Planetary Data System

Harvest Tool

Software Requirements and Design Document (SRD/SDD)

Sean Hardman

July 20, 2011
Version 1.1

Jet Propulsion Laboratory
Pasadena, California

CHANGE LOG

Revision	Date	Description	Author
0.1	2010-02-22	Initial draft.	S. Hardman
0.2	2010-02-28	Incorporated comments from the System Design Working Group and added deployment scenarios.	S. Hardman
0.3	2010-03-06	Renumbered the level 4 requirement and moved deployment content to the Implementation section.	S. Hardman
0.4	2010-09-28	Updated the controlling and applicable document references, made some minor modifications for consistency with the other design documents and added an activity diagram in section 8.	S. Hardman
0.5	2011-03-11	Updated the Detailed Design section with a new activity diagram and a description for how to handle references.	S. Hardman
1.0	2011-06-12	Updated the controlled document references and Reference Handling section based on comments from the SDWG and DDWG.	S. Hardman
1.1	2011-07-20	Updated the activity diagram to include file product registration.	S. Hardman

TABLE OF CONTENTS

1.0 INTRODUCTION

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

This tool provides functionality for capturing and registering product metadata. The tool will run locally at the Discipline Node to crawl the local data repository in order to discover products and register associated metadata with the Registry (Inventory) service.

1.1 Document Scope and Purpose

This document addresses the use cases, requirements and software design of the Harvest tool within the PDS 2010 data system. This document is intended for the reviewer of the tool as well as the developer and tester of the tool.

1.2 Method

This combined Software Requirements and Software Design Document (SRD/SDD) represents the software by defining use cases and requirements and by using architecture diagrams, functional descriptions, context diagrams and data flow diagrams for the high-level design. UML diagrams will illustrate the detailed design.

1.3 Notation

The numbering of the requirements in this document will be formatted as **LX.(REG|HVT).AA.X**, where:

- **LX** represents the requirements level where X is a number.
- **REG** is an abbreviation representing the registry requirement section for the specified level.
- **HVT** is an abbreviation representing the harvest requirement section for the specified level.
- **AA** is a two-letter abbreviation representing the requirement sub-category (optional).
- **X** is a unique number within the section and optional sub-category for the requirement.

Following the text of a requirement may be a reference to the requirement or use case from which it was derived. The reference will be in parenthesis. A paragraph following a requirement, which is indented and has a reduced font size, represents a comment providing additional insight for the requirement that it follows. This comment is not part of the requirement for development or testing purposes.

1.4 Controlling Documents

[1] Planetary Data System (PDS) Level 1, 2 and 3 Requirements, March 26, 2010.

[2] Planetary Data System (PDS) 2010 Project Plan, February 2010.

[3] Planetary Data System (PDS) 2010 System Architecture Specification, Version 1.2, May 25, 2011.

[4] Planetary Data System (PDS) 2010 Operations Concept, February 2010.

[5] Planetary Data System (PDS) General System Software Requirements Document (SRD), Version 1.0, June 11, 2011.

1.5 Applicable Documents

[6] Planetary Data System (PDS) Registry Service Software Requirements and Design Document (SRD/SDD), Version 1.0, June 12, 2011.

[7] PDS4 Information Model Specification, PDS4 Information Model Specification Team.

1.6 Document Maintenance

The component design will evolve over time and this document should reflect that evolution. This document is limited to design content because the specification content will be captured in separate documentation (e.g., Installation Guide, Operation Guide, etc.). This document is under configuration control.

2.0 COMPONENT DESCRIPTION

This tool provides functionality for capturing and registering product metadata for the PDS 2010 system (referred to as the “system” from this point forward). The tool will run locally at the Discipline Node to crawl the local data repository in order to discover products and register associated metadata with the Inventory service. The following diagram details the context of the Harvest tool within the system:

