# Harvest Tool v.1.0.0

for the Planetary Data System

NASA Jet Propulsion Laboratory,
California Institute of Technology

10 November 2011

# Table of Contents

## 1.1 Overview

### About Harvest Tool

The Harvest 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 Service.

Please send comments, change requests and bug reports to the PDS Operator at pds_operator@jpl.nasa.gov.

## 1.2 Release Notes

### Release Notes

The purpose of this section is to provide a description of a Harvest Tool release including any impact that the new or modified capabilities will have on the Discipline Nodes or the PDS user community. If viewing the web-based version of this document, a somewhat itemized list of changes for each release can be found on the Release Changes page.

### Release 1.0.0

This release of the Harvest Tool is a component of the integrated release 2.0.0 of the PDS 2010 System. This is an operational release of the system components to date. The new or modified capabilities for this release are as follows:

- Added support for providing access URLs for registered products.
- Modified to convert all cross references from PDS product labels into slots for a registered product instead of registering a separate association for each one.
- Fixed an issue where the directory path was not determined correctly for document products.

### Release 0.4.0

This release of the Harvest Tool is a component of the integrated release 1.2.0 of the PDS 2010 System. This release is intended as a prototype release in support of the assessment of the PDS4 standards and the system components to date. The new or modified capabilities for this release are as follows:

- Added support for registering PDS3 products with proxy labels.
- Added the capability to perform file object registration enabling file-level tracking.
- Modified to represent LID-based references as slots instead of associations.
- Modified to register a package with each run enabling support for actions like approval and deletion to be applied to all members of a package in the Registry Service.
- Modified to keep pace with changes occurring in the PDS4 data model, the Registry Service interface and the Security Service.

### Release 0.3.0

This release of the Harvest Tool is a component of the integrated release 1.1.0 of the PDS 2010 System. This release is intended as a prototype release in support of the assessment of the PDS4 standards and the system components to date. The new or modified capabilities for this release are as follows:

- Added the capability to validate products prior to registration.
- Added the capability to execute the tool as a daemon.
- Modified and made more robust the handling of association registrations.
- Modified to keep pace with changes occurring in the PDS4 data model.

The liens for this release are as follows:

- Needs to support registration of PDS3 products.

## Release 0.2.0

This release of the Harvest Tool is a component of the integrated release 1.0.0 of the PDS 2010 System. This release is intended as a prototype release in support of the assessment of the PDS4 standards. The new or modified capabilities for this release are as follows:

- Added support for namespaces within product labels.
- Added support to register Bundle products including the Collection products that they reference.
- Updated to support the latest changes to the PDS4 model.
- Updated to support table-formatted Collection products.
- Updated the log contents to include summary information and product GUIDs.

The liens for this release are as follows:

- Needs to support validation of products prior to registration.
- Does not yet support recognition of products that have already been registered versus newly discovered products. This is required to function in a true crawler scenario.

## Release 0.1.0

This release of the Harvest Tool is a component of the integrated release 0.1.0 of the PDS 2010 System. This release is intended as a prototype release in support of the demonstration at the Management Council Face-to-Face meeting in August 2010. This initial release of the tool provides the capability to crawl a data collection, discover data products and register them in a target Registry Service.

## 1.3 Installation

### Installation

This section describes how to install the Harvest Tool contained in the *harvest* package. The following topics can be found in this section:

- System Requirements
- Unpacking the Package
- Configuring the Tool
- Configuring the Environment

### System Requirements

The Harvest Tool was developed using Java and will run on any platform with a supported Java Runtime Environment (JRE). The software was specifically developed under Java version 1.6 and has only been tested with this version. The following commands test the local Java installation in a UNIX-based environment:

```
% which java
/usr/bin/java

% java -version
java version "1.6.0_26"
Java(TM) SE Runtime Environment (build 1.6.0_26-b03-384-10M3425)
Java HotSpot(TM) 64-Bit Server VM (build 20.1-b02-384, mixed mode)
```

The first command above checks whether the *java* executable is in the environment's path and the second command reports the version. If Java is not installed or the version is not at least 1.6, Java will need to be downloaded and installed in the current environment. Consult the local system administrator for installation of this software. For the do-it-yourself crowd, the Java software can be downloaded from the Oracle Java Download page. The software package of choice is the Java Standard Edition (SE) 6, either the JDK or the JRE package. The JDK package is not necessary to run the software but could be useful if development and compilation of Java software will also occur in the current environment.

### Unpacking the Package

Download the *harvest* package from the PDS FTP site. The binary distribution is available in identical zip or tar/gzip packages. Unpack the selected binary distribution file with one of the following commands:

```
% unzip harvest-1.0.0-bin.zip
or
% tar -xzvf harvest-1.0.0-bin.tar.gz
```

Note: Depending on the platform, the native version of *tar* may produce an error when attempting to unpack the distribution file because many of the file paths are greater than 100 characters. If available, the GNU version of tar will resolve this problem. If that is not available or cannot be installed, the zipped package will work just fine in a UNIX environment.

The commands above result in the creation of the *harvest-1.0.0* directory with the following directory structure:

- **README.txt**
  A README file directing the user to the available documentation for the project.
- **LICENSE.txt**
  The copyright notice from the California Institute of Technology detailing the restrictions regarding the use and distribution of this software. Although the license is strictly worded, the software has been classified as Technology and Software Publicly Available (TSPA) and is available for *anyone* to download and use.
- **bin/**
  This directory contains batch and shell scripts for executing the tool.
- **doc/**
  This document directory contains a local web site with the Harvest Tool Guide, javadoc, unit test results and other configuration management related information. Just point the desired web browser to the *index.html* file in this directory.
- **examples/**
  This directory contains examples of the policy file used for specifying how the Harvest Tool discovers products and extracts metadata for registration.
- **lib/**
  This directory contains the dependent jar files for the tool along with the executable jar file (harvest-1.0.0.jar) containing the Harvest Tool software.
- **keystore/**
  This directory contains the keystore file needed for Harvest to support product registration to a secured

instance of the PDS Registry.

