

## Packaging Data Products using Data Grid Middleware for Deep Space Mission Systems

Chris A. Mattmann<sup>1</sup>, Paul M. Ramirez<sup>1</sup>, Daniel J. Crichton<sup>1</sup>, J. Steven Hughes<sup>1</sup>

<sup>1</sup>Jet Propulsion Laboratory, California Institute of Technology, Pasadena, California, 91109, U.S.

### Abstract

Deep Space Mission Systems (DSMS) lack the capability to provide end to end tracing of mission *data products*. These data products are *simple products* such as telemetry data, processing history, and uplink data. Additionally, there is a need to create *Complex products* (packages) which can contain one or more of the *simple products*, along with descriptive information about the package itself. The ability to track both types of these data products is crucial for providing visibility of in-process status along the data product pipeline between space and ground. Other applications of this information include measuring performance of the space-to-ground data product pipeline and the timeliness of operational activities, which are key metrics in mission-critical, large-scale embedded systems that exist in multi-node space-to-ground missions. In order to facilitate this tracing, information about the product format and the observation (*metadata*) can be defined, captured, and cataloged per data product in the pipeline. Science teams can then use the cataloged metadata to discover the location and format of data products currently in the pipeline to engage in work earlier, and with a better understanding of the data products themselves.

There are certain challenges; however, in producing a system that can meet these requirements, namely (1) The distributed nature of the mission data products makes it very difficult to catalog and trace along their respective lifecycle in the space to ground pipeline. (2) Both the data product formats, and the system interfaces which expose the data products are heterogeneous in nature and thus very difficult to integrate. (3) No clear methodology or standard exists to describe a process for data product packaging, including the format of the package, and what metadata should be stored about the package itself.

At the National Aeronautics and Space Administration's Jet Propulsion Laboratory, we are addressing this very problem. We are investigating a distributed computing infrastructure for releasing science products out of the mission pipeline. We are creating a DSMS Product Service and DSMS Data Packaging Service based on the OODT middleware [1], a Data Grid [2] middleware technology that has proven successful in providing access to heterogeneous, disparate data, and federating that data using a common *data model*. We seek to provide the capability to (1) Package the distributed data products returned from nodes in the DSMS pipeline (nodes can be spacecraft, satellite, ground stations, etc.), and create package metadata attached to the product package itself (2) Conform our data product model to a packaging standard, such as the one proposed by CCSDS [3] and (3) Catalog Products in the DSMS pipeline so that they can be retrieved, and packaged. Early results from this work suggest that it is a feasible approach to address the challenges described above.

### Motivation

Large amounts of data are produced whenever a request is issued for a particular spacecraft instrument to make some observation of a science event. This data is typically engineering data, containing information about the status of the original observation request, and typically ignored by those not involved in the day to day operations of a particular space mission. This data; however, can be exploited

to provide science users, those typically not involved in the mission operations, but involved from an *end-user* perspective of using the data, a sort of status update on the sequence of events generated for a particular observation request. This status data can even be used in some cases as early processing information, and provide scientists with the capability to make discoveries faster, detect software errors earlier and understand the processing history of a particular data product from its inception in the observation request, to its eventual release to the scientists during the data distribution phase of the space-to-ground pipeline. Figure 1 presents a conceptual view of the flow of data in such a system.

