This value represents the maximum number of products returned from a successful query for any single request to the underlying implementation. The default page size may be configurable for a given implementation.

4.3.1.3 return-type

This parameter allows for specification of the format of the result set. By default, the format will be an XML structure containing the result set. See the Result Set section below for more details on the supported formats.

4.3.2 Related to Data Access

These parameters allow for managing the files returned from a successful data access request.

4.3.2.1 package

This parameter allows specification of the package format to be returned from the request. By default, the package will be in ZIP format.

4.3.2.2 transform

This parameter allows specification of the file format for files to be transformed into prior to being returned to the calling application.

5.0 SYNTAX

The interface is a REST-based interface over the HTTP protocol. The HTTP protocol is pretty basic in nature with respect to passing parameters. The following details the syntax for passing parameters:

```
http://<host>/<path>[?<parameter>=<value>[&
<parameter>=<value>…]]
```

As shown above the "&" symbol separates multiple clauses (parameter/value pairs) and may be interpreted as AND or OR depending on the underlying implementation. For this protocol, the underlying implementation should interpret it as an AND meaning that each clause must evaluate to true in order for a specific product to be included in the result set. The exception to this rule is when a parameter is passed multiple times in a single request. In this case, the multiple values for a given parameter will be evaluated with OR instead of AND.

A query is transmitted as an HTTP GET or POST request. Depending on who you ask, an HTTP GET request has a limit of 2000 characters in length. Because of this limitation, HTTP POST requests are also supported which effectively increases the limit to 5000 characters.

Parameter names in a request are case sensitive, meaning they should be represented in the case for which they are defined in the Parameters section above. Values on the other hand, are case insensitive (except for the *identifier* parameter). For example, the investigation Cassini is represented as "CASSINI" in a product label. A query request may represent that value in any case (e.g., Cassini, cassini, etc.) and still result in a hit on the original value that was in upper case.

The base URL, depicted as *http://<host>/<path>* in the example above, will vary based on the underlying implementation and its deployment. The main deployment of the Search Service for PDS, hosting the catalog-level search, will have the following as its base URL:

```
http://pds.nasa.gov/services/search
```

For the examples that follow, the base URL is represented as *<base-url>*. The following sections discuss the differences between the simple and advanced syntaxes.

5.1 Simple Syntax

The simple syntax essentially follows the HTTP protocol for passing parameters as detailed in the previous section. The following is an example term-based query request using the simple syntax:

```
<base-url>?term=cassini saturn
```

The above query would return products where the configured attributes associated with the *term* parameter contained either “cassini” or “saturn”. For term-based search, the default operator is “OR”. The following is an example field-based query request using the simple syntax:

```
<base-url>?investigation=cassini&target=saturn
```

The above query would return products where *investigation* is equal to “cassini” and *target* is equal to “saturn”.

The simple syntax also includes support for wildcards when specifying parameter values. Single character wildcard substitution is accomplished using the “?” symbol and multiple character wildcard substitution is accomplished using the “*” symbol. Although wildcards are not that useful when it comes to PDS parameter values, the following example demonstrates its use:

```
<base-url>?investigation=cassini&target=saturn&
start-time=2011-10*
```

The above query would return the same result set as the previous query example but only include products where *start-time* occurs in October 2011.

5.2 Advanced Syntax

Because the simple syntax does not support all of the desired query features, the advanced syntax is also available for specifying query requests. Besides support for additional features, the main difference between the simple and advanced syntaxes is that the query expression is contained in a single parameter. The term-based query from the Simple Syntax section above would be specified as follows:

```
<base-url>?query=cassini OR saturn
```

The difference between the simple and advanced examples is that the values are separated by the Boolean operator “OR”. The field-based query from the Simple Syntax section above would be specified as follows:

```
<base-url>?query=investigation:cassini AND target:saturn
```

The difference between the simple and advanced examples is that the parameter and value are separated by a colon and the clauses are separated by the Boolean operator “AND”.

The advanced syntax provides support for specifying a range for a given parameter value.

```
<base-url>?query=investigation:cassini AND target:saturn
AND start-time:[2011-10-01T00:00:00.000Z TO
2011-10-31T23:59:59.999Z]
```

The above query will return products where *start-time* occurs between the two date/times specified, inclusive. In order to specify an exclusive range, substitute curly brackets “{}” for the square brackets “[]” in the query.

The advanced syntax also supports Boolean operators (AND, OR and NOT) along with grouping of clauses.

```
<base-url>?query=investigation:cassini AND (target:saturn
OR target:titan)
```

The above query will return products where *investigation* is equal to “cassini” and *target* is equal to “saturn” or “titan”.

The control parameters should not be included in the *query* parameter but should be passed as standard HTTP parameters as they would be with the simple syntax.

6.0 RESULT SET

TBD

APPENDIX A ACRONYMS & ABBREVIATIONS

The following acronyms made an appearance in this document:

HTTP	Hypertext Transfer Protocol
IPDA	International Planetary Data Alliance
JPL	Jet Propulsion Laboratory
JSON	JavaScript Object Notation
NASA	National Aeronautics and Space Administration
PDAP	Planetary Data Access Protocol
PDS	Planetary Data System
PDS3	Version 3.8 of the PDS Standards
PDS4	Version 4 of the PDS Standards
REST	Representational State Transfer
SDD	Software Design Document
SRD	Software Requirements Document
URL	Uniform Resource Locator
VOTable	Virtual Observatory Table
XML	Extensible Markup Language