Jet Propulsion Laboratory
California Institute of Technology

Planetary Data System
Client Developer's Pack

PDS Client v.1.0.1

for the Planetary Data System

NASA Jet Propulsion Laboratory 25 April 2006

Table of Contents

1 **User's Guide**

1.1 Introduction 1

1.2 Retrieving Products 3

1.3 Using Profiles 15

1.3.1 Information Captured in a Profile 16

1.3.2 Representation of Profiles 25

1.3.3 Querying Profile Servers 30

©2005 NASA/JPL • ALL RIGHTS RESERVED

©2005 NASA/JPL • ALL RIGHTS RESERVED

1.1 Introduction

PDS Web Services

The Planetary Data System (PDS) Data Distrubution System (PDS-D) is a distributed, network-based, web accessible series of services that enable lookup, description, profiling, and retrieval of planetary data. It's based on the Object Oriented Data Technology 's data grid services . It provides several interfaces to scientists, data analysts, and developers as *web services* using standard Hyper Text Transfer Protocol (HTTP) and Extensible Markup Language (XML) standards.

By providing interfaces using HTTP and XML standards, you can develop applications that use PDS metadata and data from all modern software environments.

Public Web Services

The PDS-D provides two web services:

- `http://starbrite.jpl.nasa.gov/prod`

  This service provides for retrieval of PDS products over HTTP. You pass in parameters that tell what discipline node to query and what product to retrieve. In response, you get PDS products.

- `http://starbrite.jpl.nasa.gov/prod`

  This service provides for resource location and description over HTTP. You pass in parameters that tells what kind of metadata to search and a search expression. You get metadata descriptions (called "profiles") back in XML format.

Both web services are centrally located, as you can see from the URLs, on a system at NASA's Jet Propulsion Laboratory. Back-end software (the data grid framework) automatically directs your query throughout PDS discpline nodes.

PDS Software

The data grid framework that the PDS uses to implement its data systems are all Java-based. As a result, you can take advantage of Java software, included in this package, to simplify some tasks (specifically, profile retrieval). You aren't required to use Java, of course, but several "jar" files are included should you wish to.

The jar files included are:

- `edm-commons`. This jar contains common components that are needed by all the other jar files. You can

©2005 NASA/JPL • ALL RIGHTS RESERVED

always grab the latest version from the Common Components web site.

- `grid-profile`. This jar contains classes for the representation of profiles, which are metadata descriptions of resources. Using these classes means you don't have to work with the XML representation of profiles that come from the `http://starbrite.jpl.nasa.gov/q` web service. You can always grab the latest version from the Profile Service web site .

See the guide section on querying PDS profile servers for instructions on using these two jars. (No jars are provided for product retrieval since Java includes everything necessary for handling products already See the guide section on getting products for more details.)

©2005 NASA/JPL • ALL RIGHTS RESERVED

1.2 Retrieving Products

Getting Products

The PDS Data Distrubution System (PDS-D) uses the OODT framework to transfer products between the various PDS discipline nodes spread throughout the country and the central node. You can access products using any web browser or any program that can speak the Hyper Text Transfer Protocol (HTTP). That means you can also embed product retrieval into your own analyis applications or other programs, since virtually every programming language supports HTTP.

This section details what's involved in using HTTP to get products in PDS-D.

Product Servers

Each of the PDS discipline nodes (and some of the subnodes) runs a *product server* that the engineering node at JPL uses to transfer product data. Product server's have the responsibility of performing any requested conversion of products (such as PDS image format to JPEG image format), lookup of label files, and packaging of sets of product files into ZIP archives.

To retrieve a product via HTTP, you need to know two things:

1. The *object name* of the product server to which you'd like the query directed.
2. The *query expression* that selects what product (or sets of files) to retrieve from the distant product server.

When you present an HTTP query for a product, the engineering node passes that query onto your selected product server, gathers the result, and then streams it back to you over the HTTP conenction.

Object Names

The table below lists each PDS discipline node and the *object name* of the product server that runs there. (Please note that as additional subnodes add product servers this list will be out of date. Check with the PDS engineering node for the latest information.)

Node	Location	Object Name
Atmospheres	New Mexico State University	`urn:eda:rmi:PDS.Atmos.Product`
Central	Jet Propulsion Laboratory	`urn:eda:rmi:PDS.CN.Product`

©2005 NASA/JPL • ALL RIGHTS RESERVED

Node	Location	Object Name
Geosciences	Washington University	`urn:eda:rmi:PDS.Geo.Product`
Imaging 1	Jet Propulsion Laboratory	`urn:eda:rmi:PDS.Img.Product`
Imaging 2	USGS (Arizona)	`urn:eda:rmi:PDS.CN.Product`
NAIF	Jet Propulsion Laboratory	`urn:eda:rmi:PDS.NAIF.Product`
Rings	SETI (California)	`urn:eda:rmi:PDS.Rings.Product`
THEMIS	Arizona State University	`urn:eda:rmi:PDS.ASU.Product`

Note the *object name* in the far right column of the node you want to query. You'll need it to make an HTTP transaction.

Query Expressions

All of the product servers deployed throughout PDS-D expect a query expression that describes a *file or directory path* as well as a *return type*. PDS-D calls the path the `ONLINE_FILE_SPECIFICATION_NAME` or just `OFSN` for short. The return type is called the `RETURN_TYPE` or just `RT` for short. You put these two things together in a query that looks like this:

```
OFSN = path AND RT = type
```

The *path* names a file to retrieve, a directory to list, a label to parse, and so forth. The *type* tells what to do with the *path*. For example, the expression,

```
OFSN = data/clem1-l-h-5-dim-mosaic-v1.0/cl6002/hxxx0190/browse/h79s0190.jpg
AND RT = RAW
```

says to retrieve the file `h79s0190.jpg` and return its contents directly without any processing. The expression

```
OFSN = data/moc/1519.lbl AND RT = PDS_ZIP
```

says to collect all of the files referenced by the label `1519.lbl` and put them into a ZIP archive. Finally, the expression:

```
OFSN = data/clem1-l-h-5-dim-mosaic-v1.0 AND RT = DIRLIST1
```

says to return a directory listing of the files and directories under `clem1-l-h-5-dim-mosaic-v1.0`.

©2005 NASA/JPL • ALL RIGHTS RESERVED

Here is a list of all supported `RETURN_TYPE` or `RT` codes. They're explained at the end of this document.

- DIRFILELIST1
- DIRFILELIST
- DIRLIST1
- DIRLIST
- FILELIST
- FILELISTZIP
- PDS_FILELIST
- PDS_FILELISTZIP
- PDS_JPEG
- PDS_JPEG_SIZE
- PDS_LABEL
- PDS_ZIP
- PDS_ZIPD
- PDS_ZIPD_SIZE
- PDS_ZIPN
- PDS_ZIPN_SIZE
- PDS_ZIPN_TES
- PDS_ZIPN_TES_SIZE
- PDS_ZIP_SIZE
- RAW
- RAW_SIZE

Retrieving Products

Once you know the *object name* and have formed the *query expression*, you're ready to get PDS-D data products. You get PDS-D by accessing a web service at the URL:

```
http://starbrite.jpl.nasa.gov/prod
```

You pass request parameters through that URL that specify the object name of the server to query and the query expression. Here's an example:

```
http://starbrite.jpl.nasa.gov/prod?
object=urn:eda:rmi:PDS.Img.Product&keywordQuery=OFSN+%3D+data/
mgs-m-moc-na_wa-2-dsdp-l0-v1.0/mgsc_0004/sp1223/sp122304.img+AND+
```

©2005 NASA/JPL • ALL RIGHTS RESERVED

```
RT+%3D+PDS_JPEG
```

We've broken this URL across multiple lines for readability, but if you click it, you'll perform a retrieval of a PDS product (an image from Mars Global Surveyor stored at the PDS Imaging Node, converted from the PDS image format into JPEG format).

This interface accepts both HTTP GET and HTTP POST style requests, as detailed in the HTTP specification . GET requests are generally easier; they encode the parameters to a request right in the URL string. POST requests transmit the parameters of the request separately. Most HTTP clients, including web browsers, as well as HTTP APIs in programming languages, support both. For this document, we'll be using GET requests. The choice of which to use is up to you.

Request Parameters

You pass in two request parameters to the `http://starbrite.jpl.nasa.gov/prod` web service:

`object`

Contains the *object name* of the product server to query.

`keywordQuery`

Contains the *query expression*.

In the example query above, the `object` parameter was set to the object name of the Imaging node's product server, `urn:eda:rmi:PDS.Img.Product`. The `keywordQuery` parameter was set to the query expression `OFSN = data/mgs-m-moc-na_wa-2-dsdp-l0-v1.0/mgsc_0004/sp1223/sp122304.img AND RT = PDS_JPEG`.