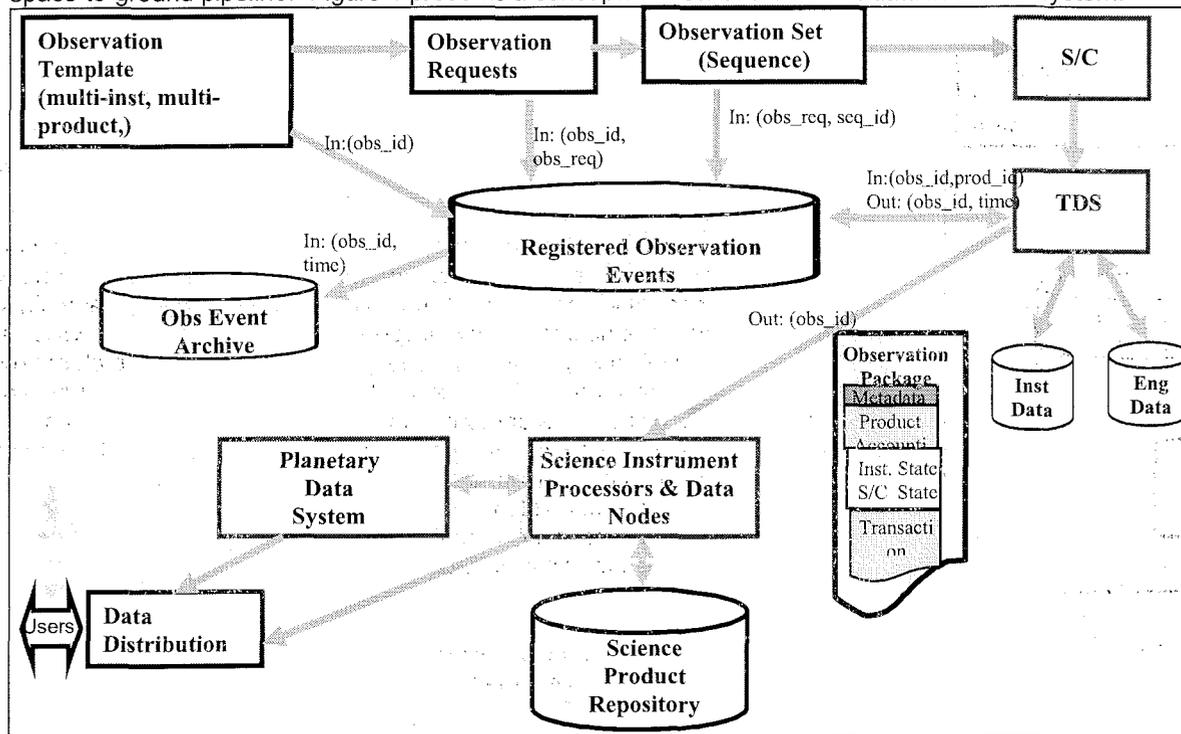
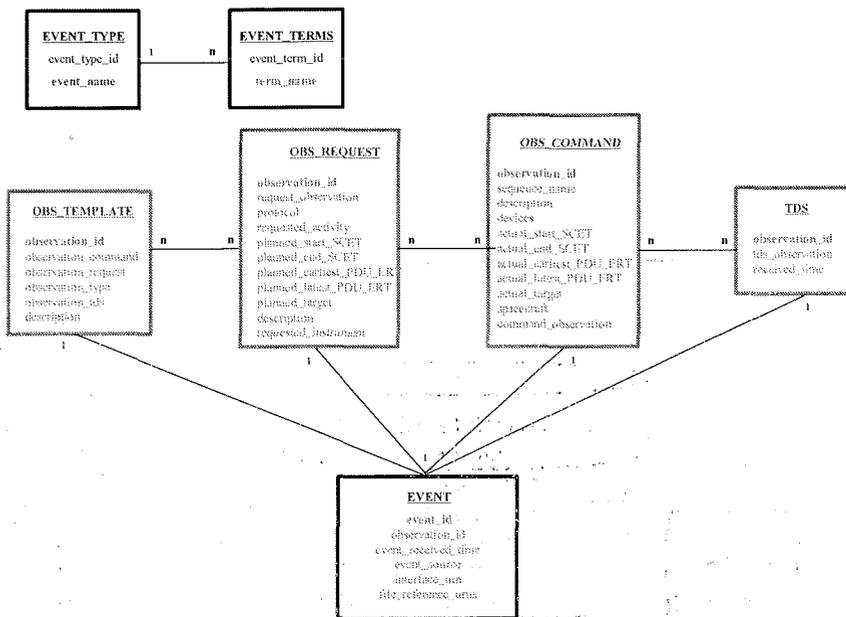


Figure 1. Accountability Service Data Flow

Our work has concentrated on building software to support the discovery of data products in the space-to-ground pipeline by providing a methodology and system for cataloguing the data products from their inception to their eventual distribution. Our system should also support the capability to aggregate several different subtypes of data products, into more complex data product *packages* which are of use in grouping together several different data product types which have been generated for a particular observation request. This paper is organized as follows. First we present our information model to support data product tracking and packaging. Then, we introduce our software component support for data product tracking. We follow by presenting our software components for data packaging. Finally we present our conclusion.

