Validation Tool v.2.2.0

for the Planetary Data System

NASA Jet Propulsion Laboratory,
California Institute of Technology

08 February 2011

Table of Contents

1.1 Overview

About Validation Tool

The Validation Tool (VTool) is intended for validating PDS product labels according to the PDS Standards and the Planetary Science Data Dictionary (PSDD).

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 Validation Tool release including any impact that the new or modified capabilities will have on the Discipline Nodes or the PDS user community. A somewhat itemized list of changes for each release can be found on the Release Changes page. If viewing this document in PDF form, the release change details are not available.

Release 2.2.0

This is a maintenance release of the Validation Tool integrated with release 3.1.0 of the Product Tools Library.

- Corrected an issue dealing with how date/time values are validated.
- Corrected a couple of issues dealing with malformed messages.
- Corrected an issue involving value manipulation when matching values.
- A number of issues were corrected in the Product Tools Library that effect interfaces utilized by other tools.

The liens for this release associated with version 3.1.0 of the Product Tools Library are as follows:

- Plural forms of valid unit names are not supported.
- Full checking of file characteristics not supported as of yet.

Release 2.1.0

This release of the Validation Tool addresses issues found during beta test for release 2.1.0-beta. These include:

- Files that do not contain a PDS_VERSION_ID are reported as warnings instead of errors.
- Truncate long values when error messages get thrown for exceeding the acceptable maximum length of an element definition.
- Corrected the issue where memory overflows occurred with large attached labels.
- Corrected to throw an error message when referenced files for non include pointers are not found.
- Corrected dictionary handling to support SPECIFIC_GROUPs and MAXIMUM values of UNK.
- Corrected to report errors found in the supplied dictionary.

- Corrected handling of pointer statements that contain multiple file references.
- Corrected handling of required/optional nested objects with descriptor names.
- Corrected an issue calculating the start byte of the data in attached labels.
- Corrected to capture errors in nested fragment labels.
- Cleaned up some messages.

The liens for this release associated with version 3.0.2 of the Product Tools Library are as follows:

- Plural forms of valid unit names are not supported.
- Full checking of file characteristics not supported as of yet.

Release 2.1.0-beta

This beta release of the Validation Tool software is integrated with release 3.0.1 of the Product Tools Library. In general, the Validation Tool should exhibit the same behavior as the previous release with the exception that the reports will be different. The messaging infrastructure in the Product Tools Library was overhauled resulting in improved messages. Although an effort was made to make the Validation Tool reports consistent with the previous release, there will be some differences. Other changes for this release include:

- Report line lengths greater than 80 characters (78 + <CR><LF>).
- Report when lines do not end in <CR><LF>.
- Combine attribute lists for identical elements when two dictionaries are merged.
- Recognize the END statement in attached labels.

The liens for this release associated with version 3.0.1 of the Product Tools Library are as follows:

- Memory problem with very large attached labels.
- Plural forms of valid unit names are not supported.
- Checking of file characteristics not supported as of yet.

Release 2.0.1

The operational release of the Validation Tool software incorporating the Phase II capability supporting validation of catalog files. This release includes some minor modifications regarding how catalog pointers are handled based on beta testing results.

Release 2.0.0

This release of the Validation Tool software incorporates the Phase II capability supporting validation of catalog files and was released to the Technical Staff for beta testing.

The major change for this release is the addition of support for validating catalog files, either standalone or as a set within a CATALOG directory. Modifications to support this capability included a general change with

regard to how pointers are handled as well as relaxing some of the PDS Label constraints where catalog files do not conform to those constraints. The result is that catalog files are now validated correctly and the spurious messages that were previously produced have been eliminated.

Other changes in this release include:

- Correct validation of quoted date/time strings. Date/time strings may not be quoted when associated with an element that is defined with one of the DATE data types.
- Fixed a spacing issue seen in the validation report on the Mac platform.

1.3 Installation

Installation

This section describes how to install the Validation Tool (VTool) contained in the *vtool* package. The following topics can be found in this section:

- System Requirements
- Unpacking the Package

System Requirements

The following sub-sections detail the system requirements for the tool.

Java Requirement

The Validation 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.

Data Dictionary Requirement

Release *1r64* or later of the Planetary Science Data Dictionary (PSDD) is required for the tools to function properly. Release *1r66* of the PSDD supports the validation of explicit FILE objects. The latest version of the PDS data dictionary can be retrieved from the PDS Data Dictionary web page.

