Validate Tool v.0.1.0

for the Planetary Data System

NASA Jet Propulsion Laboratory,
California Institute of Technology

03 November 2010

Table of Contents

1.1 Overview

About Validate Tool

The Validate Tool project contains software for validating PDS4 product labels and product data. The associated specific schema for the product label specifies syntactic and semantic constraints. The product label itself specifies the constraints for the data.

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 Validate 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 0.1.0

This release of the Validate 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:

- Support for validating a product label or set of product labels against a specified XML Schema. Validation capabilities provided by the Core Library.
- Support for validating against a set of packaged schemas, including determination of the appropriate schema based on the type of product label. Internal set of XML Schemas provided by the Core Library.
- Support for command-line input including specifying files or directories for validation target, specifying a regular expression for validation target, turn off directory recursion, specifying schemas for validation and specifying a file for the validation report.

The liens for this release are as follows:

- Need support for specifying a bundle or collection product and have the tool validate the products that are referenced by those products.

1.3 Installation

Installation

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

- System Requirements
- Unpacking the Package

System Requirements

The Core 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 *validate* package from the yet to be developed PDS 2010 web page. 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 validate-0.1.0-bin.zip
or
% tar -xzvf validate-0.1.0-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 *validate-0.1.0* 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 Validate 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.

- **lib/**

  This directory contains the dependent jar files for the tool along with the executable jar file (validate-0.1.0.jar) containing the Validate Tool software.

1.4 Operation

Operation

The following topics can be found in this section:

- 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 the Validate 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, *Validate*, 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/validate-0.1.0/bin
```

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

```
% Validate <targets> <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, *Validate.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:\validate-0.1.0\bin
```

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

```
C:\> Validate <targets> <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 tell the Validate Tool to use core schemas from some external directory rather than the core schemas loaded internally into the tool. Users that wish to not do this can skip this section.

The Java system property, *core.schema.dir*, can be used to load core schemas from an external directory. At this time, any external directory used must have a sub-directory named *0111c* with the schemas under this sub-directory. For example, if specifying an external directory named */home/pds/schemas*, the directory must have the following structure:

- **0111c/**

  This directory will contain the list of core schemas to load into the tool. This overwrites the internal core schemas used for validation.

The sections below detail how to add this system property into the Validate Tool launch scripts.

UNIX-Based Users

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

```
java -Xss256m -jar ${VALIDATE_JAR} "$@"
```

Add the *core.schema.dir* Java system property using the *-D* Java flag option and set it to the location of the schemas. For example, making the following change to the launch script allows the Validate Tool to load core schemas from a directory named */home/pds/schemas*:

```
java -Dcore.schema.dir="/home/pds/schemas" -Xss256m -jar ${VALIDATE_JAR} "$@"
```

Windows-Based Users

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

```
java -Xss128m -jar "%VALIDATE_JAR%" $*
```

Add the *core.schema.dir* Java system property using the *-D* Java flag option and set it to the location of the schemas. For example, making the following change to the launch script allows the Validate Tool to load core schemas from a directory named *c:\pds\schemas*:

```
java -Dcore.schema.dir="c:\pds\schemas" -Xss128m -jar "%VALIDATE_JAR%" $*
```

Tool Execution

The Validate 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
-t, --target <files,directories>	Explicitly specify the targets (product files, directories) to validate. Targets can be specified implicitly as well (example: Validate product.xml). For more details on target specification, see the Specifying Targets section.
-x, --schema <schemas>	Specify schema files to use during validation. By default, PDS schemas are built into the tool to perform the validation.
-r, --report-file <file>	Specify the report file name. Default is to output results to standard out.
-e, --regexp <file patterns>	Specify file patterns to look for when validating a target directory. Each pattern must be surrounded in quotes (example: "*.xml"). Pattern matching is case-insensitive in Windows, but case-sensitive for other systems.
-L, --local	Validate files only in the target directory instead of recursively traversing down the sub-directories.
-V, --version	Display the release number and copyright information.
-h, --help	Display Harvest usage.