Request parameters must be properly encoded by the rules of Uniform Resource Identifiers whether they're part of the URL in an HTTP GET or transmitted separately in an HTTP POST. In general, this means that characters like colons in object names as well as spaces and equals signs in query expressions need to be properly escaped. Most browsers and HTTP libraries will happily relax these rules, though, especially in cases where there is no ambiguity. For PDS-D queries, this means changing your spaces to `+`s and your equals signs to `%3D`s.

In the example query above, we got away without escaping the colons in the object name and the slashes in the file path. A more conforming URL might look like the following:

```
http://starbrite.jpl.nasa.gov/prod?
object=urn%3Aeda%3Armi%3APDS.Img.Product&keywordQuery=OFSN+%3D+data%2F
mgs-m-moc-na_wa-2-dsdp-l0-v1.0%2Fmgsc_0004%2Fsp1223%2Fsp122304.img+AND+
RT+%3D+PDS_JPEG
```

While HTTP APIs may relax the rules about what can go in, they often provide utility functions that will perform appropriate escapes for you. Consult your API documentation for more details.

©2005 NASA/JPL • ALL RIGHTS RESERVED

Responses

Responses back from the web service at `http://starbrite.jpl.nasa.gov/prod` contain either the PDS product you've requested or an error message (along with the appropriate HTTP status code) saying what went wrong. For returned products, the MIME type of the data will be appropriate for the type of product returned. For example, if you request an image converted from the PDS format to the JPEG format with the `RETURN_TYPE = PDS_JPEG`, you'll get a `image/jpeg` HTTP response. The directory list style return types all return XML documents, so they use the MIME type `text/xml`.

Querying from Applications

As we've mentioned, you're not limited to retrieving product data solely with a web browser. Most programming languages include either direct or add-on support for doing HTTP. In this way, you can develop analysis tools that retrieve PDS products from the PDS-D online system. To demonstrate this, here is a complete Java program that retrieves and displays a PDS image product in a window:

```
import java.net.URL;
import javax.swing.ImageIcon;
import javax.swing.JFrame;
import javax.swing.JLabel;

public class ShowImage {
  public static void main(String[] argv) throws Throwable {
    URL url = new URL("http://starbrite.jpl.nasa.gov/prod?"
      + "object=urn:eda:rmi:PDS.Img.Product&"
      + "keywordQuery=OFSN+%3D+data/"
      + "mgs-m-moc-na_wa-2-dsdp-l0-v1.0/"
      + "mgsc_0004/sp1223/sp122304.img"
      + "+AND+RT+%3D+PDS_JPEG");
    ImageIcon icon = new ImageIcon(url);
    JLabel label = new JLabel(icon);
    JFrame frame = new JFrame();
    frame.getContentPane().add(label);
    frame.setDefaultCloseOperation(JFrame.
      EXIT_ON_CLOSE);
    frame.pack();
    frame.setVisible(true);
  }
}
```

When compiled and run, a window appears on screen similar to the following:

©2005 NASA/JPL • ALL RIGHTS RESERVED

©2005 NASA/JPL • ALL RIGHTS RESERVED

Return Type Details

This section describes each of the `RETURN_TYPE` or `RT` values you can use with the PDS product servers and gives some example queries for each.

DIRLIST

The `DIRLIST` type returns an XML file with directory names (and only *directory* names) and total size of files in each directory starting in the requested directory and traversing into the subdirectories. An example response looks like:

```
<?xml version="1.0"?>
<!DOCTYPE dirresult PUBLIC "-//JPL/DTD OODT dirresult 1.0//EN"
  "http://starbrite.jpl.nasa.gov:80/dtd/dirresult.dtd">
<dirResult>
  <dirEntry>
    <OFSN>data/ody-m-rss-1-raw-v1.0/odrs_0001</OFSN>
    <fileSize>39360</fileSize>
  </dirEntry>
  <dirEntry>
    <OFSN>data/ody-m-rss-1-raw-v1.0/odrs_0008</OFSN>
    <fileSize>42880</fileSize>
  </dirEntry>
  <dirEntry>
    <OFSN>data/ody-m-rss-1-raw-v1.0/odrs_0002</OFSN>
    <fileSize>35280</fileSize>
  </dirEntry>
</dirResult>
```

Some example queries:

- `http://starbrite.jpl.nasa.gov/prod?object=PDS.CN.Product&keywordQuery=OFSN+%3D+/data/ody-m-rss-1-raw-v1.0/odrs_0001+AND+RT+%3D+DIRLIST`
- `http://starbrite.jpl.nasa.gov/prod?object=PDS.CN.Product&keywordQuery=OFSN+%3D+/data/ody-m-rss-1-raw-v1.0+AND+RT+%3D+DIRLIST`

Beware that requesting recursive directory listings of high-level directories like `data` may take quite some time to generate a response, and can produce huge XML documents that may exhaust your available memory.

DIRLIST1

The `DIRLIST1` type is identical to `DIRLIST` but does not recursively descend into subdirectories. Some example queries:

- `http://starbrite.jpl.nasa.gov/prod?object=PDS.CN.Product&`

©2005 NASA/JPL • ALL RIGHTS RESERVED

keywordQuery=OFSN+%3D+/data/ody-m-rss-1-raw-v1.0/odrs_0001
+AND+RT+%3D+DIRLIST1

- http://starbrite.jpl.nasa.gov/prod?object=PDS.CN.Product&
keywordQuery=OFSN+%3D+/data/ody-m-rss-1-raw-v1.0
+AND+RT+%3D+DIRLIST1

DIRFILELIST

The `DIRFILELIST` returns an XML document containing all of the filenames (and only the files, not directories) and their sizes, recursing into subdirectories. Some example queries:

- http://starbrite.jpl.nasa.gov/prod?object=PDS.CN.Product&
keywordQuery=OFSN+%3D+/data/ody-m-rss-1-raw-v1.0/odrs_0001
+AND+RT+%3D+DIRFILELIST
- http://starbrite.jpl.nasa.gov/prod?object=PDS.CN.Product&
keywordQuery=OFSN+%3D+/data/ody-m-rss-1-raw-v1.0
+AND+RT+%3D+DIRFILELIST

DIRFILELIST1

The `DIRFILELIST1` is identical to `DIRFILELIST` except that it does not recurse into subdirectories. Some example queries:

- http://starbrite.jpl.nasa.gov/prod?object=PDS.CN.Product&
keywordQuery=OFSN+%3D+/data/ody-m-rss-1-raw-v1.0/odrs_0001
+AND+RT+%3D+DIRFILELIST1
- http://starbrite.jpl.nasa.gov/prod?object=PDS.CN.Product&
keywordQuery=OFSN+%3D+/data/ody-m-rss-1-raw-v1.0
+AND+RT+%3D+DIRFILELIST1

FILELIST

The `FILELIST` type returns an XML file naming the requested file and also giving its size. An example query is:

http://starbrite.jpl.nasa.gov/prod?object=PDS.CN.Product&
keywordQuery=OFSN+%3D+/data/ody-m-rss-1-raw-v1.0/
odrs_0001/eop/11352136.lbl+AND+RT+%3D+FILELIST

Example response:

```
<?xml version="1.0"?>
<!DOCTYPE dirresult PUBLIC "-//JPL/DTD OODT dirresult 1.0//EN"
   "http://starbrite.jpl.nasa.gov:80/dtd/dirresult.dtd">
<dirResult>
  <dirEntry>
```

©2005 NASA/JPL • ALL RIGHTS RESERVED

```
    <OFSN>data/ody-m-rss-1-raw-v1.0/odrs_0001/eop/11352136.lbl</OFSN>
    <fileSize>1440</fileSize>
  </dirEntry>
</dirResult>
```

FILELISTZIP

The `FILELISTZIP` is similar to `FILELIST` except that it tells what the size of the requested file would be if compressed into a ZIP archive. An example query:

http://starbrite.jpl.nasa.gov/prod?object=PDS.CN.Product&keywordQuery=OFSN+%3D+/data/ody-m-rss-1-raw-v1.0/odrs_0001/eop/11352136.lbl+AND+RT+%3D+FILELISTZIP

Managing Archives

The return types `PDS_ZIP`, `PDS_FILELIST`, `PDS_ZIP_SIZE`, and `PDS_FILELISTZIP` are all related in that they provide information on the downloaded product and also on the downloaded PDS label file *and its referencees*.