## Configuring the Tool

By default, the Harvest Tool comes configured to access the Registry Service at the specific endpoint *http://localhost:8080/registry*. This should be modified to represent the endpoint of the Registry Service target installation. To change the endpoint, the *Harvest* script should be modified as follows:

```
${JAVA_HOME}/bin/java -Dpds.registry="http://localhost:8080/registry" \
-Dpds.security.keystore="${KEYSTORE}" -jar ${HARVEST_JAR} "$@"

  should be changed to (if hosted at node.nasa.gov on port 80):

${JAVA_HOME}/bin/java -Dpds.registry="http://node.nasa.gov/registry" \
-Dpds.security.keystore="${KEYSTORE}" -jar ${HARVEST_JAR} "$@"
```

To change the endpoint for Windows, the *Harvest.bat* script should be modified as follows:

```
"%JAVA_HOME%"\bin\java -Dpds.registry="http://localhost:8080/registry" \
-Dpds.security.keystore="%KEYSTORE%" -jar "%HARVEST_JAR%" %*

  should be changed to (if hosted at node.nasa.gov on port 80):

"%JAVA_HOME%"\bin\java -Dpds.registry="http://node.nasa.gov/registry" \
-Dpds.security.keystore="%KEYSTORE%" -jar "%HARVEST_JAR%" %*
```

The examples above have been broken into multiple lines for readability. The commands should be reassembled into a single line.

### Configuring to a Secured Instance of the PDS Registry

In order for the Harvest Tool to access a secured instance of the Registry Service, a keystore file must first be generated. Follow the instructions in the PDS Security Service Installation Guide on how to generate this keystore file. Once this is generated, copy the keystore file to the *keystore/* directory of the *Harvest* package and rename the file to *tomcat_self_sign_keystore* as this is what the Harvest shell script and batch file look for by default.

## Configuring the Environment

In order to execute the Harvest Tool, the local environment must first be configured appropriately. This section describes how to setup the user environment on UNIX-based and Windows machines.

### UNIX-Based Environment

This section details the environment setup for UNIX-based machines. The binary distribution includes a couple shell scripts that must be executed from the command-line. Setting the *PATH* environment variable to the location of the scripts, enables the shell scripts to be executed from any location on the local machine.

The following command demonstrates how to set the *PATH* environment variable (in Bourne shell), by appending to its current setting:

```
% export PATH=${PATH}:${HOME}/harvest-1.0.0/bin
```

In addition, the shell scripts require that the *JAVA_HOME* environment variable be set to the appropriate location of the Java installation on the local machine. The following command demonstrates how to set the *JAVA_HOME* environment variable:

```
% export JAVA_HOME=/path/to/java/home
```

The system administrator for the local machine may need to be consulted for this location. The path specified should have a *bin* sub-directory that contains the *java* executable. This variable may also be defined within the scripts. Edit the scripts (files without the .bat extension) and change the line in the example above to represent the local Java installation.

### Windows Setup

This section details the environment setup for Windows machines. The binary distribution includes a couple batch scripts that must be executed from the command-line. Setting the *PATH* environment variable to the location of the files, enables the batch scripts to be executed from any location on the local machine.

The following command demonstrates how to set the *PATH* environment variable, by appending to its current setting:

```
C:\> set PATH = %PATH%;C:\harvest-1.0.0\bin
```

In addition, the batch scripts require that the *JAVA_HOME* environment variable be set to the appropriate location of the Java installation on the local machine. The following command demonstrates how to set the

*JAVA_HOME* environment variable:

```
C:\> set JAVA_HOME = C:\path\to\java\home
```

The system administrator for the local machine may need to be consulted for this location. The path specified should have a *bin* sub-directory that contains the *java* executable. This variable may also be defined within the scripts. Edit the scripts (files with the .bat extension) and change the line in the example above to represent the local Java installation. Additional methods for setting Windows environment variables can be found in the Windows System Properties section. If viewing this document in PDF form, see the appendix for details.

# 1.4 Operation

## Operation

The following topics can be found in this section:

- Tool Execution
- Harvest Policy File
- Report Format
- Common Errors

Note: The command-line examples in this section have been broken into multiple lines for readability. The commands should be reassembled into a single line prior to execution.

## Tool Execution

Harvest Tool can be executed in various ways. This section describes how to run the tool, as well as its behaviors and caveats.

### Command-Line Options

The following table describes the command-line options available:

| Command-Line Option | Description |
|---|---|
| -u, --username | Specify a username for authentication with the PDS Security Service. |
| -p, --password | Specify a password associated with the username. |
| -k, --keystore-pass | Specify a keystore password associated with the keystore file being passed into the tool. |
| -l, --log-file | Specify a log file name. Default is standard out. |
| -P, --port | Specify a port number to use if running the tool in persistence mode. See the Persistence Mode section for more details. |
| -w, --wait | Specify the time, in seconds, to wait in between the crawls if running the tool in persistence mode. See the Persistence Mode section for more details. |
| -V, --version | Display the release number and copyright information. |

| Command-Line Option | Description |
|---|---|
| -h, --help | Display Harvest usage. |

### Execute Harvest Tool

This section demonstrates execution of the tool using the command-line options. The examples below execute the tool via the batch/shell script.

The Harvest Tool operates with a policy file to register product metadata. Details on how to create this policy file can be found in the Harvest Policy File section.

The following command demonstrates how to run the Harvest Tool against a policy file, *policy.xml*, using a valid username, password and keystore password, with the output going to standard out:

```
% Harvest policy.xml -u {username} -p {password} -k {keystorePassword}
```

The following command demonstrates how to run the Harvest Tool with the output going to a log file, *log.txt* instead of standard out:

```
% Harvest policy.xml -u {username} -p {password} -k {keystorePassword} -l log.txt
```

When registering product metadata to a non-secured instance of the Registry (such as one running on your local machine), the *-u* and *-p* command-line option flags do not need to be passed into the tool. The following command demonstrates how to run the Harvest Tool to register product metadata to a non-secured instance of the Registry Service, with the output going to a log file:

```
% Harvest policy.xml -l log.txt
```

### Persistence Mode

The Harvest Tool can be run in persistence mode through an XML-RPC accessible web service called a daemon. Under this scenario, the Harvest Tool wakes up periodically, inspects a target directory or directories, and registers the latest products. This section details how to set this up.

