

1.1.1



International Planetary Data Alliance (IPDA)

Registry Service Design Specification

Version 1.0

Editor:

S. Hardman

Authors:

S. Hardman, E. Law

Abstract

This document describes the design specification for the PDS Registry Service specifically for the IPDA.

Status of this document

This document was first generated by the Registries Implementation Project (2010-2011) with the recent release the result of the Registry Development and Coordination Project (2011-2012).

Acknowledgments

This document is based on the PDS Registry Service Software Requirements and Design Document (SRD/SDD).

Change Log

Revision	Date	Description
0.1	December 2010	Initial Draft.
1.0	July 2012	Modified to be brought up-to-date with the 1.0 version of the PDS Registry Service SRD/SDD.

Contents

Abstract.....	1
Status of this document	1
Acknowledgments.....	1
Change Log	2
Contents.....	3
1 Executive Summary	4
1.1 Purpose.....	4
1.2 Applicable Documents	4
2 Description	6
3 Use Cases	8
3.1 Manage Policy.....	9
3.2 Publish Artifact	9
3.3 Update Artifact	9
3.4 Approve Artifact.....	10
3.5 Deprecate Artifact	10
3.6 Undeprecate Artifact	10
3.7 Delete Artifact.....	10
3.8 Query Artifact	11
4 Design Philosophy, Assumptions, and Constraints	12
5 Detailed Design.....	13
5.1 Architecture	13
5.2 Interface Design.....	15
5.3 Data Model.....	16
5.4 Registry Classes	18
6 Deployment.....	19
Appendix: Acronyms.....	21

2 Executive Summary

The multi-disciplinary nature of planetary science and the increasing number of national space agencies involved in planetary exploration suggest the need for a common architecture, standards and shared services to ease discovery, access and use of planetary data by world-wide scientists regardless of which agency is collecting and distributing the data and to ensure access to and exchange of high quality planetary science data products across international boundaries.

“Registries” is an element under the International Planetary Data Alliance (IPDA) Common Architecture. It describes the shared registry services that are used by multiple IPDA members and institution. Registries are catalogs of IPDA service offerings and standard data values that are necessary to enable interoperability. For example, a registry may contain information about services offered within the IPDA (e.g., various access points for getting planetary data from an agency) or it may provide standard data values such as mission names, etc so they are used consistently across agencies.

This document addresses the use cases and software design of the Registry Service for the IPDA system.

2.1 Purpose

The purpose of this document is to initiate the development of the “Registries” element of the IPDA Architecture. This document will convey the resulting design and implementation in a manner that is understandable to the broad spectrum of IPDA stakeholders.

The IPDA, in its level 2 requirements, identified the following requirement that is the driver for the Registry Service:

3.9 IPDA will publish standards for querying planetary data system catalogs including standard query models, protocols, and templates of user interfaces

2.2 Applicable Documents

- 1) IPDA Information Level 1 and 2 Requirements, January 2008, <http://planetarydata.org/standards/ipda-requirements-20080122.pdf>
- 2) Developing a Core Set of Data Standards for the IPDA, Concept White Paper, January 2007, http://planetarydata.org/documents/white-paper-wp/ipda-wp-001_1_0_2007feb07-ipda-developing-a-core-set-of-data-standards-for-the-ipda
- 3) NASA-PDS/ESA-PSA Planetary Data Interoperability, July 2005, http://planetarydata.org/documents/white-paper-wp/IPDA-STC-WP-001_1_0-2005JUL01-NASA%20ESA%20Interoperability.pdf
- 4) IPDA System Architecture Specification, April 2009, http://planetarydata.org/standards/IPDA_SystemArch_20090518.pdf/view
- 5) CCSDS Registry Reference Model, CCSDS 314.0-W-3, Draft White Book, December 28, 2010.

6) IPDA Planetary Data Access Protocol, V1.0, November 2011.
<http://planetarydata.org/projects/inactive-projects/data-access/documents/pdap-versions/pdap-v1.0-09-11-2011>

3 Description

The Registry service provides the tracking and locating artifact function for the IPDA system (referred to as the “system” from this point forward). The intent of this service is to facilitate tracking, auditing and maintenance of artifacts within IPDA (e.g., data, dictionary definitions, services, etc.).