Unpacking the Package

Download the *vtool* package from the Label Validation Tool 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 vtool-2.2.0-bin.zip
```

```
or
% tar -xzvf vtool-2.2.0-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 *vtool-2.2.0* 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.

- **doc/**

  This document directory contains a local web site with the Validation 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 (vtool-2.2.0.jar) containing the Validation Tool software.

1.4 Operation

Operation

The goal of the Validation Tool (VTool) is to programmatically ascertain if a given data product is PDS compliant (or "valid"). Typically, this means the data product is well formed, complete, syntactically and semantically correct, and that it conforms to all applicable PDS standards. The standards themselves are defined in the PDS Standards Reference .

This section describes how to use VTool in order to perform automated validation of data products. The following topics can be found in this section:

- Tool Setup
- Tool Execution
- Report Formats
- Exit Status Values

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 VTool, 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, *VTool*, 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/vtool-2.2.0/bin
```

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

```
% VTool <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, *VTool.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:\vtool-2.2.0\bin
```

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

```
C:\> VTool <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.

Tool Execution

VTool can be executed in various ways. This section describes how to run VTool and how to generate reports, as well as the behaviors and caveats of the tool.

Command-Line Options

The following table contains command-line options available to VTool:

Command-Line Option	Description
-t, --target <labels,URLs,dirs>	Explicitly specify the targets (label files, directories, and URLs) to validate. Targets can be specified implicitly as well (example: VTool label.lbl). For more details on target specification, see the Specifying Targets section.
-d, --dict <.full files>	Specify the Planetary Science Data Dictionary full file name and any local dictionaries.
-l, --log-file <file (optional)>	(Not available in this release) Specify the file name for the machine-readable log. A file specification is optional. If no file name is given, then the log will be written to standard out. For more details on writing the log to a file, see the Report and Log Generation section.
-r, --report-file <file>	Specify the file name for the human-readable report. Default is to write to the standard out if this option is not specified. This report, however, will not print to standard out if this option is missing AND the log file option is specified with no file name. For more details on how to create reports, see the Report and Log Generation section
-s, --report-style	Specify the standard human-readable report format. Valid values are "full" for a full view, "min" for a minimal view, or "sum" for a summary view. Default is to generate a full report if this option is not specified. for more details on these report styles, see the Report Formats section
-v, --verbose	Specify the message severity level and above to include in the human-readable report (1=Info, 2=Warning, 3=Error). Default is warning and above (level 2).
-I, --include <paths>	Specify paths to search for files referenced by pointers in a label. Separate each path with a comma. Default is to always look in the directory of the label, then search the specified directories.
-F, --no-follow	Do not follow or check for the existence of files referenced by pointer statements in a label.
-f, --force	Enable standalone label fragment validation. Default is to not validate them.
-c, --config <file>	Specify a configuration file to set the default values for VTool.
-L, --local	Validate files only in the input directory rather than recursively traversing down the sub-directories.
-e, --regexp <expressions>	Specify file patterns to look for when validating a directory. Separate each pattern with a comma. Each pattern should be surrounded in quotes ("*.LBL", "*.FMT") to avoid having the system shell mistakingly interpreting them (This has been seen on non-Windows systems). Pattern matching is case-insensitive in Windows, but case-sensitive for other systems.
-X, --ignore-file <expressions>	Specify file patterns to ignore when validating a directory. Separate each pattern with a comma. Patterns should be surrounded in quotes ("*.TXT", "*.TAB") to avoid having the system shell mistakingly interpreting them (This has been seen on non-Windows systems). Pattern matching is case-insensitive in Windows, but case-sensitive for other systems.
-D, --ignore-dir <expressions>	Specify the directory patterns to ignore. Separate each pattern with a space. Patterns should be surrounded by quotes ("LABEL", "DOC*") to avoid having the system shell mistakingly interpreting them (This has been seen on non-Windows systems). Pattern matching is case-insensitive in Windows, but case-sensitive for other systems.

Command-Line Option	Description
-p, --progress	Enable validation progress reporting. Default is no. When this option is enabled, the current directory being validated will be written to standard error, followed by a series of asterisk "*" symbols, which represents a file being validated.
-h, --help	Display VTool usage.
-V, --version	Display VTool version.

Execute VTool

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.

