

# A Distributed Data Architecture for 2001 Mars Odyssey Data Distribution

Daniel J. Crichton  
Jet Propulsion Laboratory  
Dan.Crichton@jpl.nasa.gov

J. Steven Hughes  
Jet Propulsion Laboratory  
Steve.Hughes@jpl.nasa.gov

Sean Kelly  
Independent Consultant  
Sean.Kelly@jpl.nasa.gov

## Abstract

*Newer instruments and communications techniques have given scientists unprecedented amounts of data, more than can be feasibly distributed through traditional methods such as mailed CD-ROMs. Leveraging the web makes sense since it enables scientists to request specific data and retrieve products as soon as they're available. Yet defining the middleware system to support such an application has remained just out of reach, until Odyssey. For the first time ever, data from all Odyssey mission instruments were made available through a single system immediately upon delivery to the Planetary Data System (PDS).*

*The Object Oriented Data Technology (OODT) software made such an application possible.*

## 1. Introduction

The Planetary Data System (PDS) has had a long tradition of using metadata. Metadata, which is literally data about data, describes the shape and meaning of data. Metadata enables both human researchers and automated systems to determine if a certain piece of data is the right piece, what operations may be done on it, whether it is of value for the current course of research, how it relates to other data, and so forth. What the PDS lacked was the middleware to handle that metadata [8].

Middleware, which is literally software components that glue backend and frontend systems together, was exactly what the PDS needed. The Object Oriented Data Technology (OODT) task at NASA's Jet Propulsion Laboratory had been developing a software framework for handling data and metadata in a uniform way. This framework acts as a middleware system, enabling

backend data systems to plug in and share their data with frontend systems providing access. The middleware encapsulates differing representations, formats, locations, and meanings of data, making data interoperable and relieving researchers of requiring foreknowledge of localized data system organization and representation.

Using the OODT Framework, PDS developed a next-generation data distribution system that was online and web accessible, transcending the traditional CD-ROM distribution model [7].

## 2. Location Independence

The first step towards making data independent of its physical location and storage formats is to describe the data. The OODT Framework requires client components to use *profiles* to describe data [3]. Profiles literally profile resources, providing a usable description of a resource. A resource in this context can mean any electronically addressable item, be it a granule of data, a dataset, a document, an image, a web page, a software service, and so forth.

By profiling what's available, OODT software components can answer queries about what exists and to what things it may be similar. Profile servers handle queries for profiles and manipulate collections of profiles in order to answer the question, "Where is *X*?"

## Profile Representation

Profiles contain three specific kinds of metadata. First is metadata about the profile itself. This metadata includes information such as who made the profile, whether it's classified, what revisions were made, and so forth. While not directly useful to researchers, it serves to provide auditing and other maintenance information.

Second is *inception* metadata. Inception metadata describes a resource's creation. It includes knowledge of who created the resource, who contributed to it, what temporal and physical periods it covers, its title, description, keywords, and so forth. Rather than attempt to define overlapping and redundant metadata elements to

describe a resource's inception, we chose to adopt the metadata elements created by the Dublin Core Metadata Initiative [1]. Originally targeted for online libraries, the Dublin Core metadata elements have wide reusability and have been adopted by a large number of organizations.

Third is the *composition* metadata. Composition metadata describes the shape of the resource. This includes data elements that may occur within it, the limits of values of those elements, specific occurrences of characteristics or other features. As an example, a table of temperature readings should include *temperature* as a compositional metadata element, as well as the ordinate value against which temperature was measured (time, location, etc.). Included in the description of temperature would be the units of temperature and the range of values represented in the table.

To define composition metadata, we turned to the ISO/IEC 11179 standard for describing metadata elements (a metametadata standard) [6]. Such a standard indicates what information must be captured to make description of metadata elements useful, such as units, representation, synonyms with other elements, legal values, and so forth.

Table 1 includes the list of all information captured in a profile.

**Table 1. Metadata captured in a profile**

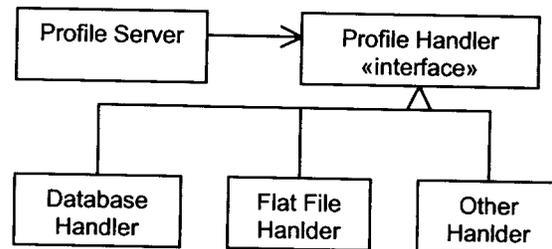
Profile Metadata	
Profile ID	ID of parent profile
Version	IDs of child profiles
Type	Registration authority
Status	Revision notes
Security	
Inception Metadata	
Identifier	Sources
Title	Languages
Formats	Relations
Description	Coverage
Creators	Rights
Subjects (keywords)	Contexts
Publishers	Aggregation
Contributors	Class
Dates	Locations
Types	
Composition Metadata	
Element ID	Min/max values
Element name	Synonyms
Element description	Obligatory
Data type of element	Max occurrence
Units	Comments
Legal values	

### Profile Servers

A profile server is the software component that handles profile queries and manages collections of profiles. For the PDS, we deployed profile servers that were equipped with profile metadata about data collected at the nodes of

the PDS. Profile servers answer queries from the PDS distribution web application about where data is located and provides researchers with overviews about what's available, leading them to further queries for actual data.