In order to run the tool through the daemon, the command-line option flags *-P* and *-w* need to be used. This tells the Harvest Tool the port number to use and how long to sleep in between crawls, respectively. When the daemon is running, it can be accessed through the following url: *http://localhost:{port number}/xmlrpc*. The following command demonstrates launching the Harvest Tool through the daemon on port *9000*, where it will wait 120 seconds in between crawls:

```
% Harvest policy.xml -u {username} -p {password} -k {keystorePassword} -l log.txt -P
9000 -w 120
```

After running the above command, the daemon will be accessible at *http://localhost:9000/xmlrpc*.

In order to stop the daemon from running, a daemon controller is needed. The *bin/* directory of the Harvest Tool release package contains a shell script, *HarvestController*, and a batch file, *HarvestController.bat*, which are used to gracefully shut down the daemon service on a UNIX-like and Windows system, respectively. In addition, they can provide a few additional statistics about the crawling.

The following table describes the command-line options available for the HarvestController:

| Command-Line Option | Description |
|---|---|
| --url | Specify the URL of the daemon service running the Harvest Tool. |
| --operation | Specify a single operation to perform. List of valid operations can be found in the next table. |

The following table describes the operation names available to pass with the *--operation* command-line option:

| Operation Option | Description |
|---|---|
| --stop | Specify this operation to shut down the daemon service. |
| --isRunning | Gives an indication whether the daemon service is running. |
| --getNumCrawls | Returns the number of crawls that have occurred. |
| --getWaitInterval | Returns the time, in seconds, that the crawler has to wait in between crawls. |
| --getMilisCrawling | Returns the amount of milliseconds spent crawling. |
| --getAverageCrawlTime | Returns the average amount of time, in milliseconds, spent during each crawl. |

The following examples demonstrate how to run the HarvestController using a few of the different operations. For demonstration purposes, assume that the daemon service is located at the following url: *http://localhost:9000/xmlrpc*.

### *Determine the Status of the Daemon Service*

The following command is used to find out if the daemon service is still running:

```
% HarvestController --url http://localhost:9000/xmlrpc --operation --isRunning
```

#### *Shutdown the Daemon Service*

The following command demonstrates shutting down the daemon service:

```
% HarvestController --url http://localhost:9000/xmlrpc --operation --stop
```

## Harvest Policy File

The Harvest policy file is an XML-based configuration file that the tool uses to find products and register their metadata. The schema for the policy file can be found in the Harvest Policy Schema section. If viewing this document in PDF form, see the appendix for details. This section details how to setup the policy file to do PDS data product registration.

### PDS4 Data Product Registration

The following is an example of a policy file to perform registration of PDS4 data products:

```
<?xml version="1.0" encoding="UTF-8"?>
<policy>
   <bundles>
      <file>/home/pds4/context-bundle/bundle.xml</file>
   </bundles>
   <collections>
      <file>/home/pds4/insthost/collection_instrument_host.xml</file>
   </collections>
   <directories>
      <path>/home/user/pds4/geo/product_files</path>
      <filePattern>*.xml</filePattern>
   </directories>
   <validation>
      <enabled>true</enabled>
   </validation>
   <storageIngestion>
      <serverUrl>http://localhost:9000</serverUrl>
   </storageIngestion>
   <accessUrls>
      <baseUrl>http://pds.nasa.gov/pds4</baseUrl>
      <baseUrl>file://pds4</baseUrl>
   </accessUrls>
   <candidates>
      <namespace prefix="geo" uri="http://pds.nasa.gov/schema/pds4/geo"/>
      <productMetadata objectType="character_table">
         <xPath>//geo:Product_Identification_Area/geo:creation_date_time</xPath>
         <xPath>//geo:Subject_Area/geo:instrument_name</xPath>
         <xPath>//Subject_Area/observing_system_name</xPath>
      </productMetadata>
      <productMetadata objectType="Product_Target">
         <xPath>//alternate_title</xPath>
         <xPath>//creation_date_time</xPath>
         <xPath>//identifier</xPath>
         <xPath>//Subject_Area/target_name</xPath>
      </productMetadata>
   </candidates>
</policy>
```

This policy file is made up of the following complex type elements: *bundles*, *collections*, *directories*, *validation*, *storageIngestion*, *accessUrls*, *candidates*, and *productMetadata*.

### *bundles*

Specify this element to tell the Harvest Tool to register and crawl a bundle file. The following table describes the elements that are allowed:

| Element Name | Description |
|---|---|
| file | Specify a bundle file. Specify this element tag more than once to point to multiple bundle files. |

In the example above, the Harvest Tool will register the bundle file named */home/pds4/context-bundle/bundle.xml*. It will then crawl the bundle file, looking for collection files to register and process.

### *collections*

Specify this element to tell the Harvest Tool to register and crawl a collection file. Crawling only occurs when the collection file is a primary collection. This is indicated by a value of *true* in the *is_primary_collection* element tag within the collection.

The following table describes the elements that are allowed:

| Element Name | Description |
|---|---|
| file | Specify a collection file. Specify this element tag more than once to point to multiple collection files. |

In the example above, the Harvest Tool will register the collection file named

*/home/pds4/insthost/collection_instrument_host.xml*. It will then crawl the file, looking for products to register if it is a primary collection.

### *directories*

Specify this element to tell the Harvest Tool where to crawl for data products. The following table describes the elements that are allowed:

| Element Name | Description |
|---|---|
| path | Specify a directory path to start crawling. Specify this element tag more than once to point to multiple directories to crawl. |
| filePattern | Specify a file pattern to look for specific files. If omitted, the default is to get all files within a directory. |

In the example above, the Harvest tool will crawl the directory location, */home/user/pds4/geo/product_files*, looking for files that have a *.xml* file extension. The default is to touch all files in the directory if the *filePattern* element is omitted from the policy file.