PDS_ZIP

The `PDS_ZIP` returns the PDS label and its references in a ZIP file. An example query is:

http://starbrite.jpl.nasa.gov/prod?object=PDS.CN.Product&keywordQuery=OFSN+%3D+/data/ody-m-rss-1-raw-v1.0/odrs_0001/eop/11352136.lbl+AND+RT+%3D+PDS_ZIP

This query generates a ZIP archive with three files in it:

- `11352136.lbl`, the requested file in the `OFSN`
- `11352136.eop`, which was referenced in the `11352136.lbl` label
- `trk_2_21.txt`, which was also referenced in the `11352136.lbl` label

PDS_FILELIST

The `PDS_FILELIST` returns a file listing of the requested PDS label file and its references in XML format. This is similar to `FILELIST` but instead of giving the size of the requested file, it parses the requested file as a PDS label and gives the sizes of all *referenced* files. An example query is:

http://starbrite.jpl.nasa.gov/prod?object=PDS.CN.Product&keywordQuery=OFSN+%3D+/data/ody-m-rss-1-raw-v1.0/odrs_0001/eop/11352136.lbl+AND+RT+%3D+PDS_FILELIST

PDS_ZIP_SIZE

The `PDS_ZIP_SIZE` returns the total size of the PDS label file and its references in XML format if you

©2005 NASA/JPL • ALL RIGHTS RESERVED

were to download the product with PDS_ZIP and unzip the archive. An example query is:

```
http://starbrite.jpl.nasa.gov/prod?object=PDS.CN.Product&
keywordQuery=OFSN+%3D+/data/ody-m-rss-1-raw-v1.0/odrs_0001/eop/11352136.lbl
+AND+RT+%3D+PDS_ZIP_SIZE
```

PDS_FILELISTZIP

The PDS_FILELISTZIP type returns an XML file showing the size of the zip archive that would be downloaded with PDS_ZIP but not if you expanded the archive. An example query is:

```
http://starbrite.jpl.nasa.gov/prod?object=PDS.CN.Product&
keywordQuery=OFSN+%3D+/data/ody-m-rss-1-raw-v1.0/odrs_0001/eop/11352136.lbl
+AND+RT+%3D+PDS_FILELISTZIP
```

PDS_ZIPN

The PDS_ZIPN type generates and returns a ZIP archive containing the requested file and all of the files that it references. An example query is:

```
http://starbrite.jpl.nasa.gov/prod?object=PDS.CN.Product&
keywordQuery=OFSN+%3D+/data/ody-m-rss-1-raw-v1.0/odrs_0001/eop/11352136.lbl
+AND+RT+%3D+PDS_ZIPN
```

This query generates a ZIP archive with three files in it:

- 11352136.lbl, the requested file in the OFSN
- 11352136.eop, which was referenced in the 11352136.lbl label
- trk_2_21.txt, which was also referenced in the 11352136.lbl label

Contact the PDS Engineering Node for the difference between PDS_ZIP and PDS_ZIPN.

PDS_ZIPN_SIZE

The PDS_ZIPN_SIZE type returns an XML document telling the size of the ZIP archive that would be generated by the corresponding PDS_ZIPN query. An example query is:

```
http://starbrite.jpl.nasa.gov/prod?object=PDS.CN.Product&
keywordQuery=OFSN+%3D+/data/ody-m-rss-1-raw-v1.0/odrs_0001/eop/11352136.lbl
+AND+RT+%3D+PDS_ZIPN_SIZE
```

PDS_ZIPN_TES

The PDS_ZIPN_TES type generates and returns a ZIP archive that contains all files matching the AAAnnnnn format in the same directory as the requsted OFSN, and all of their referenced files. An example query is:

```
http://starbrite.jpl.nasa.gov/prod?object=PDS.CN.Product&
```

©2005 NASA/JPL • ALL RIGHTS RESERVED

```
keywordQuery=OFSN+%3D+data/mgs-m-tes-3-tsdr-v1.0/mgst_0001/data/mars/TST00101
+AND+RT+%3D+PDS_ZIPN_TES
```

The above query would contain 14 files. Note that this type can sometimes take quite a bit of time to generate its ZIP archive, and the archive can often be quite large.

PDS_ZIPN_TES_SIZE

The `PDS_ZIPN_TES_SIZE` type generates an XML document that tells what the size of the ZIP archive would be from the corresponding `PDS_ZIPN_TES` type. An example query is:

```
http://starbrite.jpl.nasa.gov/prod?object=PDS.CN.Product&
keywordQuery=OFSN+%3D+data/mgs-m-tes-3-tsdr-v1.0/mgst_0001/data/mars/TST00101
+AND+RT+%3D+PDS_ZIPN_TES_SIZE
```

PDS_ZIPD

The `PDS_ZIPD` type generates and returns a ZIP archive of all of the files and directories under the given `OFSN`, which should be a directory. For example, the following query:

```
http://starbrite.jpl.nasa.gov/prod?object=PDS.CN.Product&
keywordQuery=OFSN+%3D+data/go-j_jsa-ssi-2-redr-v1.0/go_0017/j0/opnav/c034640
+AND+RT+%3D+PDS_ZIPD
```

Generates a ZIP archive with eight files in it from the various subdirectories under the `c034640` directory.

PDS_ZIPD_SIZE

The `PDS_ZIPD_SIZE` type generates an XML document telling what the size of the generated ZIP archive would be from the corresponding `PDS_ZIPD` type. An example query is:

```
http://starbrite.jpl.nasa.gov/prod?object=PDS.CN.Product&
keywordQuery=OFSN+%3D+data/go-j_jsa-ssi-2-redr-v1.0/go_0017/j0/opnav/c034640
+AND+RT+%3D+PDS_ZIPD_SIZE
```

PDS_JPEG

The `PDS_JPEG` type generates and returns the PDS-format image at the given `OFSN`, converted into the JPEG/JFIF image format. An example query is:

```
http://starbrite.jpl.nasa.gov/prod?
object=urn:eda:rmi:PDS.Img.Product&keywordQuery=OFSN+%3D+data/
mgs-m-moc-na_wa-2-dsdp-l0-v1.0/mgsc_0004/sp1223/sp122304.img+AND+
RT+%3D+PDS_JPEG
```

©2005 NASA/JPL • ALL RIGHTS RESERVED

PDS_JPEG_SIZE

The `PDS_JPEG_SIZE` type generates and returns an XML document telling what the size of the JPEG file would be generated by the corresponding `PDS_JPEG` query. An example query is:

```
http://starbrite.jpl.nasa.gov/prod?
object=urn:eda:rmi:PDS.Img.Product&keywordQuery=OFSN+%3D+data/
mgs-m-moc-na_wa-2-dsdp-l0-v1.0/mgsc_0004/sp1223/sp122304.img+AND+
RT+%3D+PDS_JPEG_SIZE
```

PDS_LABEL

The `PDS_LABEL` type examines the given `OFSN` and extracts any PDS format label it can find. The label is returned as plain text (MIME type `text/plain`). An example query is:

```
http://starbrite.jpl.nasa.gov/prod?object=PDS.CN.Product&
keywordQuery=OFSN+%3D+data/near-a-mag-3-rdr-erosflyby-v1.0/nmerfb_2001/
data/1998_350/ebf98357.lbl+AND+RT+%3D+PDS_LABEL
```

RAW

The `RAW` type merely returns the requested file without any modification (MIME type `application/octet-stream`). An example query is:

```
http://starbrite.jpl.nasa.gov/prod?object=PDS.CN.Product&
keywordQuery=OFSN+%3D+data/near-a-mag-3-rdr-erosflyby-v1.0/
nmerfb_2001/data/1998_350/ebf98357.lbl+AND+RT+%3D+RAW
```

RAW_SIZE

The `RAW_SIZE` generates an XML document telling the size of the raw file that would be returned by the corresponding `RAW` type. An example query is:

```
http://starbrite.jpl.nasa.gov/prod?object=PDS.CN.Product&
keywordQuery=OFSN+%3D+data/near-a-mag-3-rdr-erosflyby-v1.0/
nmerfb_2001/data/1998_350/ebf98357.lbl+AND+RT+%3D+RAW_SIZE
```

©2005 NASA/JPL • ALL RIGHTS RESERVED

1.3 Using Profiles

Using PDS Profiles

The PDS Data Distrubution System (PDS-D) uses the OODT framework to distribute *profiles* of PDS resources. Profiles are metadata descriptions of resources. The PDS-D runs several profile servers, each of which are responsible for serving profiles for resources like instruments, hosts, targets, and so forth.

To successfully use profiles within the PDS, you'll need to understand:

- The information captured by profiles
- The representation of profiles
- Querying profiles within the PDS

©2005 NASA/JPL • ALL RIGHTS RESERVED

1.3.1 Information Captured in a Profile

Information Captured by Profiles

A profile serves as a generic template for describing the characteristics of a resource. Within the PDS, profiles exist to describe resources that include mission hosts, instruments, Mars datasets, Lunar datasets, and more. This document describes the shape that information takes in profiles.

Information and Organization

Profiles capture three kinds of information:

- **Resource Attributes**

  Resource attributes are metadata about the resource's *inception*. These attributes include the creator of the resource, in what language it exists, when it was created, and so forth. These attributes are based on the work of the Dublin Core Metadata Initiative .

