Harvest Tool v.0.3.0

for the Planetary Data System

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 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

System Requirements

The Harvest Tool was developed using Java and will run on any platform with a supported Java Runtime Environment (JRE). The tool was specifically developed under Sun Java version 1.6, so the tool will execute correctly under 1.6 and future versions.

Since the tool was developed using Sun's Java, this is the preferred Java environment for operation. The Sun Java package can be obtained from the Sun Java web site. Other Java environments are relatively compatible with Sun's Java.

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-0.3.0-bin.zip
or
% tar -xzvf harvest-0.3.0-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-0.3.0* 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 your favorite 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-0.3.0.jar) containing the Harvest Tool software.

1.4 **Operation**

....

Operation

The following topics can be found in this section:

- Tool Setup
- Additional Tool Setup
- Tool Execution
- Report Format

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 Setup

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

UNIX-Based Setup

This section details the environment setup for UNIX-based machines. The preferred method is to specify the shell script, *Harvest*, on the command-line. Setting the *PATH* environment variable to the location of the script, enables the shell script to be executed from any location on the user's machine.

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

```
% setenv PATH ${PATH}:$HOME/harvest-0.3.0/bin
```

The tool can now be executed via the shell script as demonstrated in the following example:

```
% Harvest <policy file> <command-line arguments>
```

Additional methods for setting up a UNIX-based environment can be found in the UNIX Setup Options section. If viewing this document in PDF form, see the appendix for details.

Windows Setup

This section details the environment setup for Windows machines. The preferred method is to specify the batch file, *Harvest.bat*, on the command-line. Setting the *PATH* environment variable to the location of the file, enables the batch file to be executed from any location on the user's machine.

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

```
C:\> set PATH = %PATH%;C:\harvest-0.3.0\bin
```

The tool can now be executed via the batch file as demonstrated in the following example:

```
C:\> Harvest <policy file> <command-line arguments>
```

Additional methods for setting up a Windows environment can be found in the Windows Setup Options section. If viewing this document in PDF form, see the appendix for details.

Additional Tool Setup

This section details how to re-configure the Harvest Tool to interface with another instance of the Registry Service.

The Harvest Tool points to the Registry Service via the following Java System Property:

System Property Name	Description
pds.registry	Specify the URL to the Registry Service. This property is required.

By default, the Harvest shell script and batch file point to local installation of the Registry Service. The sections below detail how to modify these scripts to point to another instance of this service.

UNIX-Based Users

Open the *Harvest* shell script and go to the last line in the file. It should look like the following:

```
% java -Dpds.registry="http://localhost:8080/registry-service" -jar ${HARVEST_JAR} "$@"
```

Replace the URL value of *pds.registry* with the URL to the desired instance of the Registry. For example, making the following change to the script will have Harvest pointing to the secured operational instance of the Registry at the Engineering Node:

```
% java -Dpds.registry="https://pdsops2.jpl.nasa.gov/registry-service" -jar
${HARVEST_JAR} "$@"
```

Windows-Based Users

Open the *Harvest* batch and go to the last line in the file. It should look like the following:

```
% java -Dpds.registry="http://localhost:8080/registry-service" -jar "%HARVEST_JAR%" %*
```

Replace the URL value of *pds.registry* with the URL to the desired instance of the Registry. For example, making the following change to the batch file will have Harvest pointing to a secured operational instance of the Registry at the Engineering Node:

```
% java -Dpds.registry="https://pdsops2.jpl.nasa.gov/registry-service" -jar
"%HARVEST_JAR%" %*
```

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.
-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 continuous mode. See the Continuous Mode section for more details.
-w, --wait	Specify the time, in seconds, to wait in between the crawls. This option is used only when running the tool in continuous mode.
-V, --version	Display the release number and copyright information.
-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. Alternate methods for executing the tool can be found in the Tool Setup section.