Within the system, the Registry service will have a limited set of external interfaces and will mostly interact with other system components. The rationale behind this is to reduce the complexity of the service as its functions are at the core of the system. Other services will build upon the information maintained in any given registry and will expose this registry-based information via external interfaces. This separation of concerns will help the system evolve as any external requirements can be leveraged on other services and thus reducing the impact to this core component.

The Registry service supports several interfaces to other services in the system. In general, these services will interact with the registry to inform the service about a new managed artifact or lookup/update basic information about an existing registered artifact. The registry will maintain three types of registrations:

Non-Digital Object Entry

This type of entry will simply capture metadata describing a non-digital object within the system. This type of entry in the existing IPDA infrastructure is akin to descriptions captured for missions, instruments, data sets, targets, people, etc.

Digital Object Entry

This type of entry tracks back to a physical set of bits. In the current IPDA infrastructure, this includes items such as science data products, documents, schemas, etc.

Relationship Entry

This type of entry will serve as a means to tie registered products together. Such support is necessary for example to correlate collections to the set of products contained within. These relationships may span registries and thus the need for coordination amongst registries exists. Example product relationships include associations with an investigation product, an instrument product and a target product.

The following is an accounting of logical registries that would be available within the system:

Inventory

As indicated above this registry instance serves as a means to capture the products within the IPDA. Registration of products will occur by crawling local repositories at the nodes. Products will remain within their local repositories and only enough information to locate and audit the product is gathered. This information will include, but not be limited to: access points, checksum, file name, and file size.

Dictionary

This registry captures and stores object, group and element definitions that make up the data dictionary. Management of these definitions occurs in the IPDA Information Model, which exports this information periodically to this logical instance of the service.

Service

This registry captures descriptions about services provided by the system. IPDA participants can share their services via this registry to help promote reuse. These descriptions could evolve over time from simple documentation in the form of a web page or document to something along the lines of a WSDL or WADL formatted description. The service registry will not dictate interaction with a given service but rather exist as a means to document and promote existing services.

The service defined in this document will provide the IPDA system with a single implementation of registry capabilities for use by the other services and applications within the system. This service is tailor-able depending on the type of registry and types of artifacts to be registered with a given instance.

4 Use Cases

A use case represents a capability of the service element and how the user (actor) interacts with the system. It should be at a high enough level so as not to reveal or imply the internal structure of the system. An actor is an object (e.g., person, application, etc.) outside the scope of the component but interacts with the component. This section captures the use cases for the Registry service based on the description of the service from the previous section as well as use cases defined in the CCSDS Registry and Repository Reference Model [5]. These use cases will be used in the derivation of requirements for the service. The following diagram details the use cases:

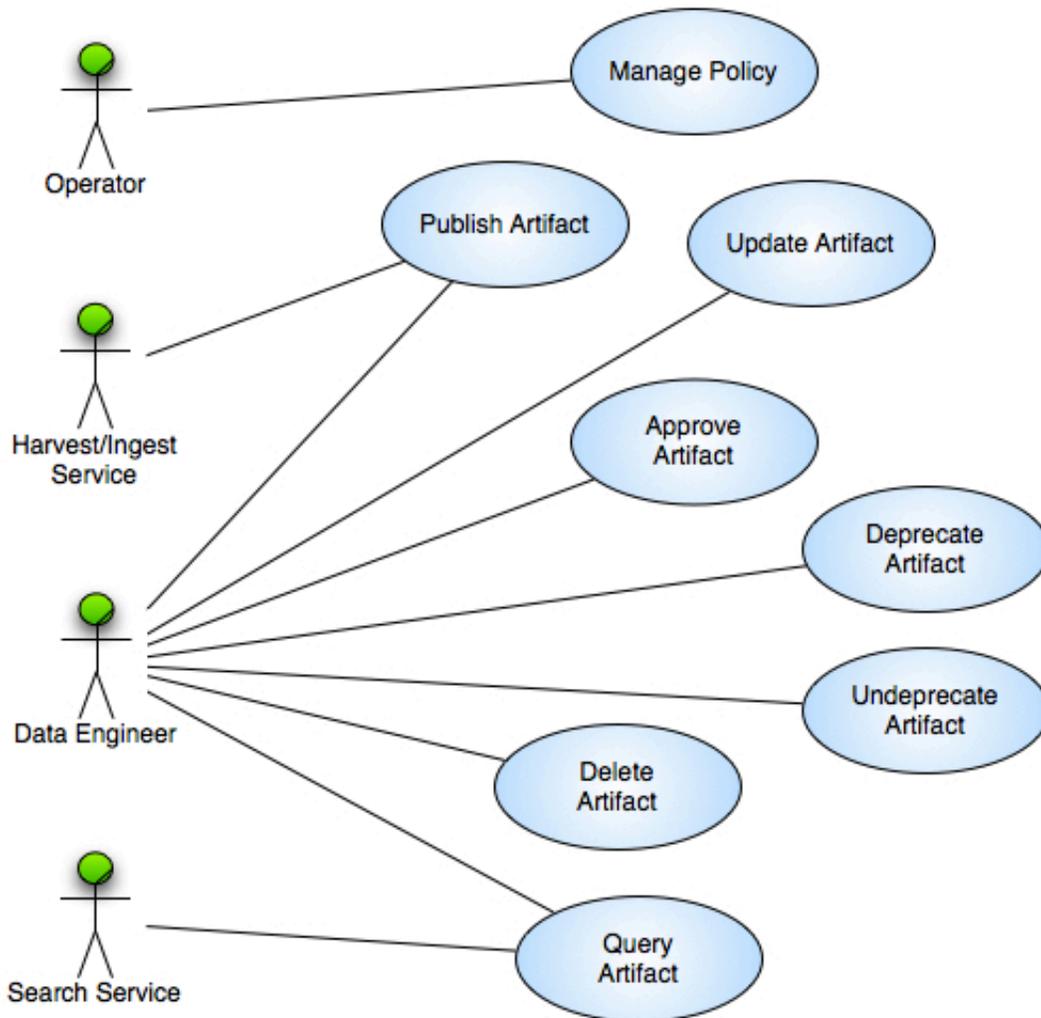


Figure 1: Registry Use Cases

The above diagram identifies the following actors (represented as stick figures):

Data Engineer

This actor represents an engineer that curates the data before and after it enters the IPDA system.

Harvest/Ingest Service

This actor represents the software within the system that will perform automated registration of artifacts.

Operator

This actor represents an engineer that is responsible for configuring and monitoring the system.

Search Service

This actor represents the software within the system that will query for registered artifacts.

The following sections detail the use cases identified in the above diagram.

4.1 Manage Policy

The Registry service is policy driven with regard to the types of artifacts that it registers, the associated metadata it expects to receive for an artifact and the allowed operations on a type of artifact. This use case pertains to the Operator actor.

1. Operator authenticates for access to the Registry service interface.
2. Operator submits an update to the Registry service policy to add, modify or delete a type of artifact via the Registry service interface.
3. Registry service accepts (verifies input against constraints) and commits (updates the underlying metadata store) the operation.

4.2 Publish Artifact

Register artifacts with the system for the purpose of tracking, discovery and retrieval. This use case pertains to the Ingest and Harvest services that will perform automated registration of artifacts. It also pertains to the Data Engineer who will perform ad hoc registrations of artifacts within the system.

1. Ingest/Harvest service authenticates for access to the Registry service API.
2. Ingest/Harvest service submits an artifact for registration via the Registry service API.
3. Registry service validates the metadata submitted for the artifact.
4. Registry service assigns a version to the artifact based on the IPDA identifier.
5. Registry service records the metadata associated with the artifact in the underlying metadata store.

Alternative: Ad Hoc Registration

At step 1, the Data Engineer initiates the artifact registration.

- a. Data Engineer authenticates for access to the Registry service interface.
- b. Data Engineer submits an artifact for registration via the Registry service interface.
- c. Return to primary scenario at step 3.

4.3 Update Artifact

Update a registered artifact and its associated metadata. This use case pertains to the Data Engineer who will perform artifact registration updates within the system.

1. Data Engineer authenticates for access to the Registry service interface.
2. Data Engineer submits an updated artifact for registration via the Registry service interface.
3. Registry service validates the metadata submitted for the artifact.
4. Registry service records the metadata associated with the artifact in the underlying metadata store.

4.4 Approve Artifact

Approve registered artifacts in order to make them visible to the public. This use case pertains to the Data Engineer who will approve registered artifacts.

1. Data Engineer authenticates for access to the Registry service interface.
2. Data Engineer marks a registered artifact as approved via the Registry service interface.
3. Registry service records the approval in the underlying metadata store.