- **Profile Elements**

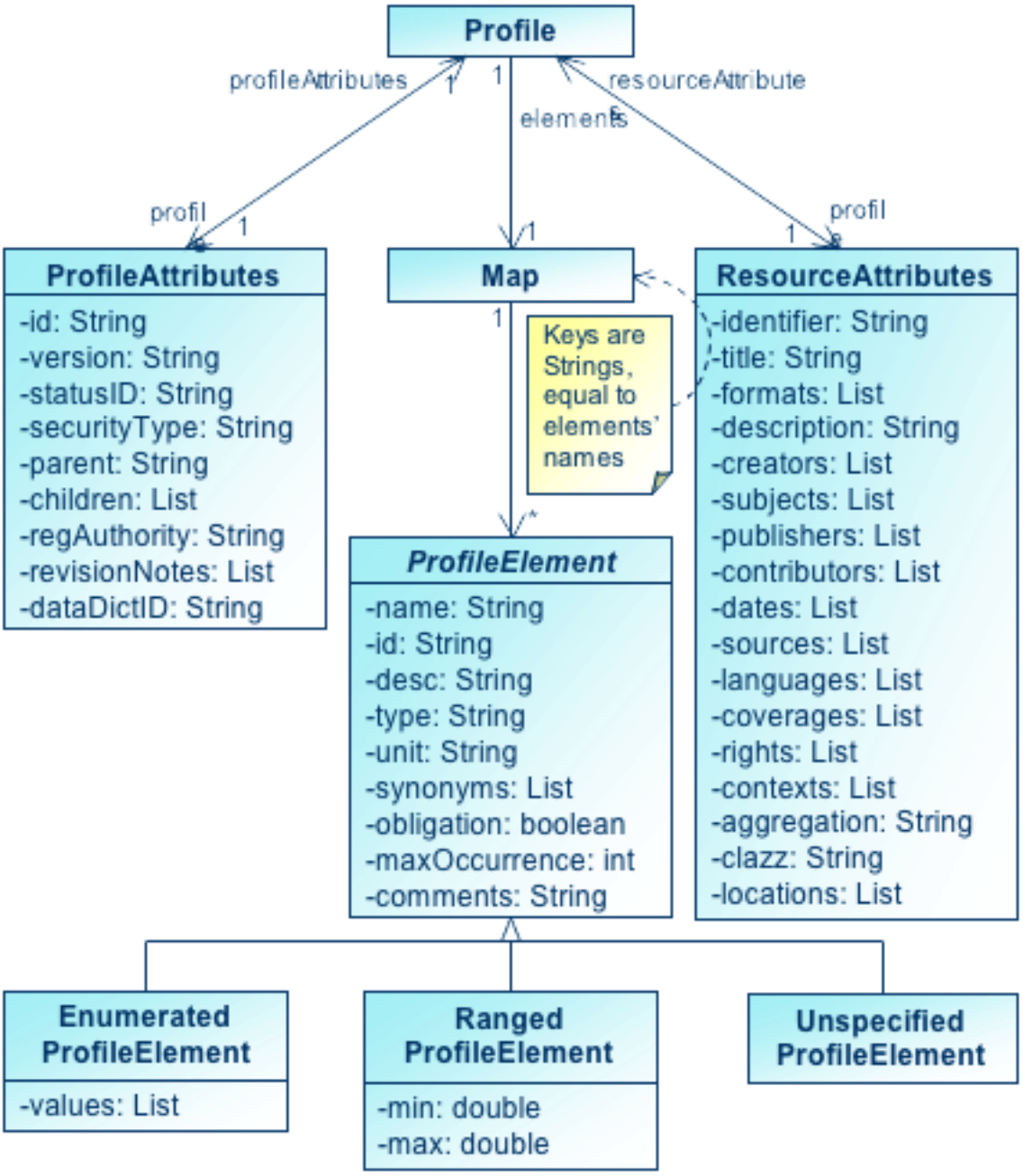
  Profile elements are metadata about the resource's *composition*. These tell you about the morphology of the resource, such as data types captured within in, minimum and maximum values, synonymous elements, and so forth. These attributes are based on ISO/IEC 11179 standards .

- **Profile Attributes**

  Profile attributes are metadata about the *profile itself*, such as who made it, whether it's classified, revision notes, and so forth. It also has a unique identifying Object Identifier (OID) .

The following class diagram shows the relationship between the different parts of a profile:

©2005 NASA/JPL • ALL RIGHTS RESERVED



While this diagram shows the Java field names and Java classes, the relationship applies to profiles whether they exist as Java objects, as RDF documents, or as XML documents in the profile vocabulary.

Inception Metadata

Profiles, whether expressed in RDF or in their own XML vocabulary, have a section for capturing information about the resource's inception. This includes information about when the resource was created,

©2005 NASA/JPL • ALL RIGHTS RESERVED

who created it, in what language it exists, and so forth. Profiles use the element set recommended by the Dublin Core Metadata Initiative (DCMI) set in order to describe the inception of a resource, with some extensions.

Collectively, these metadata are called the *resource attributes* or `resAttributes` of the profile. Every profile has one and only one set of `resAttributes`. The metadata elements within the `resAttributes` are defined in this section.

Identifier

As defined by the DCMI, the `Identifier` of a resource is some unambiguous way to identify the resource. In the profile implementation, one and only one `Identifier` is *required*.

Title

The `Title` names the resource, and is the name by which the resource is formally known. The `Title` is optional; if present, it may occur only once in a profile.

Format

The `Format` indicates the manifestation of the resource. The MIME type is usually recorded here.

Description

The `Description` element contains a free text account of the content of the resource. It's optional in a profile; if present, it may occur only once.

Creator

Zero or more `Creator`s may be specified in a profile. `Creator`s contain the name of people or organizations that created the resource.

Subject

Zero or more `Subject`s in a profile act as keywords. The purpose of the `Subject` elements is to contain a keywords that describe the resource, usually selected from a controlled vocabulary.

Publisher

Any number of `Publisher` elements may appear in a profile. They contain the organization responsible for making the resource available.

©2005 NASA/JPL • ALL RIGHTS RESERVED

Contributor

A `Contributor` is a person or organization providing auxilliary work towards the resource's creation. Any number of `Contributor`s may be listed in a profile.

Date

`Date` elements indicate the times in history when the resource was created. Any number of `Date`s may be included in a profile.

Type

The `Type` element indicates the nature of the content of the resource, such as "fiction" for a work of fiction or "image" for a dataset rendered graphically.

Source

When a resource is derived others, the `Source` element indicates the `Identifier`s of the referenced resources.

Language

For resources that contain natural language content, the `Language` element indicates the languages in use.

Relation

When a resource is related to others, a profile can specify the `Identifier`s of the related resources using zero or more `Relation` elements.

Coverage

For resources that cover a space or time or jurisdiction, use the `Coverage` element to indicate such coverage. This element may be listed any number of times in a profile, and its content usually comes from a controlled vocabulary. In the PDS, profile elements are usually used for this purpose, though.

Rights

Copyright, ownership, redistribution, use, and other legal issues may exist for a resource. When that happens, the `Rights` element lists the rights management information.

Note: The official name of element for is plural `Rights`; this is inconsistent with the other metadata elements, but is consistent with the DCMI.

©2005 NASA/JPL • ALL RIGHTS RESERVED

resContext

The `resContext` element identifies the application environment or discipline within which the resource originates and is derived from a taxonomy of scientific disciplines. This element is required in a profile and may occur multiple times.

As an example, a `resContext` of `NASA.PDS.Geoscience` tells that the resource is associated with the Geoscience node of the Planetary Data System.

resAggregation

The `resAggregation` element indicates the aggregative structure of the resource. It tells you what you'll get if you retrieve the resource: a granule, a dataset, or a collection of datasets. The legal values of this optional elements are:

- `granule`, meaning the resource is a single product
- `dataSet`, meaning the resource is a set of products
- `dataSetCollection`, meaning the resource is collection of datasets

The `resAggregation` element is optional; however, if specified, it may appear in a profile only once.

resClass

The `resClass` element identifies the kind of the resource within a taxonomy of resource types. It's a *required* element that is used by the OODT Framework to determine how to treat the profile as well as the resource named by the profile.

For example, a `resClass` of `system.productServer` indicates that the resource is an OODT product server. A query that matches this profile means that if the same query were given to the identified product server, it would yield a result. A `resClass` of `system.profileServer` means the resource is a profile server. That means that while the current profile server may or may not provide a matching profile, another profile server might, forming an implicit digraph of profile servers. Other valid `resClass` values include `data.granule`, `data.dataSet`, and `application.interface`.

resLocation

Zero or more `resLocation` elements may appear in a profile. They tell where the resource is located, easily the most important part of the profile. Because this element may appear several times, all locations should be considered valid; the application may pick the one that's most convenient. The `resLocation` may also appear zero times. This means that the profile indicates solely that the resource exists, but where is unknown.

The interpretation of the resLocation is as a URI. For example, a `resClass` of `system.productServer` or `system.profileServer` means that the `resLocation` indicates an URN to a software object name. Querying that object will yield either the desired result (for product servers) or more matching profiles (for

©2005 NASA/JPL • ALL RIGHTS RESERVED

profile servers). For a resClass of `data.granule` or `data.dataSet`, the `resLocation` is an URL to the granule or dataset.