### *validation*

Specify this element to tell the Harvest Tool to validate a data product before registering it. If the data product does not pass the validation step, the data product will not be registered. The following table describes the elements that are allowed:

| Element Name | Description |
|---|---|
| enabled | Specify a boolean value to tell the Harvest Tool whether or not to validate a data product. |
| modelVersion | Specify a model version to use during validation. By default, validation will be performed against the latest released PDS4 data model. |

By default, if the *validation* element is not specified in the policy file, validation is turned on.

### *storageIngestion*

Specify this element to tell the Harvest Tool to ingest data products to the PDS Storage. The following table describes the elements that are allowed:

| Element Name | Description |
|---|---|
| serverUrl | Specify the url to the PDS Storage Service. |

In the example above, the Harvest Tool will ingest data products to the PDS Storage Service at *http://localhost:9000*. When a data product is ingested to the PDS Storage, it returns a product id which is a reference to the ingested product. This id is placed as a slot in the registry under the slot name *storageServiceProductId.*

### *accessUrls*

Specify this element to provide links to the physical data products. The links will be placed in the registry as slots under the slot name *accessUrl*. The following table describes the elements that are allowed:

| Element Name | Description |
|---|---|
| baseUrl | Specify a base url. This element can be specified more than once if providing more than one link. |

The provided base urls will be prepended to the path of the registered data product. As an example, if a data product is located in /data/product/label.xml, using the base urls from the policy example above, the following access urls will be formed: *http://pds.nasa.gov/pds4/data/product/label.xml* and *file://pds4/data/product/label.xml.*

### *candidates*

Specify this element to tell the Harvest Tool what product types to register and what metadata to extract from a data product. This is a required element in the policy file. The following table describes the elements that are allowed:

| Element Name | Description |
|---|---|
| namespace | Specify to allow the Harvest Tool to extract metadata that is in a namespace other than the default PDS namespace. |
| productMetadata | Specify to tell the tool what object types and what metadata to register. |

By default, the Harvest Tool defines the default namespace to be the PDS namespace, *http://pds.nasa.gov/schema/pds4/pds/v04*. To override this default, specify the *default* attribute in the namespace element and give it a value of *true*. The following makes the *geo* namespace the default namespace:

```
<candidates>
  <namespace prefix="geo" uri="http://pds.nasa.gov/schema/pds4/geo" default="true"/>
         ...
```

Namespaces need to be defined in the Harvest policy file only if the metadata to be extracted exists in a namespace other than the PDS namespace. In the example above, a namespace with the prefix *geo* and uri *http://pds.nasa.gov/schema/pds4/geo* has been defined. This means that any xPath expressions defined in the policy file will be able to use the *geo* prefix to be able to extract metadata that are within the geo namespace. xPaths will be explained in greater detail in the *productMetadata* section.

### *productMetadata*

Specify this element to tell the Harvest Tool what metadata to register. It requires an attribute called *objectType* that tells the Harvest Tool what product types to register. The following table describes the elements that are allowed:

| Element Name | Description |
|---|---|
| xPath | Specify an XPath expression to extract metadata. |

In the example above, the policy file tells the Harvest Tool to look for and register the *character_table* and *Product_Target* object types.

Also in the example is a set of *xPath* elements found under each *productMetadata* element. This defines what metadata to extract from the different products. XPath is a query language that uses path expressions to select nodes in an XML document. These path expressions look very much like expressions in a traditional computer file system. In its simplest form, prepending a // before a name will find the element no matter where it is in the XML file.

The following XPath expression will find the *creation_date_time* element within the default namespace, no matter where this element is located in the file:

```
//creation_date_time
```

The following XPath expression will find the *creation_date_time* element within the geo namespace, no matter where this element is located in the file:

```
//geo:creation_date_time
```

The following XPath expression will find all *target_name* elements that are children of *Subject_Area* within the default namespace:

```
//Subject_Area/target_name
```

The following XPath expression will find all *target_name* elements that are children of *Subject_Area* and that have a value of *MARS*:

```
//Subject_Area/target_name[text()="MARS"]
```

For a more detailed explanation on XPath, go to your favorite search engine and type *XPath tutorial*.

### PDS3 Product Registration

By default, the tool registers discovered PDS3 products under the *Product_Proxy_PDS3* objectType in the registry. Additionally, the tool has to dynamically create certain metadata in order to support ingestion of PDS3 data products into the registry. The Harvesting of PDS3 Data Products section details how the Harvest Tool behaves when registering PDS3 data products. If viewing this document in PDF form, see the appendix for details.

The following is an example of a policy file to perform product registration of PDS3 data products:

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- Example of a Harvest policy configuration file that will do PDS3 data
    product registration  -->
<policy>
  <!-- Specify a single directory containing the PDS3 data products to
    register -->
  <pds3Directory>
    <path>/data/pds3/dataset</path>
    <filePattern>*.LBL</filePattern>
  </pds3Directory>
  <candidates>
    <!-- Harvest will register PDS3 data products under the objectType
      'Product_Proxy_PDS3' -->
    <pds3ProductMetadata>
        <!-- Prefix to add to the LID of a PDS3 product registration -->
        <lidPrefix>URN:JPL:PDS:ENGINEERING</lidPrefix>
        <!-- Associations to register with discovered PDS3 products -->
        <associations>
          <!-- Specify either a LID or LIDVID reference -->
          <association>
            <referenceType>has_Target</referenceType>
            <lidVidReference>URN:NASA:PDS:target.MARS::1.0</lidVidReference>
          </association>
          <association>
            <referenceType>has_Mission</referenceType>
            <lidReference>URN:NASA:PDS:mission.MER</lidReference>
          </association>
        </associations>
        <!-- Register any additional metadata. They will be registered as
          slots with their element names in lowercase form. Default is to
          register metadata defined in the identification area of the
          Product_Proxy_PDS3 schema. -->
        <ancillaryMetadata>
          <elementName>START_DATE_TIME</elementName>
          <elementName>STOP_DATE_TIME</elementName>
        </ancillaryMetadata>
        <includePaths>
          <path>/data/pds3/label</path>
        </includePaths>
    </pds3ProductMetadata>
  </candidates>
</policy>
```

This policy file is made up of the following complex type elements: *pds3Directory*, *pds3ProductMetadata*, *association*, *ancillaryMetadata*, *includePaths*.

### *pds3Directory*

Specify this element to tell the Harvest Tool the directory location to crawl. The following table describes the elements that are allowed:

| Element Name | Description |
|---|---|
| path | Specify a directory location containing the PDS3 data products to register. Only one directory location is allowed per executable run. |
| filePattern | Specify a file pattern to look for specific files. If omitted, the default is to get all files within a directory. |

In the example above, the Harvest Tool will crawl for PDS3 data products starting at the location */data/pds3/dataset*, looking for files with a *.LBL* file extension.

### *pds3ProductMetadata*

Specify this element to tell the Harvest Tool what metadata to ingest into the registry when registering PDS3 data products. This element must be specified within the *candidates* tag as shown in the example. The following table describes the elements that are allowed:

| Element name | Description |
|---|---|
| lidPrefix | Specify a prefix to add to the logical identifier. |
| associations | Specify one or more associations. |
| ancillaryMetadata | Specify ancillary metadata to ingest into the registry for every discovered PDS3 data product. |

In the example above, the logical identifiers of every discovered PDS3 data product will be prefixed with *URN:JPL:PDS:ENGINEERING*.

### *association*

Specify this element to tell the Harvest Tool what associations belong to each discovered PDS3 data product. Specifying one or more *association* elements is allowed and they must be within the *associations* tag as shown in the example. The following table describes the elements that are allowed:

| Element name | Description |
|---|---|
| referenceType | Specify the association type. |
| lidVidReference | Specify a lidvid reference. |
| lidReference | Specify a lid reference. |

Note that *lidVidReference* and *lidReference* cannot be used together within the same *association* tag. Only one can be chosen.

In the example above, each discovered PDS3 product will have two associations: one with a LIDVID of *URN:NASA:PDS:target.MARS::1.0* and association type of *has_Target*, and one with a LID of *URN:NASA:PDS:mission.MER* and association type of *has_Mission*.

### ancillaryMetadata

Specify this element to tell the Harvest tool what additional metadata to register. The following table describes the elements that are allowed:

| Element name | Description |
|---|---|
| elementName | Specify an element name found in the PDS3 data product label. |

In the example above, the values from the following elements will be extracted from a PDS3 product label: *START_DATE_TIME* and *STOP_DATE_TIME*. If they are found in the label, they will be registered as slots in the registry, using their element names in lowercase form as the slot names. In this case, *start_date_time* and *stop_date_time* will be used as slot names in the registry.

### includePaths

Specify this element to tell the Harvest tool the locations of where to find file references specified in a label. By default, the tool will look for the file reference in the location of the label file. The following table describes the elements that are allowed:

| Element name | Description |
|---|---|
| path | Specify the directory location of where to find the file references in a label. This element can be specified more than once to specify multiple search paths. |

In the example above, the tool will look at the */data/pds3/label* directory for file references if they cannot be found in the same location as the label file.

## Report Format

This section describes the contents of the Harvest Tool report. At this time, the Harvest Tool only outputs a series of log messages. The log will report the success or failure of a discovered product attempting to be registered. Additionally, any syntactical errors in a discovered product are reported. A log consists of a severity level, file name, and a message. The following is an example of some of the log messages that can be expected from the Harvest Tool:

```
PDS Harvest Tool Log

Version                     Version 0.5.0-dev
Time                        Fri, Oct 14 2011 at 08:15:53 AM
Registry Location           http://localhost:8080/registry
Registry Package Name       Harvest-Package_20111014081552
Registration Package GUID   urn:uuid:f4cb8733-b86f-4877-ad47-c7fbacf44574

INFO:   XML extractor set to the following default namespace:
http://pds.nasa.gov/schema/pds4/pds/v04
INFO:   [/pds4/context_bundle/Product_Bundle_Context.xml] Begin processing.
INFO:   [/pds4/context_bundle/Product_Bundle_Context.xml] \
Setting LID-based association, 'urn:nasa:pds:Collection_Context.personnel.affil', under
slot name 'has_context_collection'.
INFO:   [/pds4/context_bundle/Product_Bundle_Context.xml] \
Setting LID-based association, 'urn:nasa:pds:Collection_Context.dataset', under slot
name 'has_context_collection'.
INFO:   [/pds4/context_bundle/Product_Bundle_Context.xml] \
Setting LID-based association, 'urn:nasa:pds:Collection_Context.personnel.guest', under
slot name 'has_context_collection'.
INFO:   [/pds4/context_bundle/Product_Bundle_Context.xml] \
Setting LID-based association, 'urn:nasa:pds:Collection_Context.instrument_host', under
slot name 'has_context_collection'.
INFO:   [/pds4/context_bundle/Product_Bundle_Context.xml] \
Setting LID-based association, 'urn:nasa:pds:Collection_Context.instrument', under slot
name 'has_context_collection'.
INFO:   [/pds4/context_bundle/Product_Bundle_Context.xml] \
Setting LID-based association, 'urn:nasa:pds:Collection_Context.mission', under slot
name 'has_context_collection'.
INFO:   [/pds4/context_bundle/Product_Bundle_Context.xml] \
Setting LID-based association, 'urn:nasa:pds:Collection_Context.node', under slot name
'has_context_collection'.
INFO:   [/pds4/context_bundle/Product_Bundle_Context.xml] \
Setting LID-based association, 'urn:nasa:pds:Collection_Context.resource', under slot
name 'has_context_collection'.
INFO:   [/pds4/context_bundle/Product_Bundle_Context.xml] \
Setting LID-based association, 'urn:nasa:pds:Collection_Context.target', under slot name
'has_context_collection'.
INFO:   [/pds4/context_bundle/Product_Bundle_Context.xml] Captured file information for
Product_Bundle_Context.xml
SUCCESS:   [/pds4/context_bundle/Product_Bundle_Context.xml] Successfully registered
product: urn:nasa:pds:context_bundle::1.0
INFO:   [/pds4/context_bundle/Product_Bundle_Context.xml] Product has the following
```