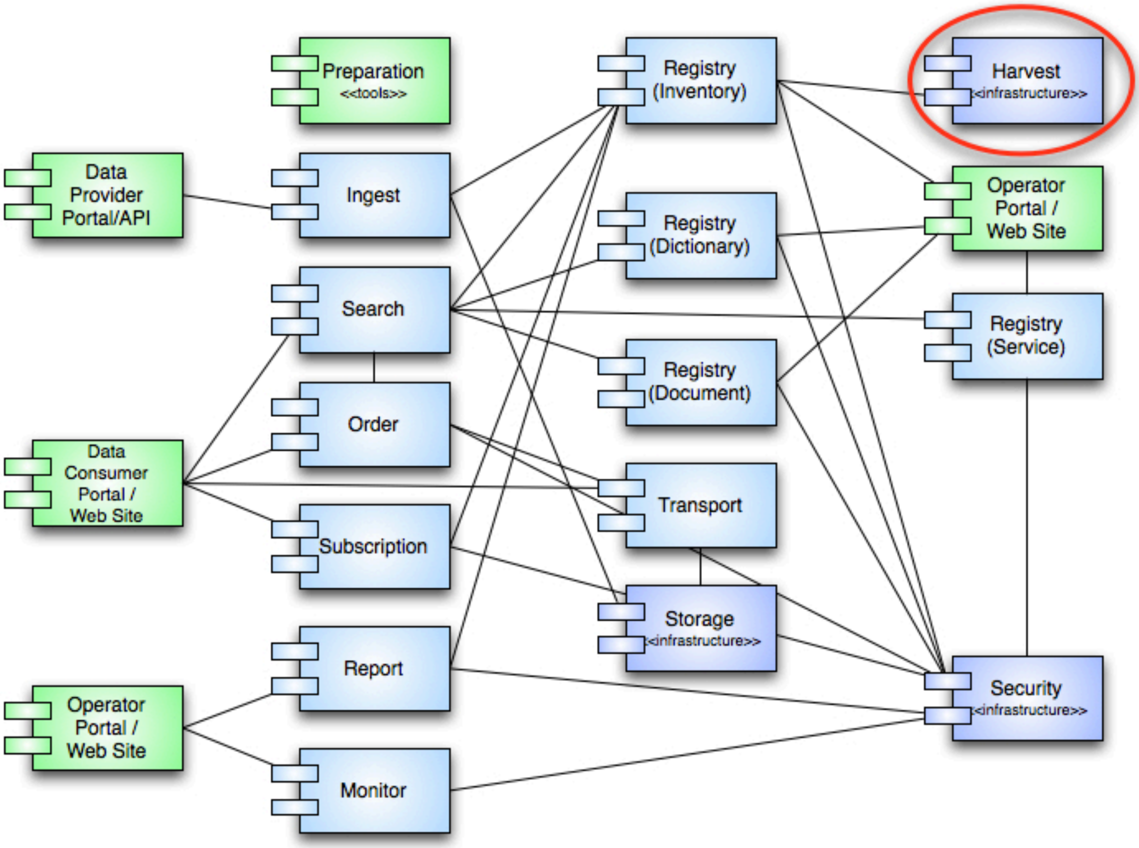

Figure 1: Harvest Tool Context

Within the system, the Harvest tool is an infrastructure component. This means that there will not be any external interfaces to the tool. As depicted in the diagram above, the Harvest tool supports a single interface with the Inventory (Registry) service. This interface has a single purpose and that is to register products from a Discipline Node’s data repository to its associated Inventory service instance. The Inventory service is an instance of the Registry service that offers an Application Programming Interface (API) for interacting with that service. The details regarding the tool interface can be found in section 6.2.

The Harvest tool provides the first line of metadata harvesting within the system in order to facilitate tracking of and access to products. The second line of harvesting occurs in the Search service where metadata from the Registry instances merges with metadata from other sources to generate search indices for targeted search applications.

Although the current PDS system does not have a common Harvest tool, several of the Discipline Nodes have implemented Node-specific capabilities for their own local systems. The tool defined in this document will provide the PDS 2010 system with a common implementation of harvest and registration capabilities for use within the system.

3.0 USE CASES

A use case represents a capability of the component and why the user (actor) interacts with the component. 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 Harvest tool based on the description of the component from the previous section. These use cases will be used in the derivation of requirements for the component. The following diagram details the use cases:

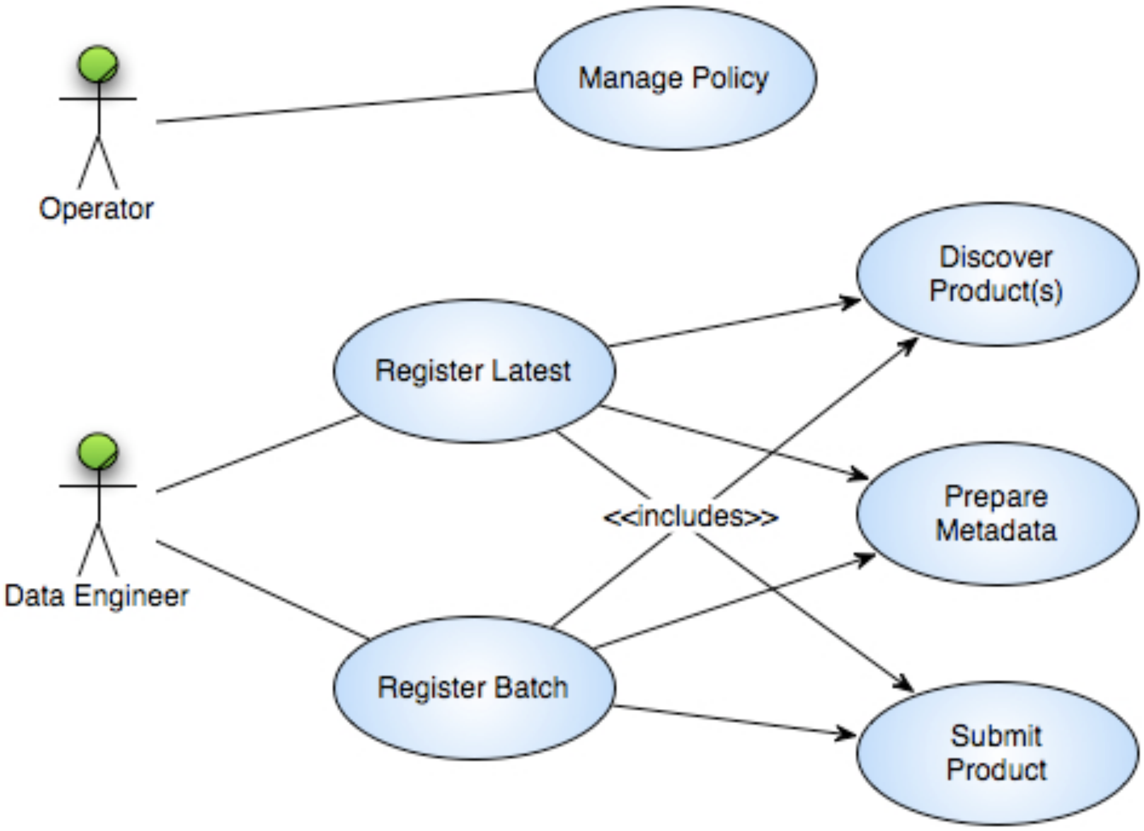

Figure 2: Harvest Tool Use Cases

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

Data Engineer
This actor represents a portion of the PDS Technical group that curates the data before and after it enters the PDS system.

Operator
This actor represents a portion of the PDS Technical group that is responsible for configuring and monitoring the system.

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

3.1 Manage Policy

The tool is policy driven with regard to how it discovers products and prepares metadata for those products. This use case pertains to the Operator actor.

1. Operator accesses a configuration file for an instance of the Harvest tool.
2. Operator edits the configuration file specifying policy for identifying target products and mapping the product type to a corresponding type supported by the target Registry service instance.
3. Operator saves the configuration file.

3.2 Register Latest

The tool runs in a mode where it wakes up periodically, inspects a target directory or directories and registers the latest products discovered. This use case pertains to the Data Engineer actor.

1. Data Engineer executes the Harvest tool specifying the configuration file.
2. Harvest tool searches for products in the target directory or directories (include Discover Product(s) use case).
3. Harvest tool prepares metadata for each discovered product (include Prepare Metadata use case).
4. Harvest tool registers each product with the target Registry service instance (include Submit Product use case).
5. Harvest tool sleeps the specified amount of time and then the scenario returns to step 2.

3.3 Register Batch

The tool runs in a mode where it performs one pass against a target directory and registers the products discovered. This mode of execution is appropriate for registration of an entire collection of products. It gives the Data Engineer more control over the registration of products by allowing a batch to be registered and then reviewed in the registry before approval. This use case pertains to the Data Engineer actor.

1. Data Engineer executes the Harvest tool specifying the configuration file.
2. Harvest tool searches for products in the target directory or directories (include Discover Product(s) use case).
3. Harvest tool prepares metadata for each discovered product (include Prepare Metadata use case).
4. Harvest tool registers each product with the target Registry service instance (include Submit Product use case).

3.4 Discover Product(s)

Products are discovered based on the directories where they reside and the names of the files that make up the product. There are two methods for product discovery. The first utilizes the manifest file that will accompany collections in PDS4. The second is with a regular expression. The configuration will specify both methods for the tool. This use case is included as part of the Register Latest and Register Batch use cases.

1. Harvest tool obtains criteria for product discovery from the configuration file.
2. Harvest tool traverses the target directory or directories searching for products that match the criteria.
3. Harvest tool discovers candidate product(s).

<u>Alternative: Previously Discovered Product</u>
At step 3, the tool has already registered the discovered product.

a. Harvest tool determines a previous registration for a candidate product and skips it.
b. Return to primary scenario at step 2.

3.5 Prepare Metadata

Metadata is prepared for a discovered product based on the product type specified in the configuration file for the service. This use case is included as part of the Register Latest and Register Batch use cases.

1. Harvest tool determines the metadata for a product based on the product type.
2. Harvest formats the metadata for submission to the Registry service.

3.6 Submit Product

A product and its associated metadata are submitted to the target instance of the Registry service. This use case is included as part of the Register Latest and Register Batch use cases.

1. Harvest tool authenticates for access to the Registry service API (include Security service Authenticate User use case).
2. Harvest tool submits the associated metadata for a product for registration via the Registry service API.
3. Registry service responds with a successful status regarding the registration and the global unique identifier for the product.
4. Harvest tool logs the registration.