4.5 Deprecate Artifact

Deprecate registered artifacts when no longer pertinent. This could be due to the availability of a newer version of the artifact. This use case pertains to the Data Engineer who will deprecate registered artifacts.

1. Data Engineer authenticates for access to the Registry service interface.
2. Data Engineer marks a registered artifact as deprecated via the Registry service interface.
3. Registry service records the deprecation in the underlying metadata store.

4.6 Undeprecate Artifact

Undeprecate registered artifacts when their pertinence has been restored. This use case pertains to the Data Engineer who will undeprecate registered artifacts.

4. Data Engineer authenticates for access to the Registry service interface.
5. Data Engineer marks a registered artifact as undeprecated via the Registry service interface.
6. Registry service records the undeprecation in the underlying metadata store.

4.7 Delete Artifact

Delete registered artifacts from the registry. This will normally be utilized during testing but could be utilized during operations if a registration was made by mistake. Privilege for this capability should be limited. This use case pertains to the Data Engineer actor who will delete registered artifacts.

1. Data Engineer authenticates for access to the Registry service interface.
2. Data Engineer marks a registered artifact as deleted via the Registry service interface.
3. Registry service deletes the metadata associated with the artifact in the underlying metadata store.

Alternative: Operation Not Allowed

At step 3, the Registry service does not allow the operation per policy.

- a. Registry service checks policy for allowed operations.

- b. Registry service does not allow deletion of the artifact per policy.

4.8 Query Artifact

Discover registered artifacts from the registry by submitting queries against the registered metadata attributes. This use case pertains to the Data Engineer and Search service actors.

1. Search service submits a query for artifact(s) via the Registry service API.
2. Registry service accepts the query and returns metadata for one or more artifacts from the underlying metadata store matching the criteria.

5 Design Philosophy, Assumptions, and Constraints

The intent of the Registry service is to provide a generic and simple solution for registering artifacts within the system. The design of this service heavily leverages current work efforts by CCSDS in the form of the Registry Reference Model [5]. This reference model in turn, heavily leverages the ebXML suite of standards managed by OASIS.

6 Detailed Design

The design covers the component breakdown within the service, external/internal interfaces and the associated data model.

6.1 Architecture

The following diagram details the component breakdown for the Registry service:

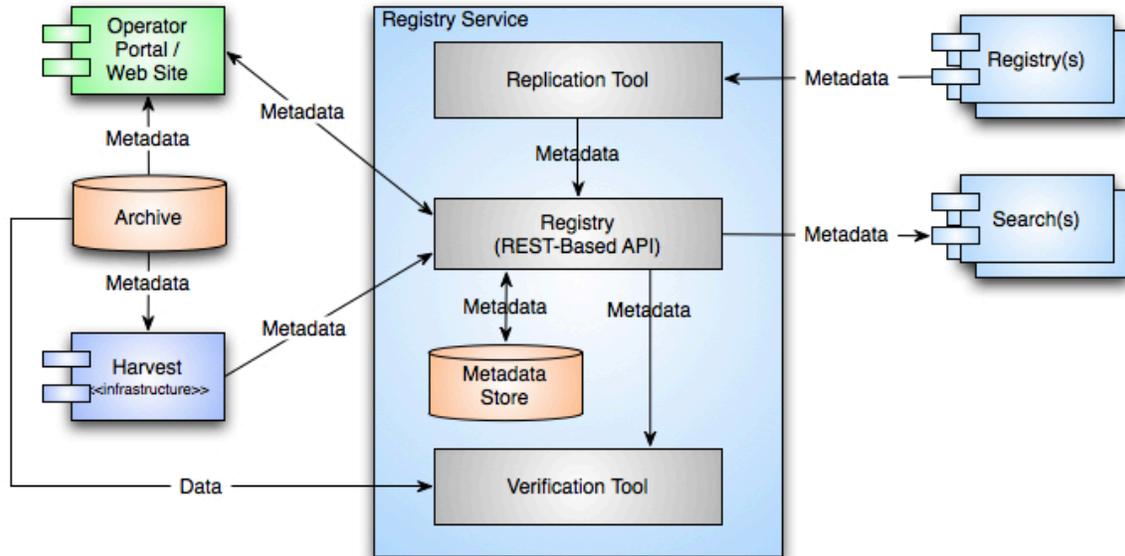


Figure 2: Registry Service Architecture

There are two scenarios for populating a registry:

Ad hoc Access via Portal

Although this is somewhat of a misnomer because the portal will use the REST-based API to access the service, this is where the Data Engineers can perform ad hoc registrations as well as the perform functions like approve and deprecate which are probably not suitable for automated access. Ad hoc access also includes performing functions like query for the purposes of managing the registry.