```
GUID: urn:uuid:25f3043e-0655-4d2c-9205-8406a254ec70
SUCCESS:   [/pds4/context_bundle/Product_Bundle_Context.xml] \
Successfully registered product:
urn:nasa:pds:context_bundle:Product_Bundle_Context_20111012.xml::1.0
INFO:   [/pds4/context_bundle/Product_Bundle_Context.xml] Product has the following
GUID: urn:uuid:554175b2-30d6-4465-adf3-795c1c59cf27
INFO:   [/pds4/context_bundle/context_dataset/collection_context_dataset.xml] Begin
processing.
INFO:   [/pds4/context_bundle/context_dataset/collection_context_dataset.xml] No
associations found.
INFO:   [/pds4/context_bundle/context_dataset/collection_context_dataset.xml] Captured
file information for collection_context_dataset_20111012.xml
INFO:   [/pds4/context_bundle/context_dataset/collection_context_dataset.xml] Capturing
file object metadata for manifest_data_set_20111012.tab
SUCCESS:   [/pds4/context_bundle/context_dataset/collection_context_dataset.xml] \
Successfully registered product: urn:nasa:pds:Collection_Context.dataset::1.0
INFO:   [/pds4/context_bundle/context_dataset/collection_context_dataset.xml] Product
has the following GUID: urn:uuid:fd5eeaf9-2de7-4022-b440-88a2ac4a694b
SUCCESS:   [/pds4/context_bundle/context_dataset/collection_context_dataset.xml] \
Successfully registered product:
urn:nasa:pds:Collection_Context.dataset:collection_context_dataset_20111012.xml::1.0
INFO:   [/pds4/context_bundle/context_dataset/collection_context_dataset.xml] Product
has the following GUID: urn:uuid:6f180715-ccde-4a3d-9404-e5a1128a3ce7
SUCCESS:   [/pds4/context_bundle/context_dataset/collection_context_dataset.xml] \
Successfully registered product:
urn:nasa:pds:Collection_Context.dataset:manifest_data_set_20111012.tab::1.0
INFO:   [/pds4/context_bundle/context_dataset/collection_context_dataset.xml] Product
has the following GUID: urn:uuid:21a13088-b552-4df6-b855-28532c5e4a70

    ....

Summary:

14362 of 14362 file(s) processed, 0 skipped
28669 of 28701 products registered.
89025 of 89061 associations registered, 0 skipped


End of Log
```

## Common Errors

Execution of the Harvest Tool may result in the following message appearing in the log:

```
INFO:   XML extractor set to the following default namespace: \
http://pds.nasa.gov/schema/pds4/pds
INFO:   [/pds4/VG2PLS_archive/Product_Archive_Bundle.xml] Begin processing.
SKIP:   [/pds4/VG2PLS_archive/Product_Archive_Bundle.xml] No product_class \
element found.
```

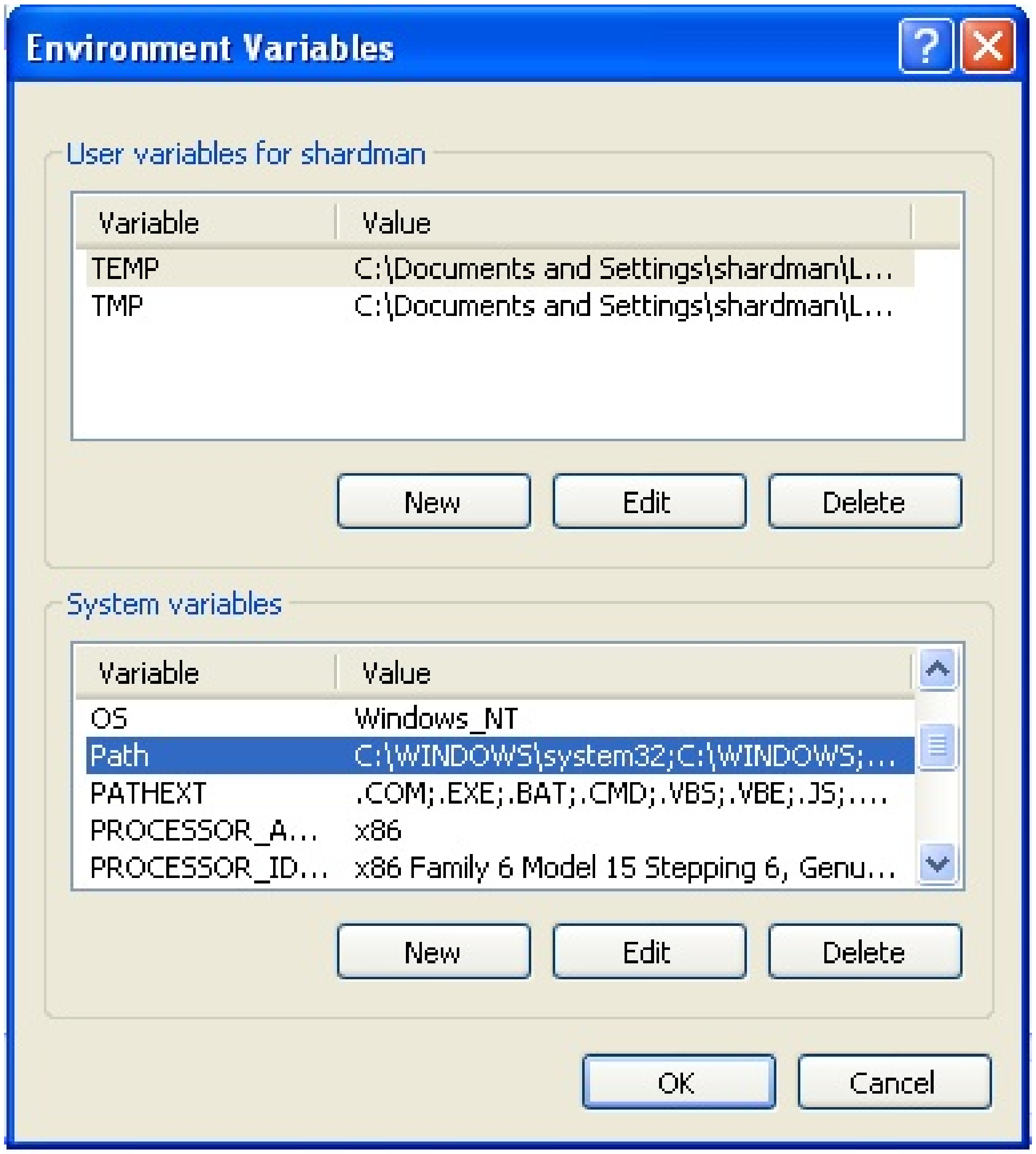
The message above is normally the result of a namespace mismatch between the Harvest Tool configuration and the product labels being registered. See the PDS4 Data Product Registration section above for details on specifying the namespace in the configuration file. By the way, the message could be telling the truth where the product label does not contain the <product_class> element. If this is the case, then the file is not a valid PDS product label.

# 1.5 Appendix A - Windows System Properties

### Windows System Properties

The required environment variables for the *harvest* package can also be set through the Windows system properties panel. The *Path* environment variable can be modified as follows:

- Right-click on *My Computer* icon on your desktop and select the *Properties* menu item.
- Navigate to the *Advanced* tab and select the *Environment Variables* button. At this point, you should now see a window like the one below:



| Variable | Value |
|---|---|
| TEMP | C:\Documents and Settings\shardman\L... |
| TMP | C:\Documents and Settings\shardman\L... |

| Variable | Value |
|---|---|
| OS | Windows_NT |
| Path | C:\WINDOWS\system32;C:\WINDOWS;... |
| PATHEXT | .COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;.... |
| PROCESSOR_A... | x86 |
| PROCESSOR_ID... | x86 Family 6 Model 15 Stepping 6, Genu... |

- Highlight the *Path* variable in the System Variables list and select the *Edit* button.
- Append to the current contents of the variable, the path to the *bin* directory within *harvest* package. Separate the package path from the current contents of the variable with a semicolon.
- Select the *OK* button when you are finished editing the *Path* variable, then select the *OK* button on the Environment Variables window to apply the changes.

New environment variables (e.g., JAVA_HOME) may also be specified in the system properties panel. Instead of selecting the *Edit* button from the System Variables list, select the *New* button and enter the variable name and value. Select the *OK* button when you are finished, then select the *OK* button on the Environment Variables window to apply the changes.

Note: If you already have a DOS window open, you will need to close and re-open the window for the environment variable changes to take effect.

# 1.6 Appendix B - Harvest Policy Schema

## Harvest Policy Schema

The XML schema file for validating policy files follows:

```
<?xml version="1.0" encoding="UTF-8" ?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
            xmlns:jxb="http://java.sun.com/xml/ns/jaxb"
            jxb:version="2.0">

<xsd:element name="path" type="xsd:string"/>
<xsd:element name="file" type="xsd:string"/>
<xsd:element name="filePattern" type="xsd:string"/>
<xsd:element name="xPath" type="xsd:string"/>
<xsd:element name="enabled" type="xsd:boolean" default="true"/>
<xsd:element name="lidPrefix" type="xsd:string"/>
<xsd:element name="lidReference" type="xsd:string"/>
<xsd:element name="lidVidReference" type="xsd:string"/>
<xsd:element name="referenceType" type="xsd:string"/>
<xsd:element name="elementName" type="xsd:string"/>
<xsd:element name="modelVersion" type="xsd:string"/>
<xsd:element name="baseUrl" type="xsd:string"/>
<xsd:element name="serverUrl" type="xsd:string"/>
<xsd:element name="dataTransferType" type="xsd:string"/>

<xsd:attribute name="objectType" type="xsd:string"/>
<xsd:attribute name="prefix" type="xsd:string"/>
<xsd:attribute name="uri" type="xsd:string"/>
<xsd:attribute name="default" type="xsd:boolean" default="false"/>

<xsd:element name="directories" type="Directory"/>
<xsd:complexType name="Directory">
   <xsd:sequence>
      <xsd:element ref="path" maxOccurs="unbounded"/>
      <xsd:element ref="filePattern" minOccurs="0" maxOccurs="unbounded"/>
   </xsd:sequence>
</xsd:complexType>

<xsd:element name="bundles" type="Bundle"/>
<xsd:complexType name="Bundle">
   <xsd:sequence>
      <xsd:element ref="file" maxOccurs="unbounded"/>
   </xsd:sequence>
</xsd:complexType>

<xsd:element name="collections" type="Collection"/>
```

```
<xsd:complexType name="Collection">
   <xsd:sequence>
      <xsd:element ref="file" maxOccurs="unbounded"/>
   </xsd:sequence>
</xsd:complexType>

<xsd:element name="namespace" type="Namespace"/>
<xsd:complexType name="Namespace">
   <xsd:attribute ref="prefix" use="required"/>
   <xsd:attribute ref="uri" use="required"/>
   <xsd:attribute ref="default"/>
</xsd:complexType>

<xsd:element name="productMetadata" type="Pds4ProductMetadata"/>
<xsd:complexType name="Pds4ProductMetadata">
   <xsd:sequence>
      <xsd:element ref="xPath" maxOccurs="unbounded"/>
   </xsd:sequence>
   <xsd:attribute ref="objectType" use="required"/>
</xsd:complexType>

<xsd:element name="pds3ProductMetadata" type="Pds3ProductMetadata"/>
<xsd:complexType name="Pds3ProductMetadata">
   <xsd:all>
      <xsd:element ref="lidPrefix" minOccurs="0"/>
      <xsd:element ref="associations" minOccurs="0"/>
      <xsd:element ref="ancillaryMetadata" minOccurs="0"/>
   </xsd:all>
</xsd:complexType>

<xsd:element name="candidates" type="Candidate"/>
<xsd:complexType name="Candidate">
   <xsd:sequence>
      <xsd:element ref="pds3ProductMetadata" minOccurs="0"/>
      <xsd:element ref="namespace" minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element ref="productMetadata" minOccurs="0" maxOccurs="unbounded"/>
   </xsd:sequence>
</xsd:complexType>

<xsd:element name="validation" type="Validation"/>
<xsd:complexType name="Validation">
   <xsd:sequence>
      <xsd:element ref="enabled" minOccurs="0"/>
      <xsd:element ref="modelVersion" minOccurs="0"/>
   </xsd:sequence>
</xsd:complexType>