Alternative: Product Registration Fails

At step 2, the product registration fails for any number of reasons.

a. Registry service returns an exception with cause of failure.
b. Harvest tool logs the exception.

4.0 REQUIREMENTS

The architecture definition phase of the PDS 2010 project resulted in the decomposition of the system into several elements [3]. The Harvest tool derives from the Catalog/Data Management element, which was derived from requirements 2.6 and 2.2.2 of the PDS Level 1, 2, and 3 Requirements document [1]. The following level 3 requirements are relevant to this tool:

> **2.2.2** PDS will track the status of data deliveries from data providers through the PDS to the deep archive
> **2.6.2** PDS will design and implement a catalog system for managing information about the holdings of the PDS

In addition to the level 4 and 5 requirements specified below, the Harvest tool must also comply with the general tool-based requirements found in the General System SRD document [5].

4.1 Level 4 Requirements

The level four requirements in PDS represent subsystem or component requirements at a high level. The following requirements pertain to the Harvest tool:

L4.REG.3 - The system shall register artifacts of a data delivery into an instance of the registry. (2.2.2, 2.6.2)

> A data delivery consists of products (artifacts) including but not limited to data, document and software.

Requirements L4.REG.1, L4.REG.2 and L4.REG.4 are defined in the Registry Service SRD/SDD document [6].

4.2 Level 5 Requirements

The level five requirements in PDS represent subsystem or component requirements at a detailed level. The following requirements pertain to the Harvest tool:

L5.HVT.1 - The tool shall accept a configuration file specifying policy for tool behavior. (L4.REG.3, UC 3.1)

L5.HVT.2 - The tool shall provide a command-line interface for execution. (L4.REG.3, UC 3.2, UC 3.3)

L5.HVT.3 - The tool shall execute from a scheduler. (L4.REG.3, UC 3.2)

L5.HVT.4 - The tool shall recursively traverse the specified directory or directories in order to identify candidate products for registration. (L4.REG.3, UC 3.4)

L5.HVT.5 - The tool shall determine candidate products for registration through a combination of the following: (L4.REG.3, UC 3.4)

a. The product is listed in a manifest file residing in the target directory.
b. The product matches the specified file naming convention from the configuration file.
c. A previously registered product has been modified.

L5.HVT.6 - The tool shall capture metadata for a candidate product specified by the product type. (L4.REG.3, UC 3.5)

> This includes identifying metadata (i.e., file location in the data repository) as well as product-specific metadata.

L5.HVT.7 - The tool shall submit the associated metadata for a candidate product to the specified Registry Service instance. (L4.REG.3, UC 3.6)

> Products are registered in place (i.e., the local data repository) and are not physically transferred to the Registry Service.

L5.HVT.8 - The tool shall track each product registration. (L4.REG.3, UC 3.6)

> This is internal tracking for determining previous product registration.

5.0 DESIGN PHILOSOPHY, ASSUMPTIONS, AND CONSTRAINTS

The intent of the Harvest tool is to provide a simple solution for crawling product archives at the Discipline and Data Nodes for the purpose of registering products into the federated system of registries. The tool will be configurable to support varying directory structures across the Nodes including PDS3 directory structures.

There are two reasons for choosing the crawler design. The first is that it provides backward compatibility and support for existing archives by simply modifying the configuration for the target directory structure and the types of products to be registered. The second is that it will cause minimal impact on current Node operations. The crawler can be run against the current archive and not interfere with existing software running at the Node.

6.0 ARCHITECTURAL DESIGN

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

6.1 Component Architecture

The following diagram details the architecture for the Harvest tool:

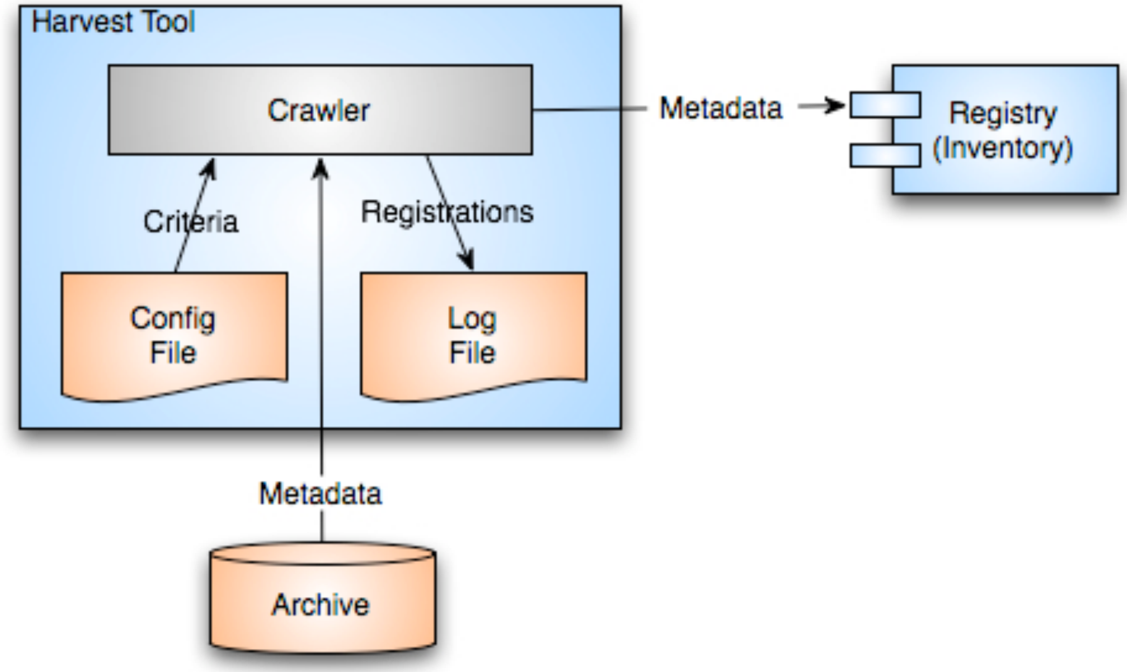

Figure 3: Harvest Tool Architecture

The Harvest tool consists simply of a Crawler component that receives its configuration from a local configuration file as well as from command-line parameters. It extracts metadata from candidate products and registers those products with the target Registry service instance using the REST-based Registry service API. The Registry Service SRD/SDD [6] documents the API in detail. A local log file captures each registration.

6.2 External Interface Design

The external interface for the Harvest tool is limited to the command-line interface and the configuration file. The tool utilizes Apache's CLI (Command-Line Interface) library for accepting options on the command-line. The command-line interface accepts the following options:

- File specification for the configuration file
- File specification for the log file
- User name and password for registering with a secured Registry service

The configuration file utilizes an XML structure for specifying additional information pertaining to a specific harvest execution. The following information is typical:

- Target directory to crawl and a regular expression for identifying candidate product labels (e.g., *.xml)
- Each product type to be registered from among the candidate product labels
- The additional metadata elements to be registered for each product type

If the target for a specific harvest execution is a PDS3 data set, the tool also requires logical identifiers for the data set, mission/investigation, instrument host, instrument and the associated targets.

6.3 Internal Interface Design

The Harvest tool does not have any internal interfaces of consequence.

6.4 Data Model

The Harvest tool does not have an associated data model but the metadata that passes to the Registry service for product registration is subject to the PDS4 Information Model Specification [7] based on the type of product registered.

7.0 ANALYSIS

The choice of a crawler-based approach was influenced by involvement of Engineering Node staff in recent data system development at the Jet Propulsion Laboratory. These include data systems developed for:

- Physical Oceanography Distributed Active Archive Center (PO.DAAC)
- Orbiting Carbon Observatory (OCO)

Both data systems rely on crawler-based approaches for identifying and registering products with their respective registries. Although both projects developed their solutions with reuse in mind, they each took a different approach with regard to extension points and the underlying library utilized to access the file system. The development staff reviewed both approaches for suitability in the Harvest tool and determined that the OCO approach, which utilized OODT's CAS Crawler component, was appropriate for PDS.

8.0 IMPLEMENTATION

The PDS 2010 system is a phased implementation with increasing capabilities delivered in three planned builds. The builds are as follows:

- **Build 1** – This build consists of the Ingestion subsystem including the Security, Harvest, Registry (Inventory, Dictionary, Document, Service) and Report components along with the Data Provider tool suite.
- **Build 2** – This build consists of the Distribution subsystem including the Search and Monitor components along with a revised web site and general portal applications.
- **Build 3** – This build consists of enhanced user capabilities include the Order and Subscription components along with integration of Discipline Node applications and science services.

The Harvest tool, implemented in conjunction with the Registry service, is scheduled for delivery in Build 1. This initial delivery will support test collection generation and registration. Additional capabilities are planned for follow-on deliveries as testing progresses and the data model matures.

The implementation platform for the Harvest tool is the Java 2 Platform Standard Edition 6.0. In addition, development will utilize publically available libraries for command-line option handling, message handling, file system access and HTTP interfacing. It will also utilize the PDS common library for parsing and reading product labels for metadata extraction.