## An Information Model to Support Tracking and Packaging of Data Products

In order to support the ability to track data products along the space-to-ground pipeline, we have focused our initial efforts on creating a *data model* defining what information is relevant to track per data product in the pipeline. Below, we present our information model and discuss its salient features and the rationale behind our choice of attributes to track.



Saturday, January 03, 2004

17

Figure 2. An Information Model supporting packaging using the OODT Archive component

At the top level of our data model, we define an *Event*, which is something of interest to be tracked. Each event has an *event type* which captures a set of domain dependent terms, which we refer to as *event terms*. In this fashion, each instance of a particular event may have domain-specific terms. In our data model, we have 4 instances of events which we can currently track:

1. *Observation Templates*
2. *Observation Requests*
3. *Observation Commands*
4. *Telemetry Data System (TDS) data*

Each type of event is associated with an *observation id*, a unique ID generated for every observation request. An observation request is some request for the spacecraft to take an observation, e.g. "Direct instrument X to move Y degrees and **execute** command **take picture**".

Our data model's ability to track events is quite useful when attempting to find all events which belong to a particular observation id. In this fashion, we can issue a single request, possibly retrieving back *multiple* events, which can then be aggregated into a *package*.

## Components for Tracking Data Products

To track and package data products, we have leveraged our experience with the OODT middleware [1]. The OODT middleware defines several software components, which are instantiated into a middleware framework suitable for deployment in data-intensive environments (environments favouring the integration of heterogeneous information). For tracking data products, we have taken advantage of the OODT *Catalog and Archive (CAS)* component. The catalog and archive component serves as the basis for an *accountability service*, a service which accounts for the status of data products produced for a particular observation request, and is also able to package retrieved events at a user's request. The CAS component is attached to an underlying data source and data catalog to provide a common software interface for both the ingestion and retrieval of data. The component is also responsible for cataloging metadata, which describes data that is present in its underlying data source. In this fashion, the CAS can

search through its metadata catalog for a particular set of data that satisfies a user query rather than searching through the data itself. The CAS provides an ingest interface for data that enables both management and processing of ingested data via *Tasks*. Tasks are processing components that take as input a set of data D and produce an output data D'. Tasks are similar to filters in the pipe and filter [4] architectural style. Tasks are passive internal components, which are activated when a particular set of data is passed into the CAS component's ingest interface. The CAS component contains a set of *Rules* that define what set of data causes the execution of a particular Task component. In this fashion, data can be both validated to ensure consistency upon ingestion into a data source, as well as processed if required. The ingest interface not only ingests data into the domain data source associated with the particular CAS, but it also ingests a metadata description of the data resource which it ingests into its catalog.