Composition Metadata

The most interesting part of a profile is in the metadata that describes the composition of the resource that the profile profiles. The composition metadata is what enables a profile server to tell if a particular resource can answer a query.

The composition metadata is based on the data element description standards in ISO/IEC standard 11179. They are the *profile elements* or `profElement`s of a profile. Every profile may have zero or more `profElement`s, the components of which are discussed in this section.

elemId

The `elemId` is an optional universally unique identifier applied to the element.

elemName

The `elemName` is the *required* name of the profile element. It serves as the title role of one of the components of the resource.

elemDesc

The `elemDesc` is the description of the profile element. Although the title may often be enough to identify the purpose of the profile element, the description should be used to provide any further, free-text information that may be of importance to analysts and profile administrators. The PDS uses the `elemDesc` field extensively just as it does in PDS labels.

elemType

The `elemType` indicates the type of data represented in the profile element, synonymous to the ISO/IEC 11179 `Datatype` attribute. The permissible values are:

- `boolean`
- `character`
- `date_time`
- `enumerated`
- `integer`
- `ordinal`
- `rational`

©2005 NASA/JPL • ALL RIGHTS RESERVED

- `scaled`
- `real`
- `complex`
- `state`
- `void`

This element is optional within a profile element. When it's not present, the profile element merely indicates that the resource's content possesses the attribute, but more is not known.

elemUnit

The `elemUnit` indicates the units associated with the values of the data element. This element is synonymous to the ISO/IEC 11179 attribute `unit.of.quantity`. Values for this optional element should be selected from standardized tables of units.

elemEnumFlag, elemValue, elemMinValue, and elemMaxValue

The `elemEnumFlag` tells how possible values of the profile element are specified. It works with the `elemValue`, `elemMinValue`, and `elemMaxValue` elements:

- If the `elemEnumFlag`'s value is `T` and one or more `elemValue`s appear, then the values listed are the valid values of the element.
- If the value is `F`, then a closed range of values bounded by the profile's `elemMinValue` and `elemMaxValue` elements indicates the valid values.
- If the value is `T` but no `elemValue`s appear, then it means that any value is a valid value for the resource.

elemSynonym

Often, a characteristic of a resource will go by several names, especially between scientific disciplines. What one person may call *latitude*, another may call *x coordinate*, for example.

The `elemSynonym` provides a way to indicate synonyms. Zero or more `elemSynonym`s may appear in a profile element. The values of this element are names from data dictionaries other than the discipline data dictionary hosting the profile.

elemComment

The `elemComment` field provides a remark concerning the application of the data element. This element is synonymous to the ISO/IEC 11179 attribute `Comment`, and is optional within a profile element.

Metadata about the Profile

©2005 NASA/JPL • ALL RIGHTS RESERVED

For a profile server to manage a set of profiles, it's necessary to have metadata contained within the profile that describes the profile itself. This metadata, collectively called the profile attributes, or `profAttributes`, serves that purpose.

Most of the elements within the `profAttributes` are optional. This sections describes each of them.

profId

The `profId` serves to give a unique identifier to the profile. It may be expressed as a URI, and often as an URN, or as an OID.

profVersion

The `profVersion` identifies the version number of the profile.

profType

The `profType` identifies the type of the profile. The type that typically appears here is `profile`, meaning the profile is a profile (obviously).

Another type that can be here is `dataDict`, which indicates that the profile doesn't describe a resource, but instead is a data dictionary for other profiles. Such a profile's composition elements name the expected profile elements and ranges of valid valuese that will appear in other profiles. The `profDataDictId` element identifies the profile serving as its data dictionary.

profStatusId

The `profStatusId` identifies the state of the profile. Profiles may be either `active` or `inactive`. An inactive profile is likely maintained for historical or exemplary reasons but is otherwise not currently used for searches or resource descriptions.

profSecurityType

The `profSecurityType` identifies whether the information contained in the profile may be of a sensitive nature. Any string is valid here as the PDS does not use this field.

profParentId

The `profParentId` optionally identifies the URI of the parent of this profile. Profiles may be arranged hierarchically in a singly rooted tree in a forest.

profChildId

©2005 NASA/JPL • ALL RIGHTS RESERVED

The `profChildId` identifies zero or more children (by duplicating the element) of this profile.

profRegAuthority

The `profRegAuthority` names the registration authority responsible for authoring and maintaining the profile.

profRevisionNote

The `profRevisionNote` appears zero or more times in the profile to describe changes made to it over time. The notes are free form text, and each element is ordered from newest to oldest note.

profDataDictId

The `profDataDictId` identifies the profile providing a data dictionary to the current profile.

©2005 NASA/JPL • ALL RIGHTS RESERVED

1.3.2 Representation of Profiles

Representation of PDS Profiles

As you've surmised by now, Profiles exist as Java objects within the PDS-D. However, they can also be represented as RDF documents or XML documents. When you query the PDS-D web service, you'll receive them as XML documents, from which you can then re-construct Java objects, or manipulate them directly, if you're not using Java.

XML Profiles

Profiles can be represented as XML documents that conform to the OODT Profile Document Type Definition (DTD). The Formal Public Identifier of the OODT Profile DTD is `-//JPL//DTD Profile 1.0//EN`. The normative System Identifier is http://oodt.jpl.nasa.gov/grid-profile/dtd/prof.dtd.

Although you should refer to the normative System Identifier for the latest reference version, see the following for the DTD:

```
<!ELEMENT profiles
  (profile*)>

<!ELEMENT profile
  (profAttributes,
   resAttributes,
   profElement*)>

    <!ELEMENT profAttributes
      (profId, profVersion?, profType,
       profStatusId, profSecurityType?, profParentId?, profChildId*,
       profRegAuthority?, profRevisionNote*)>

    <!ELEMENT resAttributes
      (Identifier, Title?, Format*, Description?, Creator*, Subject*,
       Publisher*, Contributor*, Date*, Type*, Source*,
       Language*, Relation*, Coverage*, Rights*,
       resContext+, resAggregation?, resClass, resLocation*)>

    <!ELEMENT profElement
      (elemId?, elemName, elemDesc?, elemType?, elemUnit?,
       elemEnumFlag, (elemValue* | (elemMinValue, elemMaxValue)),
       elemSynonym*,
       elemObligation?, elemMaxOccurrence?, elemComment?)>

    <!ELEMENT profId (#PCDATA)>
```

©2005 NASA/JPL • ALL RIGHTS RESERVED

```
<!ELEMENT profVersion (#PCDATA)>
<!ELEMENT profType (#PCDATA)>
<!ELEMENT profParentId (#PCDATA)>
<!ELEMENT profChildId (#PCDATA)>
<!ELEMENT profStatusId (#PCDATA)>
<!ELEMENT profSecurityType (#PCDATA)>
<!ELEMENT profRegAuthority (#PCDATA)>
<!ELEMENT profRevisionNote (#PCDATA)>

<!ELEMENT Identifier (#PCDATA)>
<!ELEMENT Title (#PCDATA)>
<!ELEMENT Format (#PCDATA)>
<!ELEMENT Description (#PCDATA)>
<!ELEMENT Creator (#PCDATA)>
<!ELEMENT Subject (#PCDATA)>
<!ELEMENT Publisher (#PCDATA)>
<!ELEMENT Contributor (#PCDATA)>
<!ELEMENT Date (#PCDATA)>
<!ELEMENT Type (#PCDATA)>
<!ELEMENT Source (#PCDATA)>
<!ELEMENT Language (#PCDATA)>
<!ELEMENT Relation (#PCDATA)>
<!ELEMENT Coverage (#PCDATA)>
<!ELEMENT Rights (#PCDATA)>
<!ELEMENT resContext (#PCDATA)>
<!ELEMENT resAggregation (#PCDATA)>
<!ELEMENT resClass (#PCDATA)>
<!ELEMENT resLocation (#PCDATA)>

<!ELEMENT elemId (#PCDATA)>
<!ELEMENT elemName (#PCDATA)>
<!ELEMENT elemType (#PCDATA)>
<!ELEMENT elemEnumFlag (#PCDATA)>
<!ELEMENT elemDesc (#PCDATA)>
<!ELEMENT elemSynonym (#PCDATA)>
<!ELEMENT elemUnit (#PCDATA)>
<!ELEMENT elemValue (#PCDATA)>
<!ELEMENT elemMinValue (#PCDATA)>
<!ELEMENT elemMaxValue (#PCDATA)>
<!ELEMENT elemObligation (#PCDATA)>
<!ELEMENT elemMaxOccurrence (#PCDATA)>
<!ELEMENT elemComment (#PCDATA)>
```

Collections of Profiles

As you can see from the above,an XML element, `profiles` is a container element to hold zero or more profiles. Responses from PDS servers always start with this element.

Java Representation of Profiles

The OODT source code includes a class `jpl.eda.profile.Profile` for object represenation of a

©2005 NASA/JPL • ALL RIGHTS RESERVED

profile. You can construct a `Profile` object from an XML document or create a blank one to populate with metadata later. See the Profile Javadocs for more information.