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 and password, with the output going to standard out:

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

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} -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
```

Harvest Policy File

The Harvest policy file is an XML 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.

The following is an example of a policy file:

```xml
<?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>
   <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>
```

The policy file is made up of the following complex type elements: *bundles*, *collections*, *directories*, *validation*, *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.

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

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*. 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*.

Continuous Mode

The Harvest Tool can be run in continuous 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} -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 operation names is found in the next table.

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

Operation Name	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.

Operation Name	Description
--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.*

Shutdown the daemon service

The following command demonstrates shutting down the daemon service:

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

Find Out 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
```

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 0.2.0-dev
Time              Wed, Sep 29 2010 at 02:02:27 PM
Registry Location http://localhost:8080/registry-service

INFO:   [C:\pds4\geo\BUGLAB_Archive_Bundle.xml] Begin processing.
SKIP:   [C:\pds4\geo\BUGLAB_Archive_Bundle.xml] 'archive bundle' is not
an object type found in the policy file.
INFO:   [C:\pds4\geo\schema\BUGLAB_Archive_Bundle.xml] Begin processing.
```

```
SKIP:    [C:\pds4\geo\schema\BUGLAB_Archive_Bundle.xml] 'XML_Schema' is not
an object type found in the policy file.
INFO:    [C:\pds4\geo\schema\BUGLAB_Collection.xml] Begin processing.
SKIP:    [C:\pds4\geo\schema\BUGLAB_Collection.xml] 'XML_Schema' is not an
object type found in the policy file.
INFO:    [C:\pds4\geo\schema\BUGLAB_Schema_Collection.xml] Begin processing.
SKIP:    [C:\pds4\geo\schema\BUGLAB_Schema_Collection.xml] 'collection' is
not an object type found in the policy file.
INFO:    [C:\pds4\geo\schema\BUG_BDRF_product.xml] Begin processing.
SKIP:    [C:\pds4\geo\schema\BUG_BDRF_product.xml] 'XML_Schema' is not an
object type found in the policy file.
INFO:    [C:\pds4\geo\schema\BUG_Document_Set.xml] Begin processing.
SKIP:    [C:\pds4\geo\schema\BUG_Document_Set.xml] 'XML_Schema' is not an
object type found in the policy file.
INFO:    [C:\pds4\geo\schema\Data_Dict_2010-04-22f.xml] Begin processing.
SKIP:    [C:\pds4\geo\schema\Data_Dict_2010-04-22f.xml] 'XML_Schema' is not
an object type found in the policy file.
INFO:    [C:\pds4\geo\schema\Data_Dict_commpds3_2010-04-22f.xml] Begin processing.
SKIP:    [C:\pds4\geo\schema\Data_Dict_commpds3_2010-04-22f.xml] 'XML_Schema'
is not an object type found in the policy file.
INFO:    [C:\pds4\geo\schema\Data_Types_2010-04-22f.xml] Begin processing.
SKIP:    [C:\pds4\geo\schema\Data_Types_2010-04-22f.xml] 'XML_Schema' is not an
object type found in the policy file.
INFO:    [C:\pds4\geo\schema\Product_XML_Schema.xml] Begin processing.
SKIP:    [C:\pds4\geo\schema\Product_XML_Schema.xml] 'XML_Schema' is not an
object type found in the policy file.
INFO:    [C:\pds4\geo\mars_analog_data\aref_235_450.xml] Begin processing.
SUCCESS:    [C:\pds4\geo\mars_analog_data\aref_235_450.xml] Succesfully registered
product: \
URN:NASA:PDS:BUGLAB-GB:BUGLAB-GB:MARS-ANALOG-SAMPLE-DATA:AREF_235_450::1.0
INFO:    [C:\pds4\geo\mars_analog_data\aref_235_480.xml] Begin processing.
SUCCESS:    [C:\pds4\geo\mars_analog_data\aref_235_480.xml] Succesfully registered
product: \
URN:NASA:PDS:BUGLAB-GB:BUGLAB-GB:MARS-ANALOG-SAMPLE-DATA:AREF_235_480::1.0
INFO:    [C:\pds4\geo\mars_analog_data\aref_235_530.xml] Begin processing.
SUCCESS:    [C:\pds4\geo\mars_analog_data\aref_235_530.xml] Succesfully registered
product: \
URN:NASA:PDS:BUGLAB-GB:BUGLAB-GB:MARS-ANALOG-SAMPLE-DATA:AREF_235_530::1.0
INFO:    [C:\pds4\geo\mars_analog_data\aref_235_600.xml] Begin processing.
SUCCESS:    [C:\pds4\geo\mars_analog_data\aref_235_600.xml] Succesfully registered
product: \
URN:NASA:PDS:BUGLAB-GB:BUGLAB-GB:MARS-ANALOG-SAMPLE-DATA:AREF_235_600::1.0
INFO:    [C:\pds4\geo\mars_analog_data\aref_235_670.xml] Begin processing.
SUCCESS:    [C:\pds4\geo\mars_analog_data\aref_235_670.xml] Succesfully registered
product: \
URN:NASA:PDS:BUGLAB-GB:BUGLAB-GB:MARS-ANALOG-SAMPLE-DATA:AREF_235_670::1.0
INFO:    [C:\pds4\geo\mars_analog_data\aref_235_750.xml] Begin processing.
SUCCESS:    [C:\pds4\geo\mars_analog_data\aref_235_750.xml] Succesfully registered
product: \
URN:NASA:PDS:BUGLAB-GB:BUGLAB-GB:MARS-ANALOG-SAMPLE-DATA:AREF_235_750::1.0
INFO:    [C:\pds4\geo\mars_analog_data\aref_235_800.xml] Begin processing.
SUCCESS:    [C:\pds4\geo\mars_analog_data\aref_235_800.xml] Succesfully registered
product: \
URN:NASA:PDS:BUGLAB-GB:BUGLAB-GB:MARS-ANALOG-SAMPLE-DATA:AREF_235_800::1.0
INFO:    [C:\pds4\geo\mars_analog_data\aref_235_860.xml] Begin processing.
SUCCESS:    [C:\pds4\geo\mars_analog_data\aref_235_860.xml] Succesfully registered
product: \
```