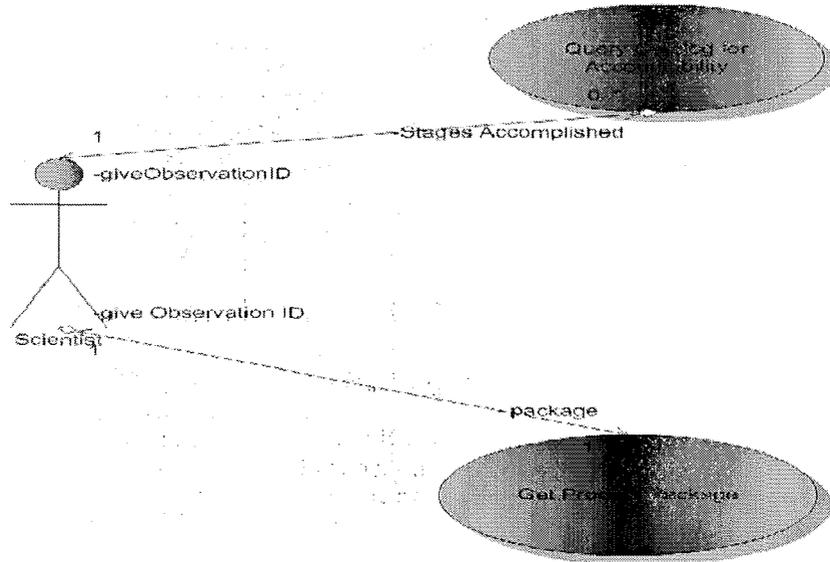


Figure 3. Use-case for our accountability service.

We have leveraged the CAS ingest interface to store event data which is produced by data systems in the space-to-ground pipeline (shown as OODT Product Server components in Figure 3 below). The CAS cataloged metadata per event is essentially instances of the data model which we have described above. In this sense, the CAS provides the ability to track, and retrieve event data in our tracking and packaging system. Figure 3 above depicts the tracking capability of our accountability service. Below, we detail our approach to the packaging problem.