Accessing Profile Metadata

To access the metadata of a profile, call the methods to retrieve the profile attributes, the resource attributes, or the profile elements.

Accessing the Profile Attributes

You retrieve the profile attributes by calling `getProfileAttributes` on a `Profile`. This returns an `ProfileAttributes` object which provides methods to get and set the various attributes. Setting a value sets it for the `Profile` to which the `ProfileAttributes` belongs.

A value of `null` for an optional attribute means the value isn't set.

Accessing the Resource Attributes

You retrieve the resource attributes similarly as for profile attributes, calling `getResourceAttributes` to yield a `ResourceAttributes` object.

The `ResourceAttributes` has methods to get and set various attributes. Note that many of the attributes are multi-valued. For example, the resource profiled will likely cover several subjects. In this case, the "get" method, `getSubjects`, returns a `java.util.List` of `String`s. There is no set method. Instead, you just manipulate the list to add and remove subjects.

Note: All of the "get" functions that return `List`s return lists of `String`s, except for `getDates`, which returns a list of `java.util.Date`s.

For other attributes which are singly valued, there is both a set and get method. For optional values, a value of `null` means the attribute isn't set.

Accessing the Profile Elements

The profile stores its profile elements in a `java.util.Map`, mapping the name of the profile element (as a `String`) to an object of class `ProfileElement`. To access this map, call the method `getProfileElements` on a `Profile`. Because this method returns a reference to the `Profile`'s map, any updates to the map affect the profile immediately.

Caution: Never store anything but `String`s as keys and `ProfileElement`s (or objects of its subclasses) as values in a profile element map. The software will not operate correctly if any other kind of object is stored.

Common Attributes of Profile Elements

The class `jpl.eda.profile.ProfileElement` contains the common parts of every profile element,

©2005 NASA/JPL • ALL RIGHTS RESERVED

such as its required name, its optional description, and so forth. Use the value `null` for any optional attribute that's unset.

The `ProfileElement` stores its synonyms as a `java.util.List` of `String`s. Manipulate the list directly to add or remove synonyms.

The `ProfileElement` class is abstract. To create new profile elements for a profile, you need to create objects of one of the concrete `ProfileElement` subclasses:

- `EnumeratedProfileElement`
- `RangedProfileElement`
- `UnspecifiedProfileElement`

The following sections detail each kind of profile element.

Elements with Enumerated Values

For profile elements that maintain a specified list of valid values, use the `EnumeratedProfileElement` class. Objects of this class maintian a `java.util.List` of values. You can pass in a list of values when constrcuting the object, or can call the `getValues` method and manipulate the list directly.

Queries that arrive for an enumerated profile element must match one of the listed elements exactly unless it's a negative (not-equal-to) query. For example, suppose we had an enumerated profile element `filter` with values `infrared`, `visible`, and `ultraviolet`. A query asking for a filter equal to infrared should match, as well as a query asking for a filter greater than or equal to infrared. A query asking for a filter not equal to infrared shouldn't match, while a query asking for a filter not equal to x-ray should match.

Elements with a Range of Values

You can represent profile elements that have a range of valid values with the `RangedProfileElement` class. Construct this class with the minimum and maximum values, which must be numeric.

Use the `getMinValue` to get the minimum value and `getMaxValue` to get the maximum value.

Querying a profile that uses a range of values considers the range as inclusive. For example, suppose the ranged profile element `temperature` has a minimum value of 32 and a maximum value of 212. A query that requests temperatures less than 32 shouldn't match, but less than or equal to 32 should match.

Elements with no Specified Values

For profile elements that you always want to match a query without explicitly listing each valid value or a range of legal numeric values you can use the class `UnspecifiedProfileElement` . This class identifies an element with no range or list of valid values.

Queries that arrive at such an element will *always* match, even if they're negative (not-equal-to) queries.

©2005 NASA/JPL • ALL RIGHTS RESERVED

©2005 NASA/JPL • ALL RIGHTS RESERVED

1.3.3 Querying Profile Servers

Querying PDS Profile Servers

To retrieve a profile of a resource, you send a query to a PDS web service. It, in turn, passes it to your selected profile server, which queries its data stores and synthesizes profile objects. Finally, the web service returns those matching profile objects in an XML document. You can then examine and manipulate this XML document directly or construct Java profile objects out of it and manipulate them.

The Query Web Service

Retrieving profiles via HTTP uses a web service available at the PDS single point of entry website:

`http://starbrite.jpl.nasa.gov/q`

Here the "q" means "query," and it enables you to pass queries into the OODT framework serving the PDS-D. Profile queries will respond with XML documents representing a set of matching profiles.

Making Requests

To retrieve profiles using this web service, you need to know two things:

1. The *object name* of the profile server to which you'd like the query directed.
2. The *query expression* that selects what profiles to retrieve.

The object name selects a single profile server from the various profile servers available. The query expression selects profiles. Different profile servers accept different expressions and each serve different kinds of metadata profiles. The table below summarizes the available profile servers.

Server	Object Name
Catalog	`urn:eda:rmi:JPL.PDS.Profile`
Host	`urn:eda:rmi:JPL.PDS.Host`
Instrument	`urn:eda:rmi:JPL.PDS.Instrument`
Lunar	`urn:eda:rmi:PDS.LunarDB.Profile`
Mars	`urn:eda:rmi:PDS.MarsDB.Profile`

©2005 NASA/JPL • ALL RIGHTS RESERVED

Server	Object Name
Master Product	`urn:eda:rmi:JPL.PDS.MasterProd`
Mission	`urn:eda:rmi:JPL.PDS.Mission`
Profile of Profiles	`urn:eda:rmi:PDS.ProfProf.Profile`
Target	`urn:eda:rmi:JPL.PDS.Target`

The web service at `http://starbrite.jpl.nasa.gov/q` supports both HTTP GET and HTTP POST style requests to retrieve profiles, as detailed in the HTTP specification . GET requests are generally easier; they encode the parameters to a request right in the URL string. POST requests transmit the parameters of the request separately. Most HTTP clients, including web browsers, as well as HTTP APIs in programming languages, support both. For this document, we'll be using GET requests. The choice of which to use is up to you.

When calling the web service at `http://starbrite.jpl.nasa.gov/q`, you need to pass in three parameters:

type

The type tells what kind of query you're making. This should always be set to `profile`.

object

This tells what profile server to query; it should be one of the values under *Object Name* from the table above.

keywordQuery

This is the query expression to pass to the profile server

As with all most web services, you'll need to encode your parameters according to the rules in Uniform Resource Identifiers whether they're part of the URL in an HTTP GET or transmitted separately in an HTTP POST. In general, this means that characters like colons in object names as well as spaces and equals signs in query expressions need to be properly escaped. Most browsers and HTTP libraries will happily relax these rules, though, especially in cases where there is no ambiguity. For PDS-D queries, this means changing your spaces to `+`s and any equals signs to `%3D`s.

Here's an example query (broken across multiple lines for readability):

```
http://starbrite.jpl.nasa.gov/q?
type=profile&
object=urn:eda:rmi:JPL.PDS.Profile&
keywordQuery=resclass+%3D+resource
+AND+dsid+%3D+ODY-M-ACCEL-5-ALTITUDE-V1.0
+AND+RETURN+%3D+resourceid
```

If you're click on that query (and are using an XML-capable browser), you'll see an XML document returned

©2005 NASA/JPL • ALL RIGHTS RESERVED

containing three matching profiles. With such a document, you can scan it for interesting resources, pass it through an XSL stylesheet and present it prettily to a user, or even create corresponding language-specific objects out of it, such as Java objects.

From XML to Java

If you're not into XML, you might find it more convenient to work with Java objects that represent profiles and their attributes. Creating Java objects out of the XML document returned by the `http://starbrite.jpl.nasa.gov/q` web service is really quite simple. First, make the URL to query the service just as above. Open the URL and parse the response document into a Document Object Model (DOM) tree. Then call the createProfiles method on the root element of the DOM tree.