- Validating Against a Single Dictionary
- Validating Against Multiple Dictionaries
- Validating Files with a Specific File Pattern
- Ignoring Sub-Directories During Validation
- Ignoring Files with a Specific File Pattern
- Checking For Referenced Files in Different Locations
- Not Following a Pointer
- Standalone Label Fragment Validation
- Progress Reporting
- Changing Tool Behaviors With The Configuration File

Validating Against a Single Dictionary

The following command demonstrates the validation of a single data product label against the PSDD:

```
% VTool LABEL.LBL -d pdsdd.full
```

If there is a dictionary error, the tool will not perform label validation.

Validating Against Multiple Dictionaries

The following command demonstrates the validation of a single data product label against the PSDD and a local dictionary:

```
% VTool LABEL.LBL -d pdsdd.full, localdd.full
```

For more information on how the tool behaves when multiple dictionaries are passed in, see the Multiple Dictionary Support section.

Validating Files with a Specific File Pattern

The following command demonstrates the validation of multiple data product labels in a specified target directory, where validation only occurs on file names that end in ".LBL" or begins with the letters "MER":

```
% VTool $HOME/DIR -d pdsdd.full -e "*.LBL", "MER*"
```

Ignoring Sub-Directories During Validation

By default, the tool will recursively traverse down a directory tree when a target directory is specified, validating files within the sub-directories. Sub-directories can be ignored with the ignore directories option.

The following command demonstrates the validation of multiple data product labels in a specified target directory, where directories named "EXTRAS" or "LABEL" are ignored.

```
% VTool $HOME/DIR -d pdsdd.full -D "EXTRAS", "LABEL"
```

The local option can also be used to stop the tool from recursing down a directory tree.

The following command demonstrates the same validation as above, but without recursion:

```
% VTool $HOME/DIR -d pdsdd.full -L
```

Ignoring Files with a Specific File Pattern

The ignore files option can be used to tell the tool which files to ignore during validation.

The following command demonstrates the validation of multiple data product labels contained in a directory except for files ending in a ".IMG" or ".TAB":

```
% VTool $HOME/DIR -d pdsdd.full -X "*IMG", "*TAB"
```

Checking For Referenced Files in Different Locations

If a data product label contains a pointer statement that references a file, it will always assume it is co-located with the label. If it cannot be found there, then VTool will look for that referenced file in the paths specified by the include directories option.

The following command demonstrates the validation of a data product label that contains a pointer statement to a file located in a directory called DIR.

```
% VTool LABELPTR.LBL -d pdsdd.full -I $HOME/path
```

If the data product label contains pointer statements that reference files located in two different locations, then multiple paths can be specified.

The following command demonstrates the validation of a data product label that contains pointer statements that reference files located in two different locations, path1 and path2.

```
% VTool LABELPTR2.LBL -d pdsdd.full -I $HOME/path1, $HOME/path2
```

Not Following a Pointer

If a data product label points to a label fragment, VTool has the capability of validating this label without having to validate the associated label fragment with the no follow pointers option. This would be used in cases where the label fragment is not complete or in cases where the user wishes to simply check that the parent label structure is PDS compliant.

The following command demonstrates the validation of a data product label that contains a label fragment, but specifying to the tool to not follow the pointer:

```
% VTool LABEL.LBL -d pdsdd.full -F
```

Standalone Label Fragment Validation

The force option is used to force standalone label fragment validation. When this option is specified, standalone label fragment validation will occur on files whose extension ends in a .FMT

The following command demonstrates the validation of multiple data products, including performing standalone label fragment validation on any files it finds with a .FMT extension:

```
% VTool $HOME/DIR -d pdsdd.full -f
```

Progress Reporting

Validation progress reporting is enabled with the progress option. With progress reporting enabled, VTool will print, to the standard error, the current location that is being validated followed by a series of asterisk "*" symbols that represents a file it has encountered. The default behavior is to have this feature disabled.

The following command demonstrates the validation of multiple data products found in a directory, *DIR*, and its sub-directories with progress reporting enabled.

```
% VTool /home/user/DIR -d pdsdd.full -p
```

The progress reporting would look something like the following:

```
Validating file(s) in: file:/home/user/DIR
*
Validating file(s) in: file:/home/user/DIR/subDir1
*****
Validating file(s) in: file:/home/user/DIR/subDir2
***

...
```

In the above example, 1 file was validated in the DIR directory, 5 files were validated in DIR/subDir1, and 3 files were validated in DIR/subDir2.