Running the Validate Tool

This section demonstrates some of the ways that the tool can be executed using the command-line option flags:

- Validating a Target File
- Validataing a Target Directory
- Validating Against User-Specified Schemas
- Validating Specific Files in a Target Directory
- Ignoring Sub-Directories During Validation
- Changing Tool Behaviors With The Configuration File

Validating a Target File

The following command demonstrates the validation of a single data product label against the core PDS schemas:

```
% Validate product.xml
```

Validating a Target Directory

The following command demonstrates the validation of a target directory against the core PDS schemas:

```
% Validate /home/pds/collection
```

Validating Against User-Specified Schemas

Specifying schemas on the command line will allow the Validate Tool to validate against the user-specified schemas instead of the schemas that come with the tool. The following command demonstrates the validation of a single product label against a user-specified schema:

```
% Validate proudct.xml -x product.xsd
```

The following command demonstrates the validation of a set of target files against a set of user-specified schemas:

```
% Validate producta.xml, productb.xml -x producta.xsd, productb.xsd
```

Validating Specific Files in a Target Directory

The following command demonstrates the validation of any file that has a *.xml* extension in a target directory:

```
% Validate /home/pds/collection -e "*.xml"
```

Note: File patterns should be surrounded in quotes to avoid having the system shell mistakingly interpreting them. In addition, pattern matching is case-insensitve in Windows, but case-sensitive for other systems.

Ignoring Sub-Directories During Validation

By default, the Validate Tool will recursively traverse a target directory during validation. The *local* flag option is used to tell the Validate Tool to not perform recursion. The following command demonstrates the validation of a target directory without directory recursion:

```
% Validate /home/pds/collection -L
```

Changing Tool Behaviors With The Configuration File

A configuration file can be passed into the command-line to change the default behaviors of the tool and to also provide users a way to perform validation with a single flag. For more details on how to setup the configuration file, see the Using a Configuration File section.

The following command demonstrates performing validation using a configuration file:

```
% Validate -c config.txt
```

Specifying Targets

Targets are validated in the order in which they are specified on the command-line. They can be specified implicitly and explicitly.

To specify targets implicitly, it is best to specify them first on the command-line before any other command-line option flags. The following command demonstrates the validation of an implicitly defined, single target product label:

```
% Validate product.xml
```

The following command demonstrates the validation of implicitly defined, multiple targets:

```
% Validate product.xml, /home/pds/collection
```

Implicit targets should not be specified after option flags that allow multiple arguments (see example below). Unexpected results can occur.

```
% Validate -x product.xsd product.xml
```

In this example, the Validate Tool will inadvertently treat the implicit target, *product.xml*, as a schema file.

Targets can be specified both implicitly and explicitly at the same time. Targets specified implicitly are validated first, followed by those that are specified explicitly with the target flag.

The following command demonstrates the validation of multiple product labels, specified both implicitly and explicitly:

```
% Validate producta.xml, productb.xml -t productc.xml, /home/pds/collection
```

In this example, *producta.xml* and *productb.xml* will get validated first, then *productc.xml* and the product labels in */home/pds/collection* will get validated next.

Using a Configuration File

A configuration file is an alternative way to set the different behaviors of the tool instead of the command-line option flags. It consists of a text file made up of keyword/value pairs. The configuration file follows the syntax of the stream parsed by the Java Properties.load(java.io.InputStream) method.

Some of the important syntax rules are as follows:

- Blank lines and lines which begin with the hash character "#" are ignored.
- Values may be separated on different lines if a backslash is placed at the end of the line that continues below.
- Escape sequences for special characters like a line feed, a tabulation or a unicode character, are allowed in the values and are specified in the same notation as those used in Java strings (e.g. \n, \t, \r).

Since backslashes (\) have special meanings in a configuration file, keyword values that contain this character will not be interpreted properly by the Validate Tool even if it is surrounded by quotes. A common example would be a Windows path name (e.g. c:\pds\collection). Use the forward slash character instead (c:/pds/collection) or escape the backslash character (c:\\pds\\collection).

Note: Any flag specified on the command-line takes precedence over any equivalent settings placed in the configuration file.

