Generate Tool v.0.2.0

for the Planetary Data System

NASA Jet Propulsion Laboratory,
California Institute of Technology

10 November 2011

Table of Contents

1.1 Overview

About Generate Tool

The Generate Tool provides a command-line interface for generating PDS4 Labels from either a PDS3 Label or a PDS-specific DOM object.

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 Generate 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 0.2.0

This release of the Generate 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. This release is a roll up of the 0.1.1 intermediate release and includes some documentation updates.

Release 0.1.0

This release of the Generate 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. This release includes initial capabilities for generating PDS4 product labels from PDS3 product labels.

1.3 Installation

Installation

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

- System Requirements
- Unpacking the Package
- Configuring the Environment

System Requirements

The Generate 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 *generate* 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 generate-0.2.0-bin.zip
or
% tar -xzvf generate-0.2.0-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 *generate-0.2.0* 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.

- **conf/**

  This directory contains configuration files for the tool.

- **doc/**

  This document directory contains a local web site with the Generate 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 example files for the tool.

- **lib/**

  This directory contains the dependent jar files for the tool along with the executable jar file (generate-0.2.0.jar) containing the Generate Tool software.

Configuring the Environment

In order to execute the Generate 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}/generate-0.2.0/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:\generate-0.2.0\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

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

Generate 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
-c, --conf-home	Specify the path for the configuration files
-d, --debug	Directs output to screen, not file.
-f, --file-list	Specify the path for a file containing a list of file paths for PDS3 Labels
-h, --help	Display usage.
-o, --output-file	Specify an output filename. Default is PDS3 label name with _pds4 suffix.
-p, --pds3-label	Specify the file path for the PDS3 Label to be converted to PDS4
-t, --template	Specify the file path for the Velocity template for PDS3 -> PDS4 conversion
-V, --version	Display application version.

Execute Generate 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 following command demonstrates how to run the Generate Tool to convert from a PDS3 Label to a PDS4 Label using a Velocity template file and output to standard out:

```
% PDS4Generate -d -p <pds3-label-path> -t <velocity-template-path>
```

The following command demonstrates how to run the Generate Tool to convert from a PDS3 Label to a PDS4 Label using a Velocity template file and output to a file:

```
% PDS4Generate -p <pds3-label-path> -t <velocity-template-path>
```

The output would go into a file <pds3-label-path>.xml (i.e. $HOME/mpf123.img -> $HOME/mpf123.xml

1.5 Template Guide

Template Guide

The Generate Tool leverages Apache Velocity Templates to provide a robust, generic method of creating PDS4 Labels. The labels can be generated from an existing PDS3 label (attached or detached) or some other defined data object. This section contains the various scenarios that the base version of the tool handles. Scenarios, in this context, can be defined as the specific types of problems a user may come across when attempting to develop a PDS4 label. These will be extended as the the data objects and PDS4 standard become more refined.

This section contains the following information:

Scenario Format

Each scenario described below will follow a format in order to provide a tutorial-like experience when developing a PDS4 label. The progression through a scenario will include a problem description, PDS3 label input (if applicable), the desired PDS4 output, and the Velocity Template solution.

PDS3 (if applicable) (pds3_example.lbl)

```
PDS3 Label Input
```

PDS4 (pds4_example.xml)

```
PDS4 XML Output
```

Velocity (template_example.vm)

```
Velocity Template Solution
```

Scenarios

The scenarios provided are some of the problems elicited from translating from data objects, specifically PDS3 standard, into PDS4. They are by no means all inclusive. These solutions are NOT PDS4-compliant, they are simply proof-of-concept examples to prove the functionality of the tool and how they can be applied to PDS4-compliant labels.

- Hard-Coded Value
- Base Element
- Sub-Element
- Multiple Instance
- Same Class-Different Value
- Units
- Generated Value

Hard-Coded Value

Allows the user to specify default values for a tag.

PDS4

```
<Product_Array_2D_Image>
  <Data_Standards>
    <dd_version_id>0311B_20110709</dd_version_id>
  </Data_Standards>
</Product_Array_2D_Image>
```

Velocity

```
<Product_Array_2D_Image>
  <Data_Standards>
    <dd_version_id>0311B_20110709</dd_version_id>
  </Data_Standards>
</Product_Array_2D_Image>
```

Base Element

Provides a mechanism for translating base elements from PDS3 to PDS4. A base element in a PDS3 label is a KEYWORD-VALUE that is not a part of an OBJECT or GROUP. Often seen flush up against the left side of the label file.