```
URN:NASA:PDS:BUGLAB-GB:BUGLAB-GB:MARS-ANALOG-SAMPLE-DATA:AREF_235_860::1.0
INFO:    [C:\pds4\geo\mars_analog_data\aref_235_900.xml] Begin processing.
SUCCESS:   [C:\pds4\geo\mars_analog_data\aref_235_900.xml] Succesfully registered
product: \
URN:NASA:PDS:BUGLAB-GB:BUGLAB-GB:MARS-ANALOG-SAMPLE-DATA:AREF_235_900::1.0
INFO:    [C:\pds4\geo\mars_analog_data\aref_235_930.xml] Begin processing.
SUCCESS:   [C:\pds4\geo\mars_analog_data\aref_235_930.xml] Succesfully registered
product: \
URN:NASA:PDS:BUGLAB-GB:BUGLAB-GB:MARS-ANALOG-SAMPLE-DATA:AREF_235_930::1.0
INFO:    [C:\pds4\geo\mars_analog_data\aref_235_990.xml] Begin processing.
SUCCESS:   [C:\pds4\geo\mars_analog_data\aref_235_990.xml] Succesfully registered
product: \
URN:NASA:PDS:BUGLAB-GB:BUGLAB-GB:MARS-ANALOG-SAMPLE-DATA:AREF_235_990::1.0
INFO:    [C:\pds4\geo\mars_analog_data\MAS_Data_Collection.xml] Begin processing.
SKIP:    [C:\pds4\geo\mars_analog_data\MAS_Data_Collection.xml] 'collection' is not
an object type found in the policy file.
INFO:    [C:\pds4\geo\geometry\BUGLAB_Geometry_Collection.xml] Begin processing.
SKIP:    [C:\pds4\geo\geometry\BUGLAB_Geometry_Collection.xml] 'collection' is not
an object type found in the policy file.
INFO:    [C:\pds4\geo\geometry\geominfo.xml] Begin processing.
SKIP:    [C:\pds4\geo\geometry\geominfo.xml] 'document_set' is not an object type
found in the policy file.
INFO:    [C:\pds4\geo\context\BUGLAB_Context_Collection.xml] Begin processing.
SKIP:    [C:\pds4\geo\context\BUGLAB_Context_Collection.xml] 'collection' is not an
object type found in the policy file.
INFO:    [C:\pds4\geo\context\bug_instrument.xml] Begin processing.
SKIP:    [C:\pds4\geo\context\bug_instrument.xml] 'document_set' is not an object
type found in the policy file.
INFO:    [C:\pds4\geo\context\bug_laboratory.xml] Begin processing.
SKIP:    [C:\pds4\geo\context\bug_laboratory.xml] 'document_set' is not an object
type found in the policy file.
INFO:    [C:\pds4\geo\context\bug_mars_data_set.xml] Begin processing.
SKIP:    [C:\pds4\geo\context\bug_mars_data_set.xml] 'document_set' is not an object
type found in the policy file.
INFO:    [C:\pds4\geo\about\aareadme.xml] Begin processing.
SKIP:    [C:\pds4\geo\about\aareadme.xml] 'document_set' is not an object type found
in the policy file.
INFO:    [C:\pds4\geo\about\BUGLAB_About_Collection.xml] Begin processing.
SKIP:    [C:\pds4\geo\about\BUGLAB_About_Collection.xml] 'collection' is not an object
type found in the policy file.