The following table contains valid keywords that can be specified in the configuration file:

Property Keyword	Associated Command-Line Option
validate.target	-t, --target
validate.schema	-x, --xsd
validate.report	-r, --report-file
validate.regexp	-e, --regexp
validate.local	-L, --local

The following example demonstrates how to set a configuration file:

```
# This is a Validate Tool configuration file
```

```
validate.target = ./collection
validate.report = report.txt
validate.regexp = "*.xml"
```

This is equivalent to running the tool with the following flags:

```
-t ./collection -e "*.xml" -r report.txt
```

The following example demonstrates how to set a configuration file with multiple values for a keyword:

```
# This is a Validate Tool configuration file with multiple values

validate.target = product.xml, ./collection
vtool.regexp = "*.xml", "Mars*"
```

This is equivalent to running the tool with the following flags:

```
-t product.xml, ./collection -e "*.xml", "Mars*"
```

The following example demonstrates how to set a configuration file with multiple values that span across multiple lines:

```
# This is a Validate configuration file with multiple values
# that span across multiple lines

validate.target = product.xml, ./collection
validate.regexp = "*.xml", "Mars*"
```

As previously mentioned, any flag options set on the command-line will overwrite settings set in the configuration file. The following example demonstrates how to override a setting in the configuration file.

Suppose the configuration file named *config.txt* is defined as follows:

```
```

```
validate.target = ./collection
validate.regexp = "*.xml"
```

This configuration allows the tool to validate files with a *.xml* extension in the *collection* directory. To change the behavior to validate all files instead of just files ending in *.xml*, then specify the *regexp* flag option on the command-line to overwrite the *validate.regexp* property:

```
% Validate -c config.txt -e "*"
```

Report Format

This section describes the contents of the Validate Tool report. At this time, the Validate Tool outputs a series of log messages. A log message consists of the severity level, file name, line number, and a message. The following is an example of some of the log messages that can be expected from the Validate Tool:

```
PDS Validate Tool Report

Configuration:
  Version                  0.1.0
  Time                     Thu, Oct 28 2010 at 10:05:03 AM
  Core Schemas             [a.xsd, b.xsd, c.xsd]

Parameters:
  Target(s)                [C:\pds4]
  User-Specified Schemas   [C:\pds4\element-definitions\schemas\ \
                           Product_Data_Element_Definition_2010-04-22B.xsd]
  Severity Level           Warnings
  Recurse Directories      true

Validation Details:

  ERROR  [file:/C:/pds4/coordinate_system_positive_azimuth_direction_0111c.xml] \
  line 8, 66: cvc-elt.5.2.2.2.2: The value 'Product_Element_Definition' of \
  element 'product_class' does not match the {value constraint} value \
  'Product_Data_Element_Definition'.
  ERROR  [file:/C:/pds4/coordinate_system_positive_azimuth_direction_0111c.xml] \
  line 24, 34: cvc-complex-type.2.4.a: Invalid content was found starting with \
  element 'Reference_Entry_Generic'. One of \
  '{"http://pds.nasa.gov/schema/pds4/pds":data_reference, \
  "http://pds.nasa.gov/schema/pds4/pds":Bibliographic_Reference, \
  "http://pds.nasa.gov/schema/pds4/pds":Observing_System}' is expected.
  ERROR  [file:/C:/pds4/coordinate_system_positive_azimuth_direction_0111c.xml] \
  line 29, 25: cvc-complex-type.2.4.a: Invalid content was found starting with \
  element 'Element_Definition'. One of \
  '{"http://pds.nasa.gov/schema/pds4/pds":File_Area}' is expected.
```

```
  ERROR  [file:/C:/pds4/test/ \
0001_NASA_PDS_1_img_Az_el_coordinate_system_reference_coordinate_system_name_0111c.xml]
\
  line 5, 26: cvc-complex-type.2.4.a: Invalid content was found starting with \
  element 'Identification_Area'. One of \
  '{"http://pds.nasa.gov/schema/pds4/pds":Product_Identification_Area}' is expected.


End of Report
```

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 the Validate 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/validate-0.1.0/lib \
gov.nasa.pds.validate.ValidateLauncher \
<targets> <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/validate-0.1.0/lib/validate-0.1.0.jar \
<targets> <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 the Validate 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:\validate-0.1.0\lib \
gov.nasa.pds.validate.ValidateLauncher \
<targets> <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:

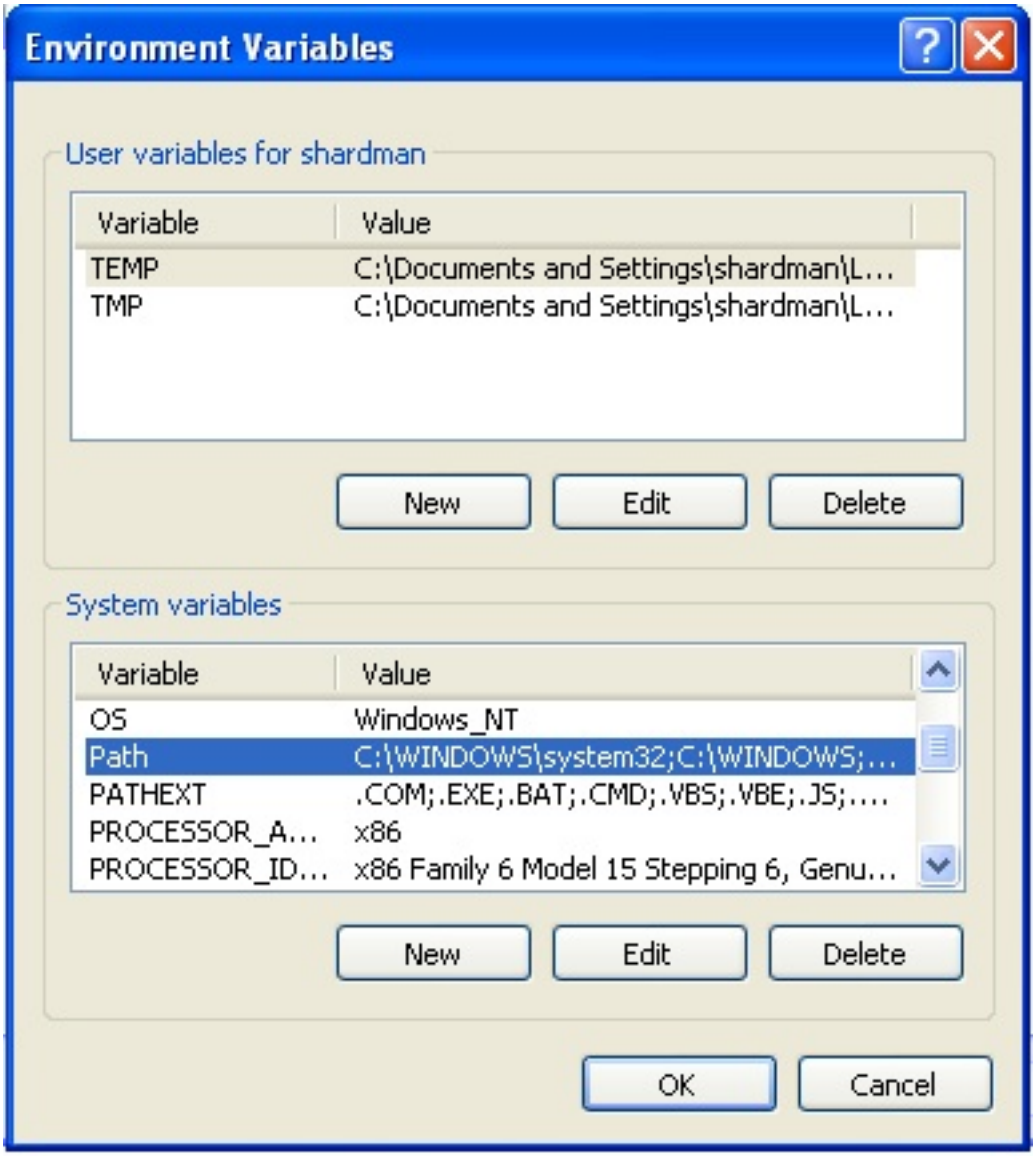
```
C:\> java -jar c:\validate-0.1.0\lib\validate-0.1.0.jar \
<targets> <command-line arguments>
```

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 *validate* 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.