Let's look at a specific example.

```
import java.util.Iterator;
import java.util.List;
import java.util.Map;
import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;
import jpl.eda.profile.Profile;
import jpl.eda.profile.ProfileAttributes;
import jpl.eda.profile.ProfileElement;
import jpl.eda.profile.ResourceAttributes;
import org.w3c.dom.Document;

public class UseProfiles {
  public static void main(String[] argv) throws Throwable {
    String url = "http://starbrite.jpl.nasa.gov/q?"
      + "type=profile&object=urn:eda:rmi:JPL.PDS.Profile"
      + "&keywordQuery=resclass+%3D+resource+"
      + "AND+dsid+%3D+ODY-M-ACCEL-5-ALTITUDE-"
      + "V1.0+AND+RETURN+%3D+resourceid";
    DocumentBuilderFactory fac = DocumentBuilderFactory.newInstance();
    DocumentBuilder builder = fac.newDocumentBuilder();
    Document doc = builder.parse(url);
    List profiles = Profile.createProfiles(doc.getDocumentElement());
    for (Iterator i = profiles.iterator(); i.hasNext();) {
      Profile p = (Profile) i.next();
      ProfileAttributes a = p.getProfileAttributes();
      ResourceAttributes r = p.getResourceAttributes();
      System.out.println("PROFILE " + a.getID() + ":");
      System.out.println("\tResource title: " + r.getTitle());
      List locs = r.getResLocations();
      System.out.println("\tLocations (" + locs.size() + "):");
      for (Iterator j = locs.iterator(); j.hasNext();)
        System.out.println("\t\t" + j.next());
      Map elems = p.getProfileElements();
      System.out.println("\tElements (" + elems.size() + "):");
      for (Iterator j = elems.values().iterator(); j.hasNext();) {
        ProfileElement e = (ProfileElement) j.next();
        System.out.println("\t\tName: " + e.getName());
        System.out.println("\t\tLegal values: " + e.getValues());
        System.out.println("\t\tMin value: " + e.getMinValue());
```

©2005 NASA/JPL • ALL RIGHTS RESERVED

```
        System.out.println("\t\tMax value: " + e.getMaxValue());
      }
    }
  }
}
```

This program sends a query to the web service and parses the response into an in-memory DOM tree. It then creates a `java.util.List` of `Profile` objects out of that tree, and iterates through the list, printing some details about each `Profile`.

To compile and run this program, you'll need a few other components:

- Profile Service , which contains definitions for classes such as `Profile`, `ProfileAttributes`, and so forth.
- OODT Common Components , which contains basic utilities used by all other OODT components.
- Jena v.1.6.1 , a Resource Description Framework library.

Since Java tools and integrated development environments vary, consult your own documentation on how to make your system aware of these components. If you're using the Java command-line tools, then download each of the binary distributions of the above components and copy the jar file from each into a directory, say `lib`. Then put `UseProfiles.java` into a directory called `src`. You can then compile `UseProfiles.java` into a directory called `classes` and run the class:

```
% ls -l lib
total 344
-rw-r--r--   1 kelly  kelly   154210 Dec  1 16:54 edm-commons-2.2.9.jar
-rw-r--r--   1 kelly  kelly   165823 Jan 12 12:42 grid-profile-3.0.7.jar
-rw-r--r--   1 kelly  kelly  1144107 Apr 25 07:56 jena-1.6.1.jar
% ls -l src
total 4
-rw-r--r--  1 kelly  kelly  1950 12 Mar 07:42 UseProfiles.java
% mkdir classes
% javac -extdirs lib -d classes src/UseProfiles.java
% ls -l classes
total 4
-rw-r--r--  1 kelly  kelly  2810 12 Mar 07:42 UseProfiles.class
% java -Djava.ext.dirs=lib -classpath classes UseProfiles
PROFILE 1:
          Identifier: ...
```

Using the PDS Profile Servers

The PDS Data Distribution System runs *nine* profile servers, each serving different kinds of metadata and each responding to different kinds of queries. In this section, we'll cover each one.

©2005 NASA/JPL • ALL RIGHTS RESERVED

Catalog Profile Server

The Catalog Profile Server contains metadata about datasets, volumes, resources, releases, targets, and other information. To query the Catalog Profile Server, you pass in the class of the resource you want and then an identifier telling which resource out of that class. The object name of the Catalog Profile Server is `urn:eda:rmi:JPL.PDS.Profile`.

Host Profile Server

The Host Profile Server contains metadata descriptions of *hosts*. The job of hosts is to host instruments; hosts consist of spacecraft and other components participate in a mission by providing a host environment for instruments and their measurements. The Host Profile Server returns profiles of hosts.

The object name of the Host Profile Server is `urn:eda:rmi:JPL.PDS.Host`.

Valid query expressions for the Host Profile Server include:

- `INSTRUMENT_HOST_TYPE = type` , where *type* can be `N/A`, `UNK`, `DATA BASE`, `SPACECRAFT`, or `EARTH BASED`. For example, the query

  ```
  http://starbrite.jpl.nasa.gov/q?
  type=profile&
  object=urn:eda:rmi:JPL.PDS.Host&
  keywordQuery=INSTRUMENT_HOST_TYPE+%3D+SPACECRAFT
  ```

  returns 50 profiles each describing a spacecraft-based host.

- `INSTRUMENT_HOST_ID = id` , where *id* is a host ID like `AAO`, `CLEM1`, `IRAS`, `ODY`, and so forth. For example, the query

  ```
  http://starbrite.jpl.nasa.gov/q?type=profile&
  object=urn:eda:rmi:JPL.PDS.Host&
  keywordQuery=INSTRUMENT_HOST_ID+%3D+ODY
  ```

  returns a profile describing the Mars Odyssey Spacecraft and Deep Space Network systems.

- `INSTRUMENT_HOST_NAME = name` , where *name* is a host name, like `PIONEER`, `VOYAGER 1`, `MARS OBSERVER`, and so forth. For example, the query

  ```
  http://starbrite.jpl.nasa.gov/q?type=profile&
  object=urn:eda:rmi:JPL.PDS.Host&
  keywordQuery=INSTRUMENT_HOST_NAME+%3D+%222001 MARS ODYSSEY%22
  ```

  returns the same query as above, but queries by name instead of by ID.

- `RETURN = DATADICTIONARY`. This returns a profile that describes all of the searchable parameters (INSTRUMENT_HOST_TYPE, INSTRUMENT_HOST_ID, INSTRUMENT_HOST_NAME) and other profile elements that appear in profiles returned by this server.

©2005 NASA/JPL • ALL RIGHTS RESERVED

Instrument Profile Server

The Instrument Profile Server contains metadata descriptions of *instruments*. Instruments are the scientific apparatus that take measurements. They reside within host environments and are part of a mission.

The object name of the Instruments Profile Server is `urn:eda:rmi:JPL.PDS.Instrument`.

Valid query expressions for the Host Profile Server include:

- `INSTRUMENT_TYPE = type` , where *type* can be `CCD`, `RADAR`, `ELECTROSTATIC ANALYZER`, `THERMAL INFRARED SPECTROMETER`, and so forth. For example, the query

  ```
  http://starbrite.jpl.nasa.gov/q?type=profile&
  object=urn:eda:rmi:JPL.PDS.Instrument&
  keywordQuery=INSTRUMENT_TYPE+%3D+CCD
  ```

  returns two profiles each describing a CCD instruments.

- `INSTRUMENT_ID = id` , where *id* is an ID like `IIRAR`, `SWOOPS`, `WINDSOCK`, `RCAC31034A`, and so forth. For example, the query

  ```
  http://starbrite.jpl.nasa.gov/q?type=profile&
  object=urn:eda:rmi:JPL.PDS.Instrument&
  keywordQuery=INSTRUMENT_ID+%3D+NAVCAM
  ```

  returns a profile describing the Mars Exploration Rover Navigation Camera.

- `INSTRUMENT_NAME = name` , where *name* is an instrument name, like `PHOTOMETER`, `RADIO TELESCOPE`, `VERY LARGE ARRAY`, and so forth. For example, the query

  ```
  http://starbrite.jpl.nasa.gov/q?type=profile&
  object=urn:eda:rmi:JPL.PDS.Instrument&
  keywordQuery=INSTRUMENT_NAME+%3D+%22NAVIGATION CAMERA%22
  ```

  returns the same query as above, but queries by name instead of by ID.

- `INSTRUMENT_HOST_ID = id` , where *id* is a host ID.
- `RETURN = DATADICTIONARY`. This returns a profile that describes all of the searchable parameters (INSTRUMENT_TYPE, INSTRUMENT_ID, INSTRUMENT_NAME, INSTRUMENT_HOST_ID) and other profile elements that appear in profiles returned by this server.

Lunar Profile Server

The Lunar profile server serves information about Lunar datasets. It uses metadata like mission names, map projection types, instrument IDs, and so forth, to profile and locate lunar datasets. Most importantly, it returns a resource location (a product query) that yields the matching dataset.

The object name of the Lunar Profile Server is `urn:eda:rmi:PDS.LunarDB.Profile`. You can query

©2005 NASA/JPL • ALL RIGHTS RESERVED

the lunar profile server using the following parameters:

- CENTER_LONGITUDE
- DATA_SET_ID
- EASTERNMOST_LONGITUDE
- FILE_SPECIFICATION_NAME
- Format
- INSTRUMENT_ID
- Identifier
- MAP_PROJECTION_TYPE
- MAP_RESOLUTION
- MAP_SCALE
- MAXIMUM_LATITUDE
- MINIMUM_LATITUDE
- MISSION_NAME
- OID
- PRODUCT_ID
- SPACECRAFT_ID
- TARGET_NAME
- Title
- VOLUME_ID
- WESTERNMOST_LONGITUDE
- profId
- profStatusId
- profType
- resClass
- resContext
- resLocation

This example query finds the single dataset CLEM1-L-U-5-DIM-BASEMAP-V1.0:BI87S030:
http://starbrite.jpl.nasa.gov/q?type=profile&
object=urn:eda:rmi:PDS.LunarDB.Profile&
keywordQuery=Title+%3D+CLEM1-L-U-5-DIM-BASEMAP-V1.0:BI87S030 .