PDS3

```
TARGET_NAME         = "DEIMOS"
```

PDS4

```
<Subject_Area>
  <target_name>DEIMOS</target_name>
</Subject_Area>
```

Velocity

```
<Subject_Area>
  <target_name>$label.TARGET_NAME</target_name>
</Subject_Area>
```

Sub-Element

Provides a mechanism for translating sub-elements from PDS3 to PDS4. A sub-element in a PDS3 label is a KEYWORD-VALUE that is included within an OBJECT or GROUP.

PDS3

```
OBJECT                          = IMAGE
        MEAN                  = 8.6319
        MEDIAN                  = 8
        MINIMUM                  = 8
END_OBJECT                = IMAGE
```

PDS4

```
<Object_Statistics>
  <mean>8.6319</maximum>
  <median>8</mean>
  <minimum>8</median>
</Object_Statistics>
```

Velocity

```
<Object_Statistics>
  <mean>$label.IMAGE.MEAN</mean>
  <median>$label.IMAGE.MEDIAN</median>
  <minimum>$label.IMAGE.MINIMUM</minimum>
```

```
</Object_Statistics>
```

Note: For keywords within nested OBJECTs/GROUPs, simply continue the dot notation, i.e. $label.OBJECT1.OBJECT2.KEYWORD

Multiple Instance

Provides a mechanism for translating PDS3 Keywords with an array of values into multiple instances of a PDS4 class.

PDS3

```
GROUP                          = BAND_BIN
  BANDS                          = 4
  BAND_BIN_UNIT                  = MICROMETER
  CENTER                 = (0.374, 0.384, 0.394, 0.404)
  WIDTH                          = (0.0155, 0.0115, 0.0114, 0.0112)
END_GROUP                 = BAND_BIN
```

PDS4

```
<Band_Bin_Set>
  <Band_Bin>
    <center>0.374</center>
    <width>0.0155</width>
  </Band_Bin>
  <Band_Bin>
    <center>0.384</center>
    <width>0.0115</width>
  </Band_Bin>
  <Band_Bin>
    <center>0.394</center>
    <width>0.0114</width>
  </Band_Bin>
  <Band_Bin>
    <center>0.404</center>
    <width>0.0112</width>
  </Band_Bin>
</Band_Bin_Set>
```

Velocity

```
<Band_Bin_Set>
#set( $bandBinList = $label.getRecords('BAND_BIN.CENTER','BAND_BIN.WIDTH') )
#foreach ( $bandBin in $bandBinList )
  <Band_Bin>
    <center>$bandBin.CENTER</center>
```

```
    <width>$bandBin.WIDTH</width>
  </Band_Bin>
#end
</Band_Bin_Set>
```

Same Class-Different Value

Provides a mechanism for translating multiple KEYWORD-VALUE combinations into multiple instances of the same class.

PDS3

```
OBJECT                                = IMAGE
  INTERCHANGE_FORMAT                 = BINARY
  LINES                                 = 192
  LINE_SAMPLES                          = 320
END_OBJECT                           = IMAGE
```

PDS4

```
<Array_Axis>
  <name>SAMPLES</name>
  <elements>320</elements>
  <sequence_number>1</sequence_number>
</Array_Axis>
<Array_Axis>
  <name>LINES</name>
  <elements>192</elements>
  <sequence_number>2</sequence_number>
</Array_Axis>
```

Velocity

```
<Array_Axis>
  <name>SAMPLES</name>
  <elements>$label.IMAGE.LINES_SAMPLES</elements>
  <sequence_number>1</sequence_number>
</Array_Axis>
<Array_Axis>
  <name>LINES</name>
  <elements>$label.IMAGE.LINES</elements>
  <sequence_number>2</sequence_number>
</Array_Axis>
```

Units

Provides a mechanism for translating a PDS3 KEYWORD-VALUE along with its units into PDS4 standard.

PDS3

```
INST_AZIMUTH                = 114.0210 <deg>
```

PDS4

```
<Geometry_Parameters>
  <azimuth units="deg">114.0210</azimuth>
</Geometry_Parameters>
```

Velocity

```
<Geometry_Parameters>
  <azimuth units="$label.getUnits('INST_AZIMUTH')">$label.INST_AZIMUTH</azimuth>
</Geometry_Parameters>
```

Generated Value

Provides a mechanism for generating values of keywords from known algorithms. This aspect of the software can be extended, if necessary.

PDS4

```
<File_Area>
  <md5_checksum>2a6f0be7f63d0aa032457f1f29d3e51d</md5_checksum>
</File_Area>
```

Velocity

```
<File_Area>
  <md5_checksum>$generate.md5_checksum</md5_checksum>
</File_Area>
```

Note: The currently available generated values include:

- md5_checksum
- file_name

- file_size

Examples

The following examples are NOT finished products or follow PDS4 standards. They are merely proof-of-concept.