Automated Access via API

This scenario represents access from services like Harvest, where registrations are automated and achieved through service-to-service communication via the REST-based API, as well as the Planetary Data Access Protocol (PDAP). The PDAP interface is defined in the IPDA Planetary Data Access Protocol document [6].

The diagram above assumes that the registered artifact resides in a managed repository (i.e., archive directory structure) and will be registered in place. The following diagram supports the scenario where the Storage service is utilized to manage the physical bits of the registered artifact:

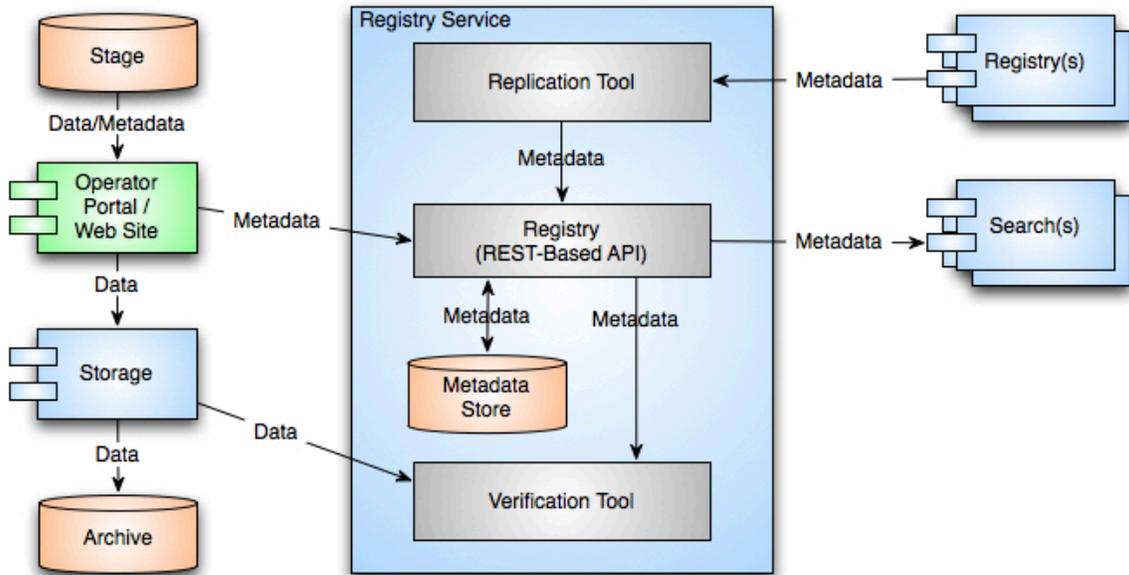


Figure 3: Registry Service Architecture (with Storage)

This scenario mainly pertains to the management of schemas and other documents within the system that will not reside in an IPDA archive directory structure. In this case the Operator Portal submits the files to the Storage service and then registers those files as an artifact with the Registry service.

In addition to population of the registry, there are two scenarios for importing/exporting metadata from the registry:

Metadata Import for Replication

There are two purposes for replication. The first is to populate an aggregate registry utilized for satisfying tracking, metrics reporting and catalog-level search requirements. The second is for sharing artifact registrations between Nodes. The Replication Tool pulls artifact registrations from other Registry service instances according to its local configuration.

Metadata Export for Search

This is where the Registry service facilitates end-user search. Instances of the Search service will query one or more instances of the Registry service in order to generate search indices. These indices are tailor-able for the search application that will utilize them.

In addition to registry population and metadata export, the service will also provide the capability to perform verification for registered artifacts. This capability is intended to be executed local to the registry or more specifically, local to the repository associated with the registry. A capability like this could utilize a lot of bandwidth if executed remotely.