Profile servers are accessible using a number of communications protocols as well as software APIs in Java and C/C++. Profile servers themselves are implemented in Java. Using Java's interface feature, we define the interface that a source of profiles must implement (called a ProfileHandler) and can then provide specialized handlers for certain tasks. Figure 1 depicts the delegation. For example, profiling individual granules of a dataset means much repeated information (principal investigator who created the data, name of the instrument that sourced it, and so forth); a query handler specific to that dataset could generate the static inception and composition metadata while varying the composition metadata as necessary for each granule.



**Figure 1. UML diagram of profile server and several profile handlers.**

### 3. Format Independence

Using profiles and profile servers, the PDS data distribution web application can provide researchers with details about what's available. Retrieving those resources, however, requires a second component of the OODT framework. That component is the product service [3].

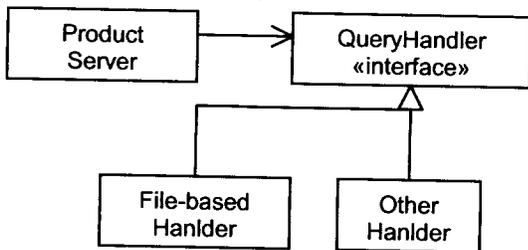
The product service consists of a series of product servers that are accessed and queried exactly like profile servers. In fact, they use the same query expression. However, instead of yielding metadata, they yield data.

Product servers have two responsibilities. One is to provide *access* to data at a curating node in such a way that storage at that node is not impacted. Much in the same way an operating system device driver encapsulates access to a device, a product server encapsulates access to the specific storage *mechanism*, providing an interface to the framework. A node of the PDS specializing in emissivity data, for example, may continue to ingest and store that data in relational database tables or in some other node-specific way that is convenient to the investigators. Using a plug-in architecture, a product server installed at that node must access that relational structure (or other proprietary or node-specific

mechanism). Once accessed, it can yield the data in an Internet standard format for sharing amongst all the OODT Framework and its users.

That's the second responsibility of product servers: to *convert* any node-specific storage *format* into a neutral format. Conversion currently is to Internet standard formats specified by MIME types [5]. Product servers receive a query for a desired product along with a list of acceptable MIME types. Emissivity tables may be converted into the text/tab-separated-value type, and proprietary image formats into the image/tiff or other type, for example.

The plug-in architecture allows many such mechanism and format conversions to be swapped in and out within a single product server, even at run time. Figure 2 depicts this delegation relationship.



**Figure 2. UML diagram of product server and two query handler implementations.**

For the PDS, our first deployment included a product server capable of yielding raw product files, PDS labels, and a variety of ZIP archives. Because many of the nodes use a PDS-mandated file system structure, it was simple to define a file-based query handler. More query handlers are under development now that provide image conversion capabilities as well as retrieval of products from non-file system sources.

#### 4. Query Representation

Profile and product servers use the same query expression. Termed an XMLQuery (due to the fact that it can be represented as an XML document), this query expression provides a uniform way of addressing a user's desired data.

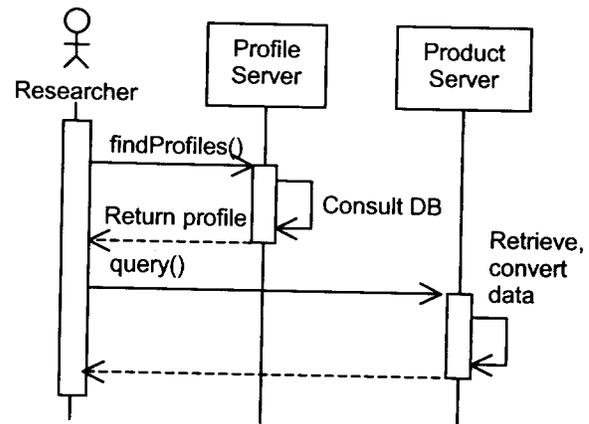
The XMLQuery tracks several sets of aspects of a query: the domain (termed the "from" set), the range (termed the "select" set), and a constraint (termed the "where" set). Each set is a collection of elements that form a boolean expression that determine what's being queried. This neutral query format is used internally by the OODT framework and passed to and from profile and product servers.

Query handlers installed in each profile and product server have the responsibility of understanding the query in a way appropriate to their underlying source of

metadata and data. As an example, a query handler in a product server delivering products from a relational database may translate the boolean expression in the "where" set into the "where" clause of an SQL expression.