The scenario for the preferred deployment is to run an instance of the Harvest tool and an instance of the Registry service locally at the Node. This will allow for the registration of PDS3 and PDS4 products with separate configurations of the Harvest tool. The following diagram depicts this deployment scenario:

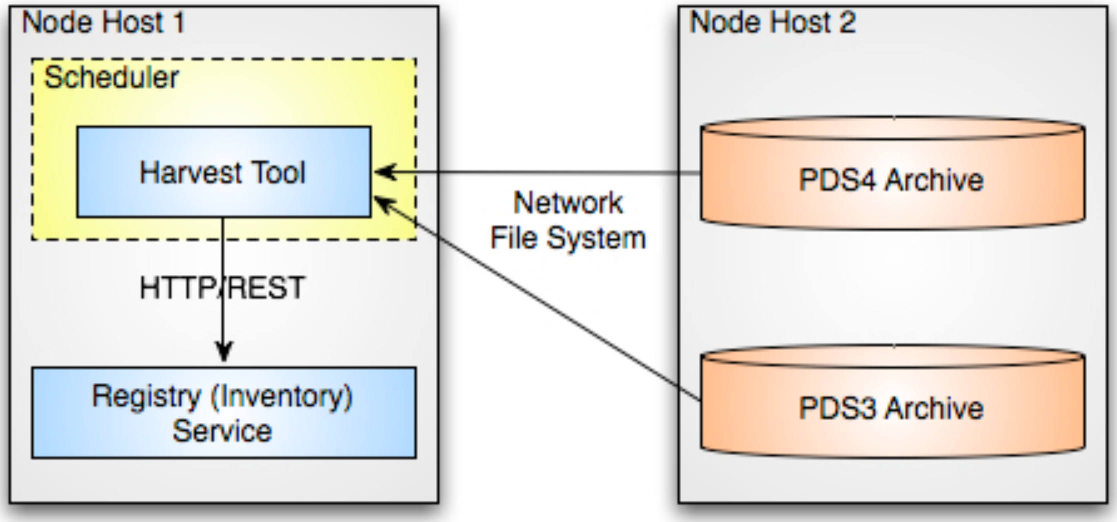

Figure 4: Harvest Tool Deployment (Local Registry)

The Harvest tool provides for execution from the command-line as well as from a scheduler application. A network file system enables access to the PDS3 and PDS4 archive data repositories. The phrase “network file system” is used generically here since the actual implementation may vary from Node to Node based on the choice of platform available at the Node. The Harvest tool submits product registrations to the Registry service via its REST-based interface using the Hypertext Transfer Protocol (HTTP).

A second deployment scenario involves deploying the Harvest tool at the Node but having the product registrations submitted to a remote Registry service hosted at the Engineering Node. This is not the preferred scenario but could be useful during the Build 1 checkout phase where only one or two candidate Nodes will have the preferred deployment. The following diagram depicts this deployment scenario:

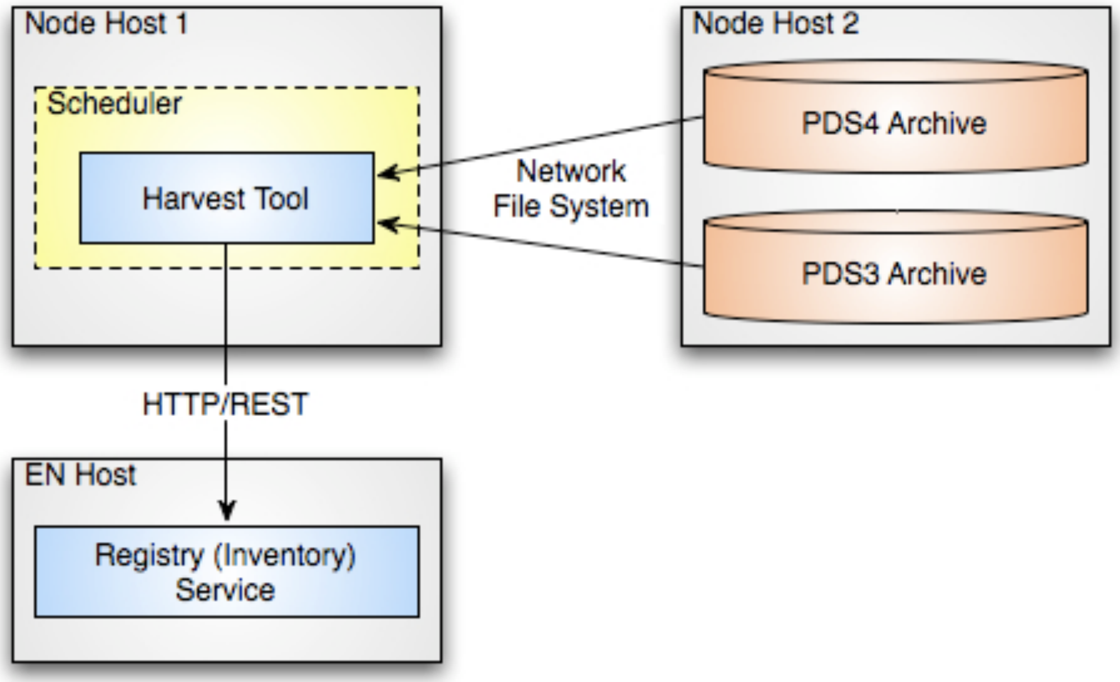

Figure 5: Harvest Tool Deployment (Remote Registry)