Summary:

11 of 30 files are candidate products, 19 skipped
11 of 11 candidate products registered.
0 of 0 associations registered.

End of Log
```

1.5 **Appendix A - UNIX Setup Options**

....

UNIX Setup Options

This section details a couple of options for setting up a UNIX environment for launching Harvest Tool.

Specify the CLASSPATH on the Command-Line

An alternative method to setting the *CLASSPATH* variable with all of the tool's dependent JAR files is to specify the *java.ext.dirs* Java property on the command-line when running the tool each time. This is done by passing the property via the Java "-D" flag as demonstrated in the following example:

```
% java -Djava.ext.dirs=$HOME/harvest-0.3.0/lib \
gov.nasa.pds.harvest.HarvestLauncher -Dpds.registry="<Registry URL>" \
<policy file> <command-line arguments>
```

Specify the JAR on the Command-Line

Another alternative method is to specify the executable JAR file on the command-line when running the tool each time. This is done by passing the JAR file specification via the Java "-jar" flag as demonstrated in the following example:

```
% java -jar $HOME/harvest-0.3.0/lib/harvest-0.3.0.jar \
-Dpds.registry="<Registry URL>" <policy file> <command-line arguments>
```

1.6 Appendix B - Windows Setup Options

Windows Setup Options

This section details a couple of options for setting up a Windows environment for launching Harvest Tool.

Specify the CLASSPATH on the Command-Line

An alternative method to setting the *CLASSPATH* variable with all of the tool's dependent JAR files is to specify the *java.ext.dirs* Java property on the command-line when running the tool each time. This is done by passing the property via the Java "-D" flag as demonstrated in the following example:

```
C:\> java -Djava.ext.dirs=c:\harvest-0.3.0\lib \
gov.nasa.pds.harvest.HarvestLauncher -Dpds.registry="<Registry URL>" \
<policy file> <command-line arguments>
```

Specify the JAR on the Command-Line

Another alternative method is to specify the executable JAR file on the command-line when running the tool each time. This is done by passing the JAR file specification via the Java "-jar" flag as demonstrated in the following example:

```
C:\> java -jar c:\harvest-0.3.0\lib\harvest-0.3.0.jar \
-Dpds.registry="<Registry URL>" <policy file> <command-line arguments>
```

Specify the Path in the Control Panel

The method for setting the executable path permanently for Harvest Tool is to set the *Path* environment variable via the control panel 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:
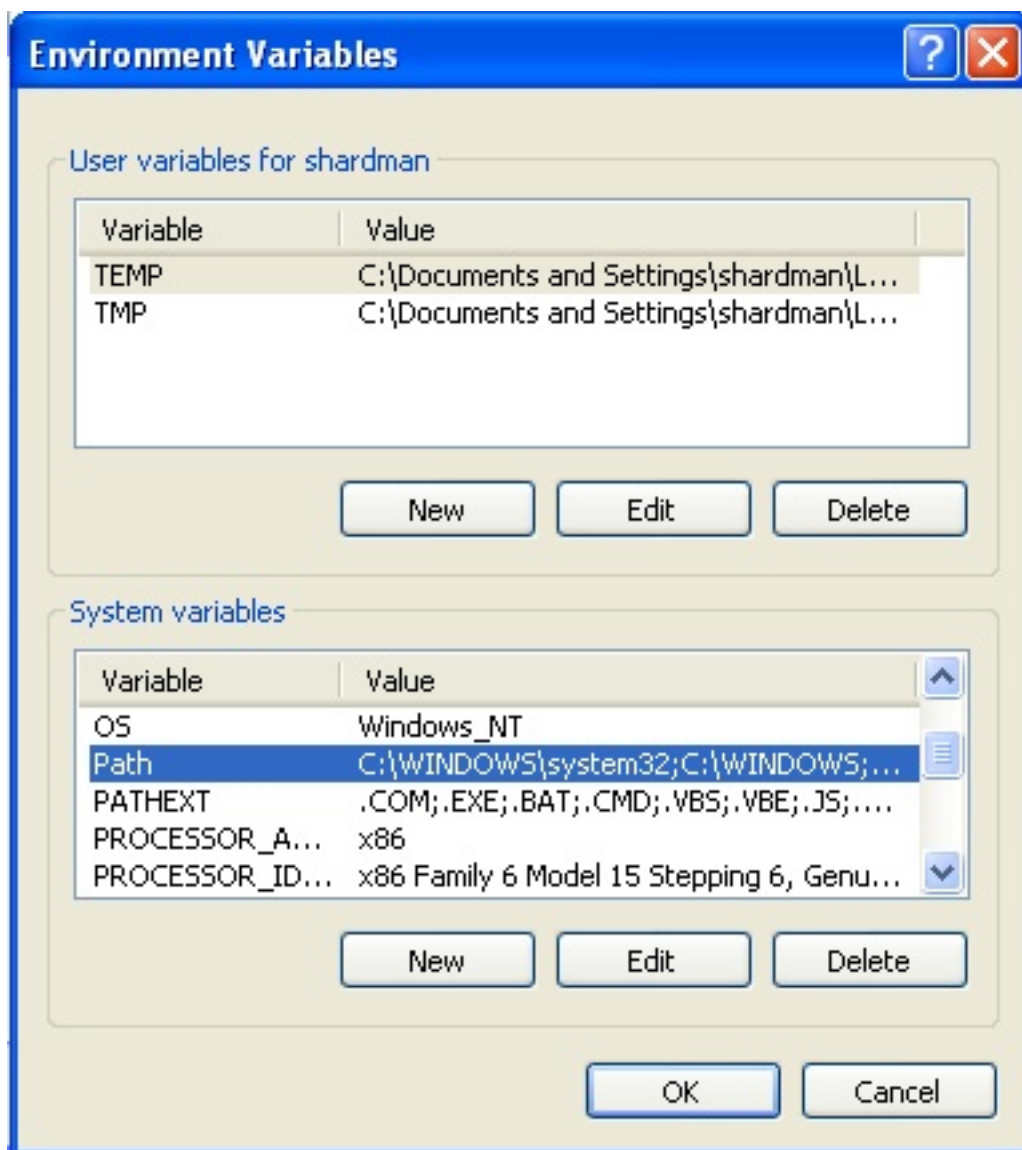
- Highlight the *Path* variable in the System Variables list and select the Edit button.

- 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 at 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 *Path* changes to take effect.

1.7 Appendix C - Harvest Policy Schema

....

Harvest Policy Schema

The XML schema file for validating policy files follows:

```xml
<?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"/>

<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"/>
```

```xml
      <xsd:attribute ref="uri" use="required"/>
      <xsd:attribute ref="default"/>
</xsd:complexType>


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


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


<xsd:element name="policy" type="Policy"/>
<xsd:complexType name="Policy">
   <xsd:sequence>
      <xsd:element ref="bundles" minOccurs="0"/>
      <xsd:element ref="collections" minOccurs="0"/>
      <xsd:element ref="directories" minOccurs="0"/>
      <xsd:element ref="validation" minOccurs="0"/>
      <xsd:element ref="candidates"/>
   </xsd:sequence>
</xsd:complexType>

</xsd:schema>
```