- To follow along with solutions above:
  - pds3_example.lbl
  - pds4_example.xml
  - template_example.vm
- Beta MPF example:
  - i985135l.img
  - MPF_IMP_EDR7.vm

Common Errors

Object Not Found

```
<generated-node>Object Not Found</generated-node>
```

When a node is populated with the text "Object Not Found", it means that the keyword specified was not found. This error applies to generated values and often means that either the mapping was not specified in the generated-mappings.xml, or the class was never created to generate that value

Advanced Users

See the Velocity User Guide for more detailed documentation on leveraging the Java objects represented by the variables noted above (label, bandBinList, etc.).

References

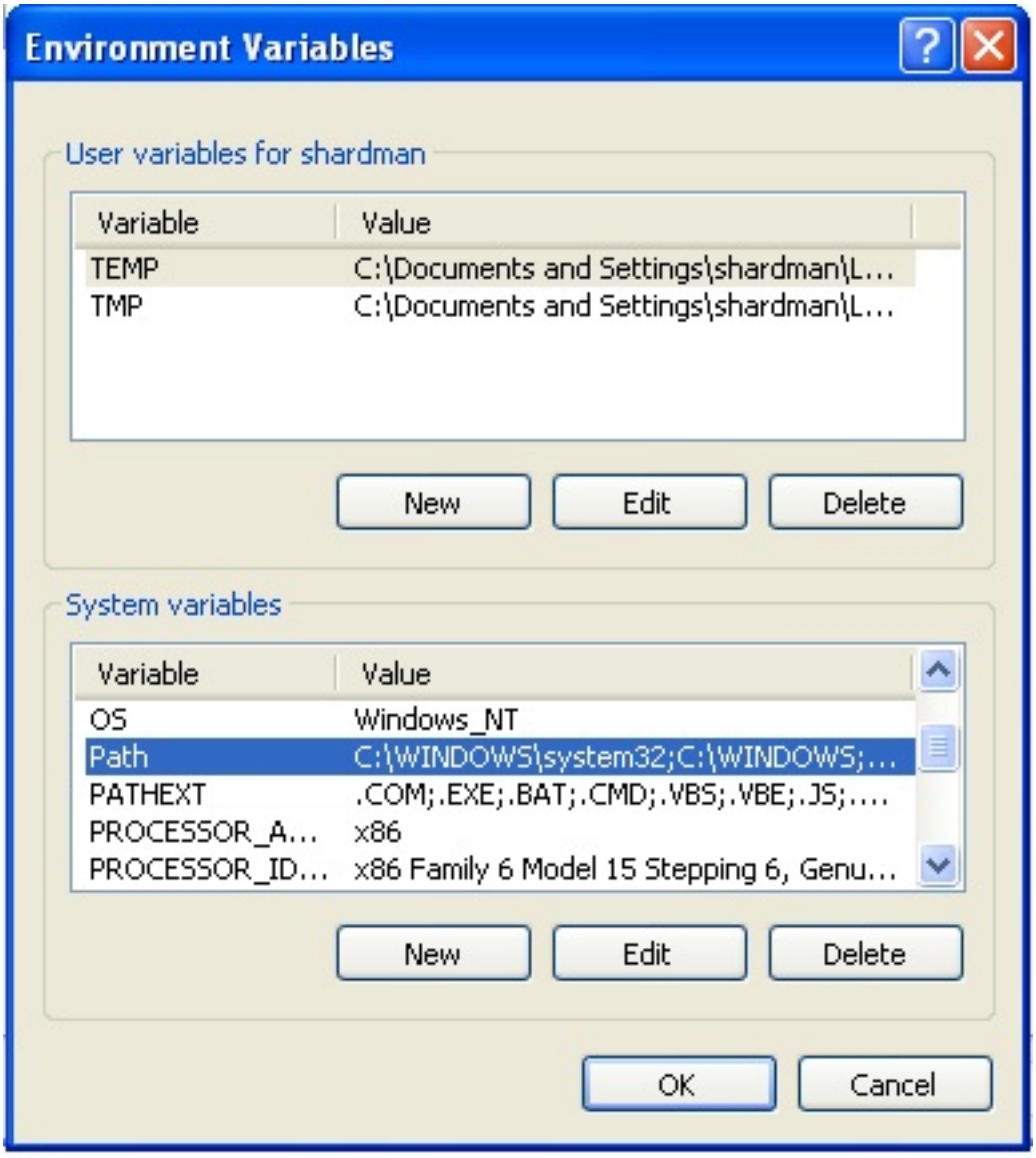
- Velocity User Guide
- IN F2F Slides - 8/24/11

1.6 Appendix - Windows System Properties

Windows System Properties

The required environment variables for the *generate* 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:



- 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 *generate* 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.