9.0 DETAILED DESIGN

This section offers a more detailed look at certain aspects of the Harvest Tool design. The following diagram details the activity flow of the software:

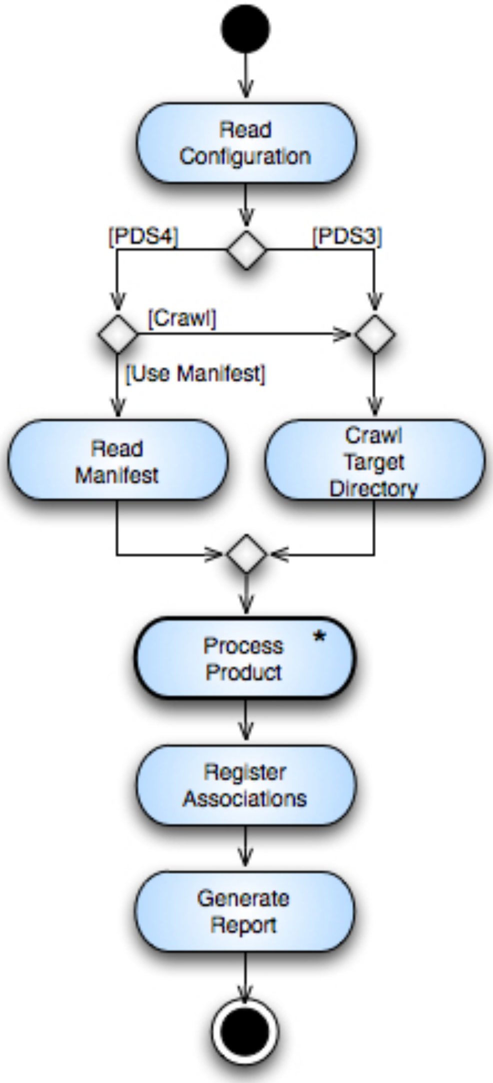

Figure 6: Harvest Tool Activity (Overview)

The activity titled “Process Product *” in the diagram above represents an iteration over all candidate products identified via the provided manifest or by crawling the specified target directory.

9.1 Process Product

The following diagram details the activity flow for this activity:

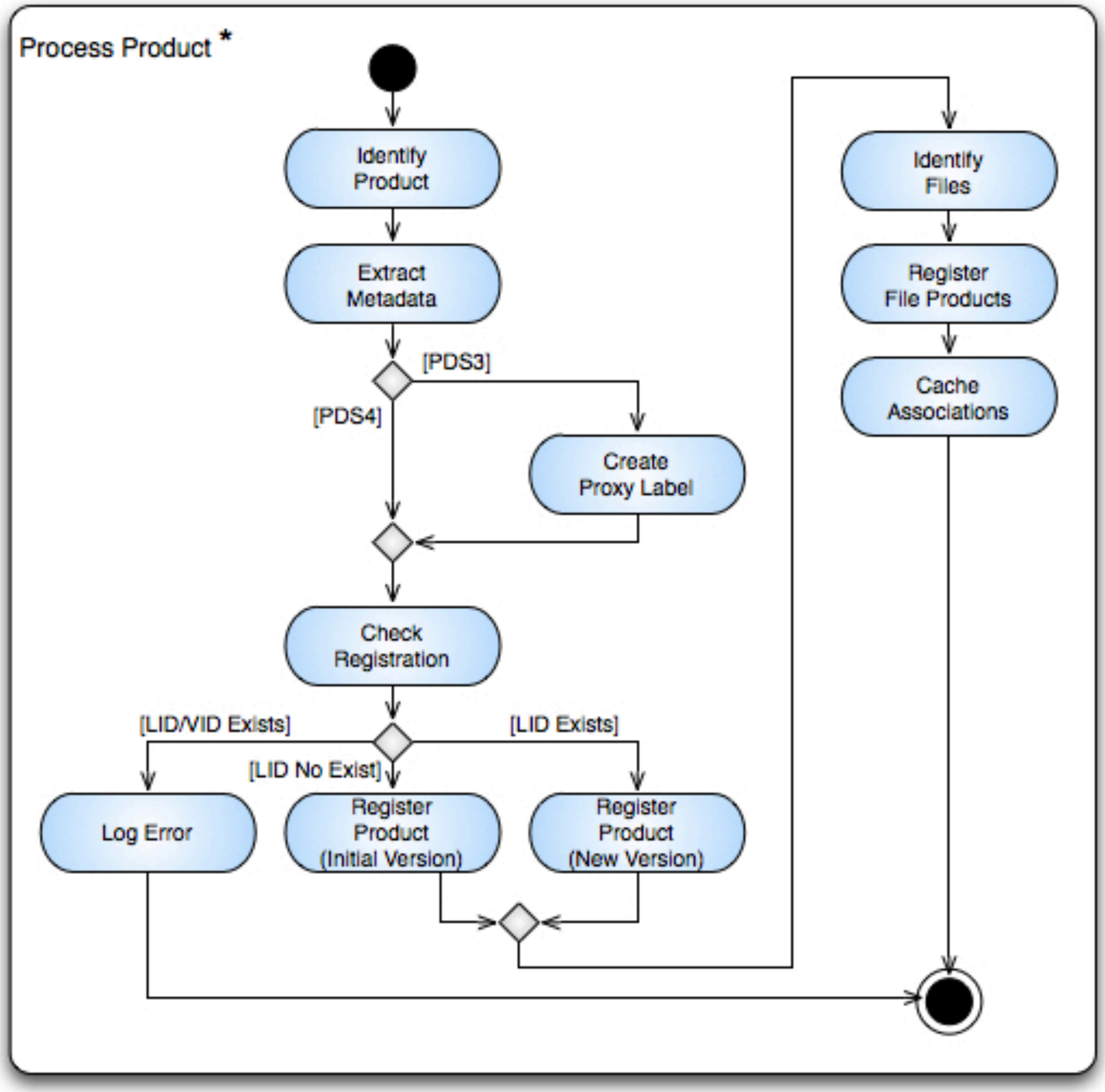

Figure 7: Harvest Tool Activity (Process Product)

Although not depicted in the diagram above, the Extract Metadata activity could result in one of two logged errors. The first is that the product label is not well-formed XML and the second is that the product type specified in the label does not match one of the product types specified in the Harvest tool configuration file.

The Register File Products activity identified above requires Harvest to identify the files associated with the registered product and then register a separate product for each file. Every product will have at least one associated file, which is the product label file. The information for this file is gleaned from the bundle or collection inventory or from crawling a bundle or collection directory. Any other associated files for a product are defined in the product label with a *File_Area_** block. Once the file-specific products have been registered, an association is registered for each associating the file with its parent product.

9.2 Register Associations

In general, references are specified in PDS product labels in order to direct users to additional information with respect to a given product. The Registry and Search services may also use these references to represent explicit links between products. This section focuses on how the Harvest tool identifies and represents these references to the Registry service.

References in the PDS product label come in two forms, specified by the Logical Identifier / Version Identifier (LIDVID) combination or just specified by the Logical Identifier (LID). Both forms of references identify the type of reference being specified. A LIDVID identifies a single unique product within PDS where the LID identifies a set of products (multiple versions). Although a LIDVID is more precise, a LID is utilized when the reference should always identify the latest version of a product.

9.2.1 LIDVID-Based References

In general, LIDVID-based references are registered as associations in the registry. The exception to this rule is if the registry configuration from the model excludes a given reference type. Examples of LIDVID-based references include a collection referencing its member products and a data product referencing calibration or browse products.

9.2.2 LID-Based References

Since LID-based references cannot be tied to a specific product in the registry, they will not result in a registered association. LID-based references are handled as follows:

Bundle to Member Collection
These references are LID-based and have different reference type values (e.g., has_data_collection) based on the collection referenced. For each reference, Harvest creates a slot named according to the reference type and populates it with the LID for the referenced collection.

Collection to Member Product
These references may be LID-based but have a single reference type (typically has_member) specified in the collection product. Harvest creates a slot named according to the reference type and populates it with the LIDs for each product that the collection references with just a LID. For products referenced with LIDVIDs, Harvest registers a corresponding association.

Any Product to Context Product

The model identifies classification schemes for each of the context product types (e.g., investigation, instrument, target, etc.). Harvest, based on the reference type (e.g., has_investigation, has_instrument, has_target, etc.), identifies a reference to a context product, looks up the corresponding classification node and then adds that classification to the product metadata upon registration.

Any Product to Another Product
These references are handled just like the Bundle to Member Collection references.

APPENDIX A ACRONYMS

The following acronyms pertain to this document:

API	Application Programming Interface
CAS	Catalog and Archive Service
CLI	Command-Line Interface
HTTP	Hypertext Transfer Protocol
JPL	Jet Propulsion Laboratory
LID	Logical Identifier
LIDVID	Logical Identifier / Version Identifier
NASA	National Aeronautics and Space Administration
OODT	Object Oriented Data Technology
PDS	Planetary Data System
PDS3	Version 3.8 of the PDS Standards
PDS4	Version 4 of the PDS Standards
REST	Representational State Transfer
SDD	Software Design Document
SRD	Software Requirements Document
UC	Use Case
XML	Extensible Markup Language