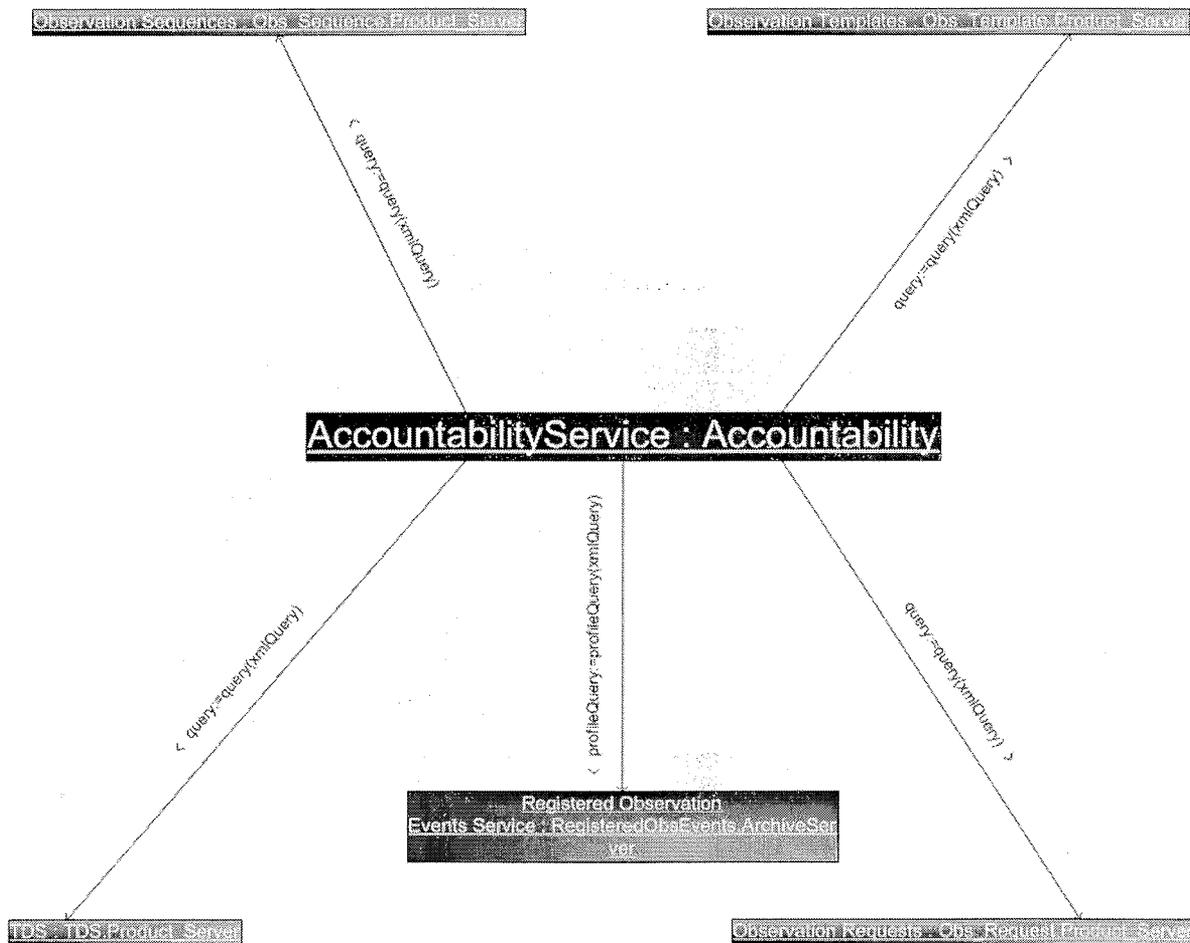


Figure 4 A UML Collaboration Diagram depicting the query catalog for tracking capability.

## Packaging Data Products using Resource Profiles

The CAS component described above supports the retrieval of event data in the form of resource *profiles* [5]. Resource profiles are an OODT data structure which uses the Dublin Core set of Data Elements [6], and the ISO-11179 specification and standardization of data elements [7] to model *resources* of interest in a data grid [2] software system. In our case, resources are available data products, which can be aggregated together and “packaged” and returned back to the user, to appear as a single data product. Instances where this could be useful are in cases where scientists need data from multiple events that were generated because of an observation request. The event data is all associated with an underlying theme, the observation request, and it can be used to determine what the status of the observation request is and what data has been generated and is available.

Since we use OODT resource profiles to define the semantics of our packages, we have deployed an OODT profile server, which stores the resource profiles of the packages which we can create. We have also deployed an OODT product server to retrieve data from the data systems that produce event data in our accountability system. Finally, we have created a DSMS packaging component which uses the Product and Profile servers to retrieve the package from the integrated data system. Since the CAS component described above supports both OODT product and profile server interfaces, we can use a

single CAS to catalog the resource profiles of packages, as well as to retrieve the package data. Figure 4 depicts the architecture of our packaging capability, and Figure 5 shows a particular sequence of interaction among the packaging components given a request to retrieve a package.

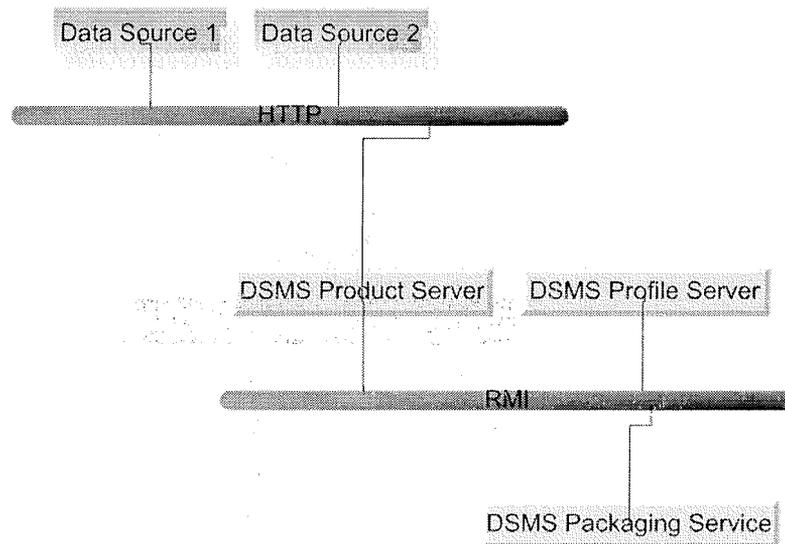


Figure 5. Packaging Capability of the Accountability Service.