<xsd:element name="associations" type="Associations"/>
<xsd:complexType name="Associations">
   <xsd:sequence>
      <xsd:element ref="association" minOccurs="0" maxOccurs="unbounded"/>
   </xsd:sequence>
</xsd:complexType>

<xsd:element name="association" type="Association"/>
<xsd:complexType name="Association">
   <xsd:sequence>
```

```
      <xsd:element ref="referenceType"/>
      <xsd:choice>
         <xsd:element ref="lidVidReference"/>
         <xsd:element ref="lidReference"/>
      </xsd:choice>
   </xsd:sequence>
</xsd:complexType>

<xsd:element name="ancillaryMetadata" type="AncillaryMetadata"/>
<xsd:complexType name="AncillaryMetadata">
   <xsd:sequence>
      <xsd:element ref="elementName" maxOccurs="unbounded"/>
   </xsd:sequence>
</xsd:complexType>

<xsd:element name="pds3Directory" type="Pds3Directory"/>
<xsd:complexType name="Pds3Directory">
   <xsd:sequence>
      <xsd:element ref="path"/>
      <xsd:element ref="filePattern" minOccurs="0" maxOccurs="unbounded"/>
   </xsd:sequence>
</xsd:complexType>

<xsd:element name="storageIngestion" type="StorageIngestion"/>
<xsd:complexType name="StorageIngestion">
   <xsd:sequence>
      <xsd:element ref="serverUrl"/>
      <xsd:element ref="dataTransferType" minOccurs="0"/>
   </xsd:sequence>
</xsd:complexType>

<xsd:element name="accessUrls" type="AccessUrl"/>
<xsd:complexType name="AccessUrl">
   <xsd:sequence>
      <xsd:element ref="baseUrl" maxOccurs="unbounded"/>
   </xsd:sequence>
</xsd:complexType>

<xsd:element name="policy">
  <xsd:complexType>
     <xsd:all>
        <xsd:element ref="bundles" minOccurs="0"/>
        <xsd:element ref="collections" minOccurs="0"/>
        <xsd:element ref="directories" minOccurs="0"/>
        <xsd:element ref="pds3Directory" minOccurs="0"/>
        <xsd:element ref="validation" minOccurs="0"/>
        <xsd:element ref="storageIngestion" minOccurs="0"/>
        <xsd:element ref="accessUrls" minOccurs="0"/>
        <xsd:element ref="candidates"/>
     </xsd:all>
  </xsd:complexType>
</xsd:element>
</xsd:schema>
```

# 1.7 Appendix C - Harvesting of PDS3 Data Products

## Harvesting of PDS3 Data Products

This section details how the Harvest Tool behaves when registering PDS3 data products. The tool does the following when it registers a PDS3 data product:

- Create a LIDVID
- Create a Name
- Ingest Default Metadata

### Create a LIDVID

The LIDVID for a PDS3 data product is made up of the following: given prefix (from the policy file) + *DATA_SET_ID* + *PRODUCT_ID* + :: + *PRODUCT_VERSION*. If a prefix is not specified in the policy file or the *DATA_SET_ID* is not found in the label, then they are not included in the formation of the LIDVID. If the *PRODUCT_ID* is not found in the label, then the tool will default to using the file name (without the extension) as part of the LIDVID. If the *PRODUCT_VERSION* is not found in the label, then the tool will default to using *1.0*.

#### *Example*

Suppose a label named *PDS3LABEL.LBL* contains the following metadata:

```
DATA_SET_ID     = "MRO-M-HIRISE-3-RDR-V1.0"
PRODUCT_ID      = "PSP_001332_2620_RED"
PRODUCT_VERSION = "1.0"
```

If the policy file specified a LID prefix of *URN:NASA:PDS*, then the LIDVID of this product would be *URN:NASA:PDS:MRO-M-HIRISE-3-RDR-V1.0:PSP_001332_2620_RED::1.0*. Now, if for example, the *PRODUCT_ID* was missing from the label, the tool will use the file name and the LIDVID would become *URN:NASA:PDS:MRO-M-HIRISE-3-RDR-V1.0:PDS3LABEL::1.0* .

### Create a Name

Each registered product in the PDS Registry has a *name* attribute, which maps to the *title* tag in the *Identification_Area* of a PDS4 product label. When a PDS3 product is registered, the tool has to dynamically generate a title (or name) to ingest into the PDS Registry.

The name is made up of the following fields: *INSTRUMENT_HOST_NAME* + *INSTRUMENT_NAME* (or *INSTRUMENT_ID*). *INSTRUMENT_ID* is used only if *INSTRUMENT_NAME* is not found in the label. If none of these elements are found in the label, then the title will default to *PDS3 Data Product*.

### *Example*

Suppose a label contained the following metadata:

```
INSTRUMENT_HOST_NAME    =   "MARS RECONNAISSANCE ORBITER"
INSTRUMENT_ID           =   "HIRISE"
INSTRUMENT_NAME         =   "HIGH RESOLUTION IMAGING SCIENCE EXPERIMENT"
```

The resulting name becomes *MARS RECONNAISSANCE ORBITER HIGH RESOLUTION IMAGING SCIENCE EXPERIMENT*. If *INSTRUMENT_NAME* is not found in the label, then the name ends up being *MARS RECONNAISSANCE ORBITER HIRISE* .

### Ingest Default Metadata

For each PDS3 data product that gets registered, the tool will automatically ingest a set of metadata. The following table shows what elements the tool looks for in a PDS3 product label and what slot name it uses when ingesting the metadata into the registry:

| Element Name | Slot Name |
|---|---|
| PRODUCT_CREATION_TIME | last_modification_date_time |
| TARGET_NAME | target_name |
| INSTRUMENT_NAME | instrument_name |
| INSTRUMENT_ID | instrument_name |
| INSTRUMENT_HOST_NAME | instrument_host_name |

For the most part, the slot name is the element name in lowercase form with the exception of *PRODUCT_CREATION_TIME* and *INSTRUMENT_ID*.