6.2 Interface Design

The following diagram focuses on the interfaces, both external and internal for the Registry service:

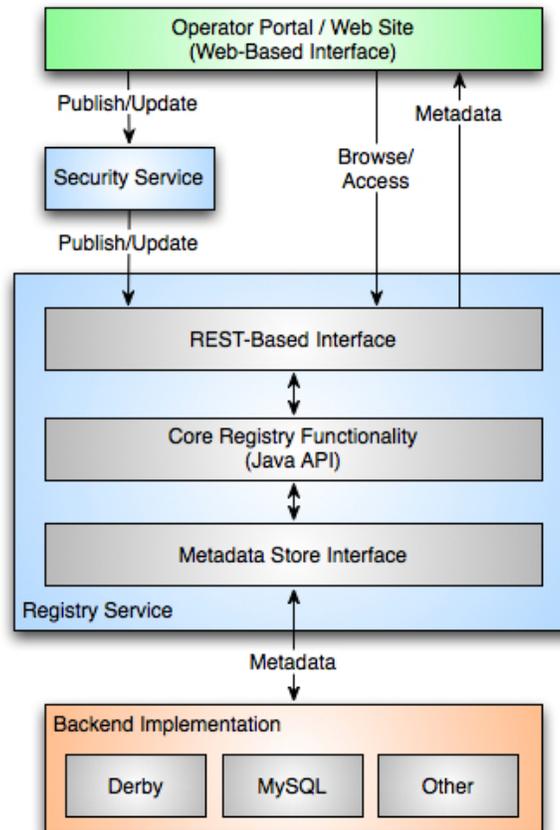


Figure 4: Registry Service Interfaces

The interfaces are described in more detail in the following sections.

6.2.1 External Interface Design

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

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

The goals for the interface are as follows:

- Keep the service simple and refrain from adding too much functionality
- Allow messaging in the form of XML or JavaScript Object Notation (JSON)

- Allow for extensibility as new artifact types are defined

In addition, each interface should adhere to the following:

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

Any interface that modifies the contents of the registry will incorporate security. This means that any interface specified below as an HTTP POST will first require interaction with the Security service. Integration with the Security service is accomplished through the Application Server and does not require any specific coding within the Registry service. The only change to these interfaces will be in terms of a required HTTP header or cookie being set that will provide the means to verify the validity of the request. These requests will require secure HTTP (HTTPS).

The following are some examples detailing the functionality of the REST-based interface using HTTP methods. This interface delegates all functions involving a product:

- <http://planetarydata.org/registry/extrinsics/>
 - GET: Retrieves a paged list of products from the registry.
 - POST: Publishes a product to the registry.

This interface acts on a specific product (lid stands for logical identifier):

- [http:// planetarydata.org/registry/extrinsics/logicals/{lid}/](http://planetarydata.org/registry/extrinsics/logicals/{lid}/)
 - GET: Retrieves the product from the registry.
 - POST: Updates the product in the registry.
 - DELETE: Removes the product from the registry.

6.2.2 Internal Interface Design

The primary internal interface for the Registry service involves communication with the underlying metadata store. This interface will follow a generic design with the intent of supporting multiple backend implementations for the metadata store. The layered design for the backend implementation allows for technology refresh and multiple deployment scenarios. The metadata store interface will support the data model detailed in the following section of this document.

6.3 Data Model

The following diagram represents the CCSDS Registry logical model (key classes) and is the basis for implementing the underlying metadata store for this service:

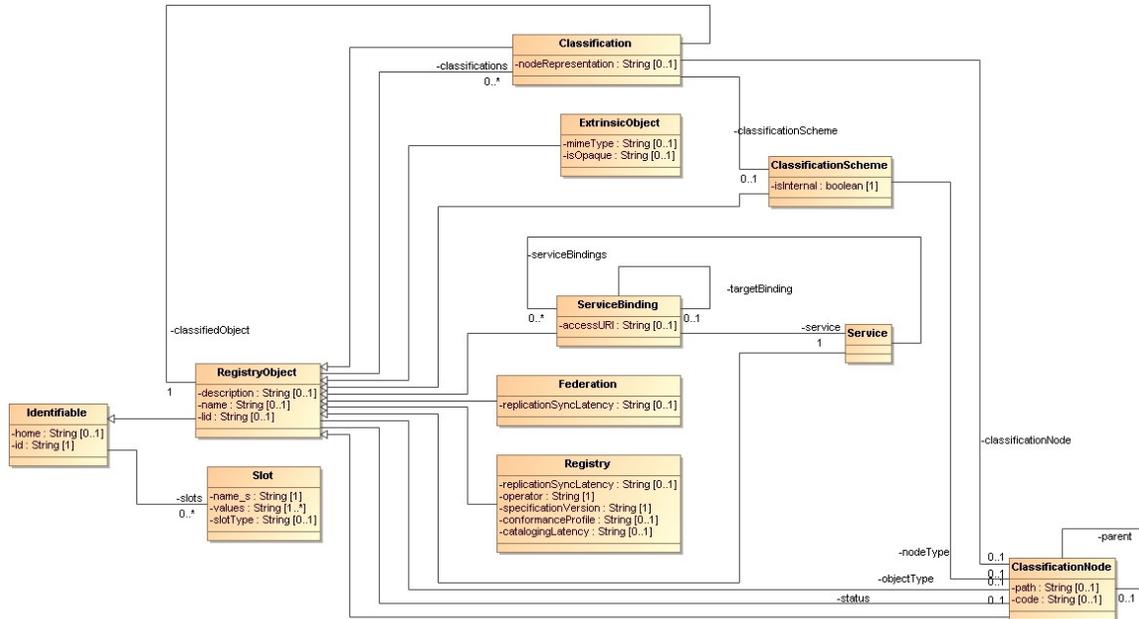


Figure 5: Registry Service Data Model

The classes detailed in the diagram above and a couple of others that are important to the design of the Registry service are defined below:

Association (not pictured)

Specifies a relationship between two RegistryObject instances. An IPDA example of an association is that a data product is a member of a collection.

AuditableEvent (not pictured)

Instances of AuditableEvent record the actions taken against a RegistryObject instance. For example, approval or deprecation of a RegistryObject is an auditable event.

Classification

Specifies the classification of a RegistryObject utilizing the ClassificationScheme and ClassificationAgency classes. Classifications utilized by the Registry service are defined in the IPDA data model.

ExtrinsicObject

This is the place holder object for products in the data model. All IPDA products (e.g., data products, investigations, instruments, personnel, etc.) will be derived from the ExtrinsicObject class. The IPDA products are defined in the IPDA data model.

Federation

An instance of the Registry service may belong to a federation of registries. There is likely to be one federation defined for the IPDA Registry service instances.

Identifiable

This class provides the ability to identify objects by an id attribute and is the parent class for all of the classes defined here.

Registry

Represents an instance of a Registry service within the IPDA.

RegistryObject

The RegistryObject class extends the Identifiable class and serves as a common super class for most classes in the data model. The term “artifact”, used throughout this document, is equivalent to an instance of the RegistryObject class.

Service

This class captures descriptions of services utilizing the ServiceBinding class.

Slot

The Slot class provides a dynamic way to add arbitrary attributes to RegistryObject instances. For example, this is where the IPDA will capture the 10 plus or minus 2 keywords to be utilized in global search scenarios.

6.4 Registry Classes

There are four key types of registry services that are applicable to IPDA:

Centralized Service Registry

A centralized service registry is responsible for managing all IPDA service registrations. It provides the ability to describe and publish, and allow discovery the offered functionality of the IPDA services (e.g., a web site) developed and maintained by all the IPDA Agencies.

Centralized Data Dictionary Registry

A centralized data dictionary registry is responsible for managing IPDA data dictionary registrations. It provides the ability to describe and publish, and allow discovery the formal definitions of the components and the organization of IPDA data defined in the IPDA Information Model.

Centralized (Replicated) Inventory Registry

A centralized service registry is responsible for replicating product metadata managed at various IPDA agencies. It provides the ability to describe and publish, and allow discovery all data products managed and archived locally at the IPDA agencies.

Local Product Registry

A product registry is responsible for managing IPDA data products. It provides the ability to describe and publish, and allow discovery the data products managed and archived at a local IPDA agency.