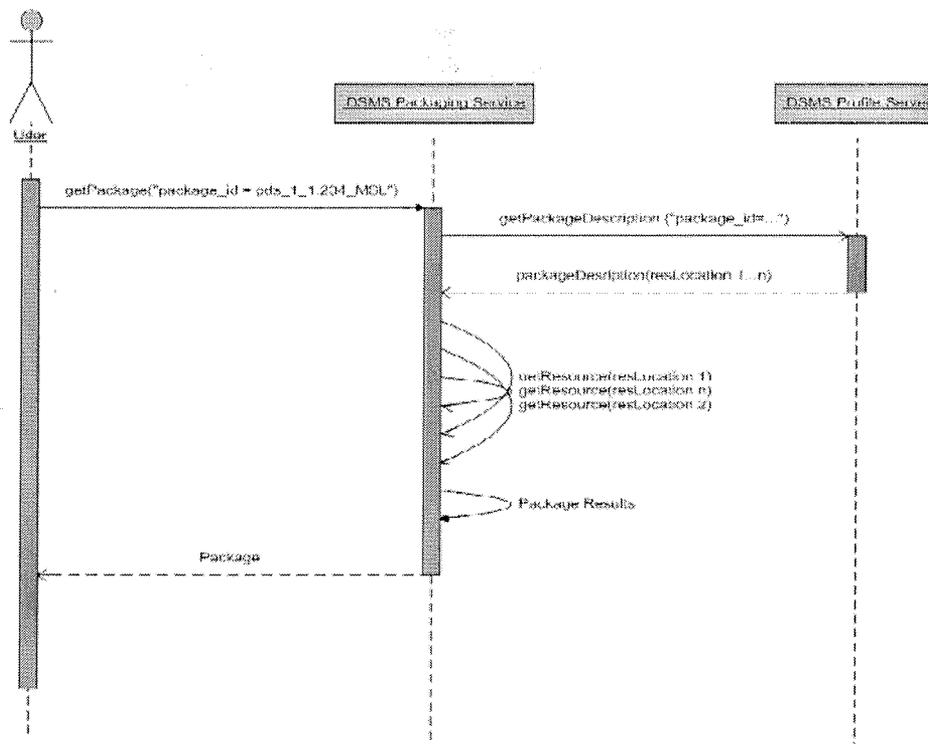


Figure 6. Packaging sequence of events.

## Conclusion

We have presented our work on supporting tracking and packaging of data products using data grid middleware for deep space mission systems. We have shown the use of the OODT catalog and archive service, product server and profile server components in the packaging and tracking tasks. Our system supports the ability to retrieve data products of interest in the space-to-ground pipeline. Our system also supports the creation and retrieval of complex packages which are aggregations of one or more data products, and associated package metadata.

## Acknowledgements

The work described in this paper was carried out at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration.

## References

- 1 D.J. Crichton, J.S. Hughes, S. Kelly and P. Ramirez. A Component Framework supporting Peer Services for Space Data Management. *In Proceedings of the IEEE Aerospace Conference*, 2002. Big Sky, Montana. [http://oodt/papers/ieee\\_bigsky\\_2002.pdf](http://oodt/papers/ieee_bigsky_2002.pdf)
- 2 A. Chervenak et al. "The Data Grid: Towards an Architecture for the Distributed Management and Analysis of Large Scientific Datasets", *Journal of Network and Computer Applications*, 2000.
- 3 L. Reich. XML Packaging for the Archiving and exchange of Binary Data and Metadata. *In Proceedings of the 2003 Open Forum on Metadata Registries*. 2003.
- 4 M. Shaw and D. Garlan. *Software Architecture: Perspectives on an Emerging Discipline*. Prentice-Hall, 1996.
- 5 D. Crichton, J.S. Hughes and S. Kelly. A Science Data System Architecture for Information Retrieval. In *Clustering and Information Retrieval*. Kluwer Academic Publishers. December 2003.
- 6 DMCI 1999. Dublin Core Metadata Element Set, Version 1.1: Reference Description. Dublin Core Metadata Initiative.
- 7 ISO/IEC 1999. Framework for the Specification and Standardization of Data Elements 11179-1, *Specification and Standardization of Data Elements 11179*. International Organization For Standardization.