The query expression is general enough to support most kinds of queries for scientific data. With this generality comes complexity. Although framework users may construct such boolean expression sets in an XMLQuery directly, it is far easier to write the query as a string. As such, the XMLQuery software class includes a facility for parsing such a string in as a keyword=value-style expression and generating the "from," "select," and "where" sets appropriately. Future facilities may include creating XMLQuery objects from SQL expressions, RDF query expressions, or other expressions.

With an XMLQuery in hand, a typical interaction that yields data takes a two step process: one is to pass the XMLQuery to a network of profile servers. The profile servers examine the query and yield profiles that describe product servers that can satisfy the query. The second step is to pick a product server and pass the *same* XMLQuery to it. The product server then returns the actual data product. Figure 3 depicts this interaction.



**Figure 3. UML diagram of profile/product two-step query.**

#### 5. User Interface

Working with PDS developers, OODT developers created the frontend application that links to the middleware and provides access to the backend data. The application is a web application providing a single point of entry on the public Internet to anyone with a web browser. The address of the application is <http://starbrite.jpl.nasa.gov/pds/>. Figure 4 shows a portion of a screen shot from a browser visiting the web application's quick search page.

Users select mission, target name, instrument, or other criteria to see what data's available. The framework handles this step using profile queries. The user can select resources or drill down and retrieve products. The framework handles this step using product queries.

The underlying mechanisms involving profile queries to locate the data and product queries to retrieve the data are not directly revealed in the web interface, although the two-step query strategy is visible.

In the interaction diagram in Figure 3, the role of the "Researcher" is replaced by the web application, which performs the resource location step and the product retrieval step on behalf of the users. Furthermore, the web application has HTTP interfaces that enable any language that can manipulate HTTP to access the OODT Framework's profile and product servers.

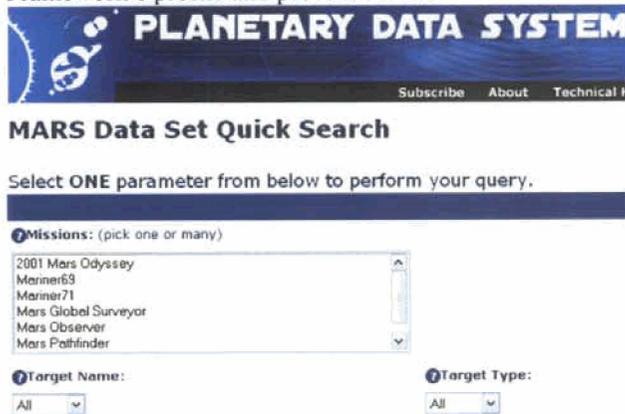


Figure 4. Screen shot of PDS web application.

## 6. Communication

OODT Framework components use Internet standard protocols in order to communicate queries and gather results between each other. The Framework supports multiple, concurrent protocols as well, encapsulating communications layer apart from metadata and data manipulation.

### Metadata and Data Exchange

In order to communicate results, we take a view that makes framework components remote objects and invoke methods on those objects remotely.

For example, finding matching profile servers means calling the profile server's remote findProfiles method, passing an XMLQuery object, and retrieving a list of matching Profile objects. Queries to product servers are similar.

The underlying communications protocol can be based on CORBA (using IIOP [10]) or Java RMI. For PDS, we're currently using Java RMI for its lighter weight and faster performance.

For secure metadata and data exchange, we also support both IIOP and RMI over TLS [4].

### Remote Management

Installation of new plug-ins for product servers and profile servers, debugging of framework components, and starting and stopping servers at remote PDS nodes requires a remote access by OODT developers and PDS central node administrators. However, remote management was further hampered by heterogeneous system environments at each node as well as various security restrictions. As a result, we bundled remote management features directly into the OODT framework.

Because the remote management components will have to administer and debug the communications protocols used by the operational framework components, the system had to use a separate communications system. We turned to XML-RPC [9] for its simplicity and the fact that it runs over HTTP. We use HTTP authentication to protect access to the remote systems as well as to assign various roles to various users. Further, the capabilities of the remote management components go only as far as that of the system environment afforded to it. For example, several sites choose to run the system with a specific user ID that can only write files that comprise the software itself.

Figure 5 shows a partial screen shot of a client of the remote management features. This graphic client is used by PDS administrators and developers to check on the status of remote servers, start and stop processes, and upgrade the system.