7 Deployment

The following diagram depicts a deployment scenario for service instances:

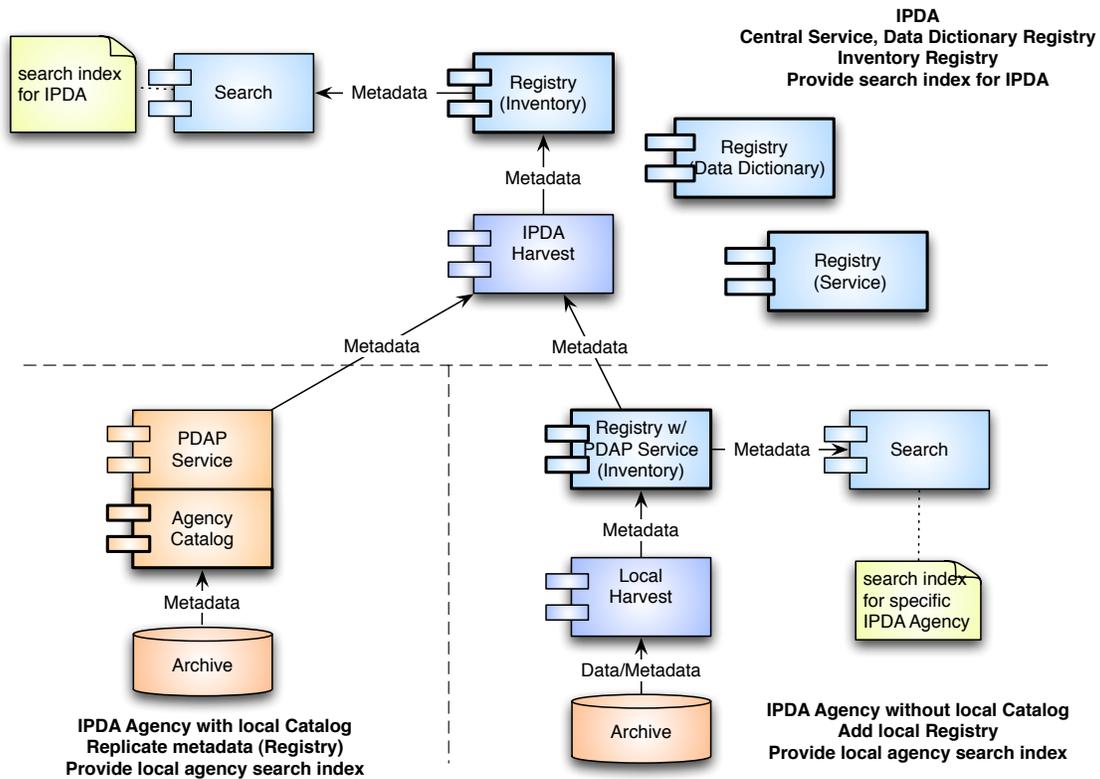


Figure 6: Registry Service Deployment

The registry deployment instances depicted in the diagram include:

IPDA Agency Local Registry

For the IPDA agencies that have local repositories without existing data management infrastructure (Registry/Catalog), a local instance of the Registry Service with PDAP service can be added for local inventory management. A local instance of the Harvest service can be configured for the local repository populates this registry. A local Search service will be able to extract metadata from this registry to create local index to support local search.

Centralized (Replicated) Inventory Registry

An IPDA centralized inventory registry for hosting replicated registry entries from all IPDA Agency Local Registry service instances. This registry will allow for catalog-level search without the need to perform live queries across the distributed registry instances. Replications to the registry and index generation are performed during off-peak hours further increasing productivity of the system. The Harvest service will be configured to extract registry entries from the existing infrastructures using PDAP services to populate the Centralized IPDA Replicated Inventory Registry.

Centralized Service Registry

An IPDA centralized registry for managing IPDA service registrations can be supported. The Operators will use the Operator Portal to populate this registry.

Centralized Data Dictionary Registry

An IPDA centralized registry for managing IPDA data dictionary registrations can be supported. The Operators can use the Operator Portal to populate this registry. Population can also occur from the IPDA Information Model, which exports the definitions into the registry.

Appendix: Acronyms

The following acronyms pertain to this document:

API – Application Programming Interface
CCSDS – Consultative Committee for Space Data Systems
ebXML – Electronic Business using XML
ESA – European Space Agency
HTTP – HyperText Transfer Protocol
IPDA – International Planetary Data Alliance
OASIS – Organization for the Advancement of Structured Information Standards
PDAP – Planetary Data Access Protocol
PDS – Planetary Data System
PSA – Planetary Science Archive
REST – Representational State Transfer
WADL – Web Application Description Language
WSDL – Web Service Definition Language
XML – Extensible Markup Language