Mars Profile Server

The Mars profile server serves information about Martian datasets. It uses metadata like mission names, map

©2005 NASA/JPL • ALL RIGHTS RESERVED

projection types, instrument IDs, and so forth, to profile and locate Mars datasets. Most importantly, it returns a resource location (a product query) that yields the matching dataset.

The object name of the Mars Profile Server is `urn:eda:rmi:PDS.MarsDB.Profile`. You can query the lunar profile server using the following parameters:

- `CENTER_LONGITUDE`
- `DATA_SET_ID`
- `FILE_SPECIFICATION_NAME`
- `Format`
- `IMAGE_ID`
- `INSTRUMENT_ID`
- `Identifier`
- `MAP_RESOLUTION`
- `MAP_SCALE`
- `MAXIMUM_LATITUDE`
- `MAXIMUM_LONGITUDE`
- `MINIMUM_LATITUDE`
- `MINIMUM_LONGITUDE`
- `MISSION_NAME`
- `OID`
- `PRODUCT_ID`
- `SPACECRAFT_ID`
- `TARGET_NAME`
- `Title`
- `VOLUME_ID`
- `profId`
- `profStatusId`
- `profType`
- `resClass`
- `resContext`
- `resLocation`

This example query finds the single dataset `VO1/VO2-M-VIS-5-DIM-V1.0:MK88S225`:
http://starbrite.jpl.nasa.gov/q?type=profile&
object=urn:eda:rmi:PDS.MarsDB.Profile&
keywordQuery=Title+%3D+VO1/VO2-M-VIS-5-DIM-V1.0:MK88S225 .

©2005 NASA/JPL • ALL RIGHTS RESERVED

Master Product Profile Server

The Master Product Profile Server serves information about every dataset cataloged by the PDS. It uses metadata like emission angle, maximum latitude, mission name, sub-solar azimuth, and so forth to profile and locate datasets. Most importantly, it returns a resource location (a product query) that yields the matching dataset.

The object name of the Master Product Profile Server is `urn:eda:rmi:JPL.PDS.MasterProd`. You can query the the server using the following parameters:

- `CENTER_LONGITUDE`
- `CENTER_RING_RADIUS`
- `CENTRAL_BODY_DISTANCE`
- `CITATION_DESC`
- `COMPRESSION_TYPE`
- `CONE_ANGLE`
- `CUBE_VOLUME_ID`
- `DATA_SET_ID`
- `DECLINATION`
- `EASTERNMOST_LONGITUDE`
- `EMISSION_ANGLE`
- `EXPOSURE_DURATION`
- `FILE_SPECIFICATION_NAME`
- `FILTER_NAME`
- `FILTER_NUMBER`
- `FRAME_DURATION`
- `Format`
- `GAIN_MODE_ID`
- `HORIZONTAL_PIXEL_SCALE`
- `IMAGE_ID`
- `IMAGE_TIME`
- `INCIDENCE_ANGLE`
- `INSTRUMENT_HOST_ID`
- `INSTRUMENT_ID`
- `INTERCEPT_POINT_LATITUDE`
- `INTERCEPT_POINT_LONGITUDE`

©2005 NASA/JPL • ALL RIGHTS RESERVED

- Identifier
- LOCAL_HOUR_ANGLE
- MAP_PROJECTION_TYPE
- MAP_RESOLUTION
- MAP_SCALE
- MAXIMUM_LATITUDE
- MAXIMUM_LONGITUDE
- MINIMUM_LATITUDE
- MINIMUM_LONGITUDE
- MISSION_NAME
- NORTH_AZIMUTH
- NTV_SAT_TIME_FROM_CLOSEST_APR
- NTV_TIME_FROM_CLOSEST_APPROACH
- OBSERVATION_ID
- OBSTRUCTION_ID
- OID
- ORBIT_NUMBER
- PHASE_ANGLE
- PRODUCT_ID
- PRODUCT_NAME
- PRODUCT_TYPE
- RIGHT_ASCENSION
- SLANT_DISTANCE
- SMEAR_MAGNITUDE
- SOLAR_DISTANCE
- SPACECRAFT_CLOCK_START_COUNT
- SPACECRAFT_ID
- START_TIME
- STOP_TIME
- SUB_SOLAR_AZIMUTH
- SUB_SOLAR_LATITUDE
- SUB_SOLAR_LONGITUDE
- SUB_SPACECRAFT_LATITUDE

©2005 NASA/JPL • ALL RIGHTS RESERVED

- SUB_SPACECRAFT_LONGITUDE
- TARGET_CENTER_DISTANCE
- TARGET_NAME
- TWIST_ANGLE
- Title
- VERTICAL_PIXEL_SCALE
- VOLUME_ID
- WESTERNMOST_LONGITUDE
- profId
- profStatusId
- profType
- resClass
- resContext
- resLocation

This example query finds the single dataset GO-J/JSA-SSI-4-REDR-V1.0:29N0034:
http://starbrite.jpl.nasa.gov/q?type=profile&
object=urn:eda:rmi:JPL.PDS.MasterProd&
keywordQuery=Title+%3D+GO-J/JSA-SSI-4-REDR-V1.0:29N0034 .

Mission Profile Server

The Mission Profile Server contains metadata descriptions of *missions*. Missions are the scientific endeavors that result in observations and measurements.

The object name of the Mission Profile Server is urn:eda:rmi:JPL.PDS.Mission.

Valid query expressions for the Mission Profile Server include:

- MISSION_START_DATE = *date* , where *date* is of the form YYYY, YYYY-MM, or YYYY-MM-DD. For example, the query

  http://starbrite.jpl.nasa.gov/q?type=profile&
  object=urn:eda:rmi:JPL.PDS.Mission&
  keywordQuery=MISSION_START_DATE+%3D+1997-10-01

  returns a profile describing the Galileo mission.

- MISSION_STOP_DATE = *date* , where *date* is of the form YYYY, YYYY-MM, or YYYY-MM-DD. For example, the query

  http://starbrite.jpl.nasa.gov/q?type=profile&
  object=urn:eda:rmi:JPL.PDS.Mission&

©2005 NASA/JPL • ALL RIGHTS RESERVED

```
keywordQuery=MISSION_STOP_DATE+%3D+2003-09-21
```

returns a profile describing the Galileo mission.

- `MISSION_NAME_OR_ALIAS = name` , where *name* is the name or alias of a mission. For example, the query

  ```
  http://starbrite.jpl.nasa.gov/q?type=profile&
  object=urn:eda:rmi:JPL.PDS.Mission&
  keywordQuery=MISSION_NAME_OR_ALIAS+%3D+CASSINI
  ```

  returns a profile describing the Cassini mission.

- `MISSION_NAME = name` , where *name* is the name of a mission.
- `RETURN = DATADICTIONARY`. This returns a profile that describes all of the searchable parameters (MISSION_START_DATE, MISSION_STOP_DATE, MISSION_NAME_OR_ALIAS, MISSION_NAME) and other profile elements that appear in profiles returned by this server.

Profile of Profiles Server

This is identical to the Master Product Profile Server, except that its object key is `urn:eda:rmi:PDS.ProfProf.Profile`.

Target Profile Server

The Target Profile Server contains metadata descriptions of *targets*. Targets are planets and other objects that are measured by instruments.

The object name of the Target Profile Server is `urn:eda:rmi:JPL.PDS.Target`.

Valid query expressions for the Target Profile Server include:

- `TARGET_NAME = name` where *name* is the name of a target, like `1055 TYNKA` or `JUPITER`. For example, the query

  ```
  http://starbrite.jpl.nasa.gov/q?type=profile&
  object=urn:eda:rmi:JPL.PDS.Target&
  keywordQuery=TARGET_NAME+%3D+MARS
  ```

  returns a single profile describing the planet Mars.

- `ORBIT_DIRECTION = dir` where *dir* is `N/A`, `UNK`, `UNKNOWN`, `PROGRADE`, or `RETROGRADE`.
- `TARGET_TYPE = type` where *type* is a type like `SUN`, `RING`, `COMET`, `PLANET`, and so forth.
- `ROTATION_DIRECTION = dir` where *dir* is `N/A`, `UNK`, `UNKNOWN`, `PROGRADE`, `RETROGRADE`, or `SYNCHRONOUS`.
- `PRIMARY_BODY_NAME = name` where *name* is the name of a primary body like `SUN`, `EARTH`, `MARS`, `CERES`, and so forth.

©2005 NASA/JPL • ALL RIGHTS RESERVED

- `RETURN = DATADICTIONARY`. This returns a profile that describes all of the searchable parameters and other profile elements that appear in profiles returned by this server.

©2005 NASA/JPL • ALL RIGHTS RESERVED