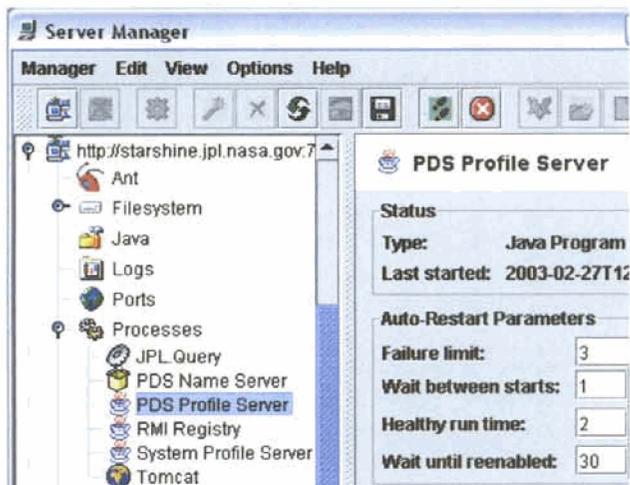


Figure 5. Screen shot of the remote management user interface.

## 7. 2001 Mars Odyssey Data Distribution

The development of an online data distribution system for the 2001 Mars Odyssey mission was initiated when it was realized that it would be cost prohibitive to distribute the large volumes of data expected on CD/DVD media. In particular the Thermal Emission Spectrometer (THEMIS) instrument was expected to produce an estimated 4 terabytes of data. Requirements gathering for the Planetary Data System Distribution system (PDS-D) began at the end of 2001 with the goal of supporting the first Odyssey data release in October 2002. The use of the OODT software allowed the design of a multi-tiered architecture that met the key development requirements including that the distribution system provide seamless search and retrieval of data products from distributed heterogeneous data repositories, support online access as the primary method of data distribution, and leverage the existing PDS resources and capabilities with minimal impact to the existing data system. PDS-D successfully supported the first release of Odyssey data in October, providing the first ever distribution of planetary science data as it was released to the public.

The first delivery of the system, PDS-D D01, supported the distribution of 14 data sets from THEMIS, Gamma Ray Spectrometer (GRS), Martian Radiation Environment Experiment (MARIE), Accelerometer, Radio Science, and navigation ancillary information. Users have access to the data through the existing Planetary Atlas and several new "default" data set browsers. Six product servers were installed at remote sites, including a THEMIS product server at Arizona State University (ASU). The ASU installation allowed the THEMIS data to be distributed from the instrument's team

site, precluding the need to transfer the data set to the PDS for distribution. An additional product server was also installed for a large disk repository at the PDS central node for backup. This was successfully used on several occasions to address Internet performance problems and machine down times. One profile server was installed to provide supporting high-level formation from the central data set catalog, including an inventory of useful Web resources. A subscription/notification service was implemented to allow planetary scientists to subscribe and receive an email notification when data was released. The notification included URLs that linked to the appropriate data set browser.

## 8. Conclusions

The OODT Framework for metadata-based middleware served the PDS in its next-generation data distribution efforts, modernizing a million-dollar CD-ROM distribution effort into a web based, scaleable, metadata-driven system.

And yet the framework is not specific to the PDS at all. As a general purpose metadata/data middleware based on standards, it's suitable for correlation, discovery, and exchange of any kind of data. We deployed the *identical* middleware software to a cancer biomarkers program run under the auspices of the National Cancer Institute and the National Institutes of Health [2]. The same software makes differing specimen databases located across the country appear as a single, vast tissue bank, improving correlative capabilities for cancer research.

We continue to develop the metadata middleware by exploring ways to improve the metadata descriptions as well as peer-to-peer protocols for metadata/data exchange.

## 9. References

- [1] DCMI, "Dublin Core Metadata Element Set, Version 1.1: Reference Description," Dublin Core Metadata Initiative, 1999.
- [2] D. Crichton et al., "An Interoperable Data Architecture for Data Exchange in a Biomedical Research Network," 14<sup>th</sup> IEEE International Symposium on Computer-Based Medical Systems, Bethesda, 2001.
- [3] D. Crichton et al., "Science Search and Retrieval using XML," CODATA, Washington DC, 2000.
- [4] T. Dierks, C. Allen, "The TLS Protocol," Internet Society, Reston 1999, RFC2246.
- [5] N. Freed, N. Borenstein, "Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types," Internet Society, Reston, 1996, RFC2046.

- [6] ISO/IEC, "Framework for the Specification and Standardization of Data Elements 11179-1," *Specification and Standardization of Data Elements 11179*, International Organization for Standardization, Geneva, 1999.
- [7] S. Kelly, D. Crichton, J.S. Hughes, "Deploying Object Oriented Data Technology to the Planetary Data System," 34<sup>th</sup> LPSC, Houston, 2003, p.1607.
- [8] S. Slavney, R.E. Arvidsen, E.A. Guinness, "Mars Global Surveyor and 2001 Mars Odyssey Science Data Archives," 33<sup>rd</sup> LPSC, Houston, 2002, p.1303.
- [9] D. Winer, "XML-RPC Specification," UserLand 1999.
- [10] "Common Object Request Broker: Architecture and Specification," Object Management Group, Lexington, 2000.