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 option. 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:

```
% VTool -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 options.

The following command demonstrates the validation of a single data product label, specified implicitly, against the PSDD:

```
% VTool LABEL.LBL -d pdsdd.full
```

The following command demonstrates the validation of multiple data product labels, both specified implicitly, against the PSDD:

```
% VTool LABEL.LBL, $HOME/DIR -d pdsdd.full
```

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

```
% VTool -d pdsdd.full LABEL.LBL
```

In this example, VTool will inadvertently treat the implicit target, *LABEL.LBL*, as a dictionary 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 option.

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

```
% VTool LABEL1.LBL LABEL2.LBL -d pdsdd.full -t LABEL3.LBL, $HOME/DIR
```

In this example, LABEL1.LBL and LABEL2.LBL will get validated first, then LABEL3.LBL and the labels in $HOME/DIR will get validated next.

Report and Log Generation

This section describes how to generate a human-readable report and machine-readable log of a validation run (machine-readable log generation not available in this release). This is done in the following ways:

- Writing a Human-Readable Report to File
- Writing a Human-Readable Report to Standard Out
- Writing the Machine-Readable Log to Standard Out with No Human-Readable Report
- Writing the Machine-Readable Log to File, Writing the Human-Readable Report to Standard Out
- Writing Both the Machine-Readable Log and Human-Readable Report to File

Writing a Human-Readable Report to File

Specify the report file option to write the human-readable report to file. The report style option is used to specify the report format. If the style option is not specified, the default is to show a full report.

The following command demonstrates writing the human-readable, full report to a file named *report.txt*:

```
% VTool LABEL.LBL -d pdsdd.full -r report.txt
```

Writing a Human-Readable Report to Standard Out

Do not specify the report file option to write the human-readable report to standard out.

The following command demonstrates the validation of a single data product label against the PSDD, where the human-readable, full report is written to standard out:

```
% VTool LABEL.LBL -d pdsdd.full
```

The examples below will demonstrate how to generate a machine-readable log of a validation run.

Writing the Machine-Readable Log to Standard Out with No Human-Readable Report

Use the log file option *with no file specification* and do not specify the report file option.

The following command demonstrates writing the machine-readable log to standard out with no human-readable report generated:

```
% VTool LABEL.LBL -d pdsdd.full -l
```

Writing the Machine-Readable Log to File, Writing the Human-Readable Report to Standard Out

Specify the report file option and the log file option *with a file name specification.*

Note: When a directory is being validated and the log will be written to a file, the log should be written to a location different from the target directory. Otherwise, the tool will attempt to validate the log and it will be seen in the report.

The following command demonstrates writing the machine-readable log to a file named *log.xml* and writing a human-readable, full report to standard out.

```
% VTool LABEL.LBL -d pdsdd.full -l log.xml
```

Writing Both the Machine-Readable Log and Human-Readable Report to File

Specify the report file option and the log file option with a file specification. The file names should be different. The log and report cannot be written to the same file.

The following command demonstrates writing the machine-readable log to a file named *log.xml* and writing the human-readable report to a file named *report.txt*

```
% VTool LABEL.LBL -d pdsdd.full -l log.xml -r report.txt
```

Writing the Machine-Readable Log to Standard Out, Writing the Human-Readable Report to File

Specify the report file option and the log file option with no file specification.

The following command demonstrates writing the machine-readable log to standard out and writing the human-readable report to a file named *report.txt*

```
% VTool LABEL.LBL -d pdsdd.full -l -r report.txt
```

Multiple Dictionary Support

VTool allows multiple dictionary files to be passed in through the command-line via the "-d" option. When passing in multiple dictionaries, element and object definitions found in the dictionaries are merged together internally. In the case where definitions are found to be identical in each of the dictionary files being passed in, the valid value list from both dictionaries are combined.

The following example demonstrates the VTool behavior when passing in two dictionary files:

Suppose a PSDD, pdsdd.full, and a local dictionary, local-dd1.full, is being passed into VTool, where the PSDD contains the following DATA_SET_ID definition:

```
OBJECT = ELEMENT_DEFINITION
  NAME = DATA_SET_ID
  STATUS_TYPE = APPROVED
  GENERAL_DATA_TYPE = IDENTIFIER
  UNIT_ID = NONE
  STANDARD_VALUE_TYPE = FORMATION
  MAXIMUM_LENGTH = 40
  DESCRIPTION = "
     The data_set_id element is a unique
     alphanumeric identifier for a data set or a data product.
     The data_set_id value for a given data set or
     product is constructed according to flight project
     naming conventions.  In most cases the data_set_id is an
     abbreviation of the data_set_name.
     Example value: MR9/VO1/VO2-M-ISS/VIS-5-CLOUD-V1.0.
     Note:  In the PDS, the values for both data_set_id and
     data_set_name are constructed according to standards
     outlined in the Standards Reference."
  STANDARD_VALUE_SET = {
     "A-5-DDR-ASTERMAG-V1.0",
     "A-5-DDR-ASTEROID-SPIN-VECTORS-V3.0",
     "A-5-DDR-ASTNAMES-V1.0",
     "A-5-DDR-POLE-POSITION-REF-V1.0",
     "A-5-DDR-POLE-POSITION-V1.0",
     "A-5-DDR-TAXONOMY-V1.0",
     "ARCB-L-RTLS-3-70CM-V1.0",
     "ARCB-L-RTLS-4-70CM-V1.0",
     ....
     }
END_OBJECT = ELEMENT_DEFINITION
END
```

and the local dictionary also contains DATA_SET_ID, but with a different set of standard values:

```
OBJECT = ELEMENT_DEFINITION
  NAME = DATA_SET_ID
  STATUS_TYPE = APPROVED
  GENERAL_DATA_TYPE = IDENTIFIER
  UNIT_ID = NONE
  STANDARD_VALUE_TYPE = FORMATION
```

```
  MAXIMUM_LENGTH = 40
  DESCRIPTION = "
     The data_set_id element is a unique
     alphanumeric identifier for a data set or a data product.
     The data_set_id value for a given data set or
     product is constructed according to flight project
     naming conventions.  In most cases the data_set_id is an
     abbreviation of the data_set_name.
     Example value: MR9/VO1/VO2-M-ISS/VIS-5-CLOUD-V1.0.
     Note:  In the PDS, the values for both data_set_id and
     data_set_name are constructed according to standards
     outlined in the Standards Reference."
  STANDARD_VALUE_SET = {
     "VENUS"
     "EARTH"
  }
END_OBJECT = ELEMENT_DEFINITION
END
```

the *STANDARD_VALUE_SET* attribute from both definitions will be combined. As a result, the *STANDARD_VALUE_SET* will have the following values:

```
...
  STANDARD_VALUE_SET = {
     "A-5-DDR-ASTERMAG-V1.0",
     "A-5-DDR-ASTEROID-SPIN-VECTORS-V3.0",
     "A-5-DDR-ASTNAMES-V1.0",
     "A-5-DDR-POLE-POSITION-REF-V1.0",
     "A-5-DDR-POLE-POSITION-V1.0",
     "A-5-DDR-TAXONOMY-V1.0",
     "ARCB-L-RTLS-3-70CM-V1.0",
     "ARCB-L-RTLS-4-70CM-V1.0",
     ....
     "VENUS"
     "EARTH"
  }
...
```

Using a Configuration File

A configuration file is used to set the default behaviors of the tool. 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. The following rules apply to the content of configuration files:

- 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 tool even if it is surrounded by quotes. A common example would be a Windows path name (e.g. c:\VTT_EN_1-1\target). Use the forward slash character instead (c:/VTT_EN_1-1/target) or escape the backslash character (c:\\VTT_EN_1-1\\target).

Note: Any option 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 Option	Valid Value(s)
vtool.target	-t	Specify labels,directories, and URLs
vtool.dict	-d	Specify dictionary files
vtool.log	-l	(Not available in this release) To write the log to standard out, set to 'true'. To write the log to a file, set to a file name. Otherwise, do not specify this keyword at all (or simply set to 'false').
vtool.report	-r	Specify the human-readable report file name (do not specify keyword if wanting to write to standard out)
vtool.style	-s	Specify the standard human-readable report format ("full", "min", or "sum"). If not specified, then a full report will be generated
vtool.verbose	-v	Specify the message severity level and above to include in the human-readable report(1=Info, 2=Warning, 3=Error). Default is warning and above (level 2).
vtool.includepaths	-I	Specify paths to search for files refererenced by pointers in a label
vtool.follow	-F	Set to 'false' to not look for files referenced by pointer statements in a label, set to 'true' otherwise
vtool.force	-f	Set to 'true' to enable standalone label fragment validation, 'false' otherwise
vtool.recursive	-L	Set to 'false' to not traverse a directory, set to 'true' otherwise
vtool.regexp	-e	Specify file patterns to search for when validating a directory.
vtool.ignorefile	-X	Specify file patterns to ignore when validating a directory.
vtool.ignoredir	-D	Specify directory patterns to ignore.

Property Keyword	Associated Option	Valid Value(s)
vtool.progress	-p	Set to 'true' to enable progress reporting, 'false' otherwise.

The following example demonstrates how to set a configuration file:

```
# This is a VTool configuration file

vtool.target = ./TEST_DIR
vtool.dict = pdsdd.full
vtool.log = log.xml
vtool.report = rpt.txt
vtool.recursive = false
vtool.includepaths = /home/path
vtool.regexp = "*.LBL"
```

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

```
-t ./TEST_DIR -d pdsdd.full -l log.xml -L -e "*.LBL" -I /home/path -r rpt.txt
```

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

```
# This is a VTool configuration file with multiple values

vtool.target = TEST.LBL, ./TEST_DIR
vtool.dict = pdsdd.full, localdd.full
vtool.recursive = true
vtool.includepaths = /home/path1, /home/path2
vtool.regexp = "*.LBL", "*.FMT"
```

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

```
-t TEST.LBL, ./TEST_DIR -d pdsdd.full, localdd.full -e "*.LBL", "*.FMT" \
-I /home/path1, /home/path2
```

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

multiple lines:

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

vtool.target = TEST.LBL, \
               ./TEST_DIR
vtool.dict = pdsdd.full, \
             localdd.full
vtool.recursive = true
vtool.regexp = "*.LBL", \
                "*.FMT"
```

The following example demonstrates how to override a setting in the configuration file.

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

```
# This is another VTool configuration file

vtool.target = ./TEST_DIR
vtool.dict = pdsdd.full
vtool.log = log.xml
vtool.recursive = false
vtool.regexp = "*.LBL"
```

If validating everything in TEST_DIR is desired rather than just files that end in "LBL" as is defined in the configuration file, then the following command demonstrates how to perform this behavior:

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

Report Formats

This section describes the contents of the validation report formats. The links below detail the validation results of the same run for each format. If viewing this document in PDF form, see the appendix for the actual examples.

The tool can represent a validation report in three different formats: a full, a summary , and a minimal format. The report style option is used to change the formatting. When this option is not specified on the command-line, the default is to generate a full report.

Full Report

In a full report, the location, severity, and textual description of each detected anomaly is reported. A 'PASS', 'FAIL', or 'SKIP' keyword is displayed next to each file to indicate when a file has passed, failed, or skipped PDS validation, respectively.

Summary Report

In a summary report, the set of detected anomalies are summarized by reporting the number of occurrences for each type of anomaly and providing the location of one example for that anomaly.

Minimal Report

In a minimal report, only the number of anomalies detected are reported for each file. The anomalies are grouped by severity level.

Exit Status Values

The Validation Tool returns an exit status based on the PDS validation results or if an application failure has occurred. This section details the values that can be returned as a result of a single validation run.

The following table shows what exit values can be returned when the tool validates one or more data product labels in a single run:

Integer Value	Binary Value	Meaning
0	0000 0000	The data product label(s) passed the PDS validation step with no errors or warnings.
1	0000 0001	An Application error has occurred.
2	0000 0010	A System error has occurred.
32	0010 0000	One or more files skipped during the validation run.
64	0100 0000	One or more validation warnings were encountered during the validation run.
96	0110 0000	One or more files skipped and validation warnings were encountered during the validation run.
128	1000 0000	One or more validation errors were encountered during the validation run.
160	1010 0000	One or more files skipped and validation errors were encountered during the validation run.

Integer Value	Binary Value	Meaning
192	1100 0000	One ore more validation warnings and errors were encountered during the validation run.
224	1110 0000	One or more files skipped. In addition, validation warnings and errors were encountered during the validation run.

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 VTool.

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/vtool-2.2.0/lib \
gov.nasa.pds.vtool.VTool <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/vtool-2.2.0/lib/vtool-2.2.0.jar <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 VTool.

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:\vtool-2.2.0\lib \
gov.nasa.pds.vtool.VTool <command-line arguments>
```

Specify the JAR on the Command-Line

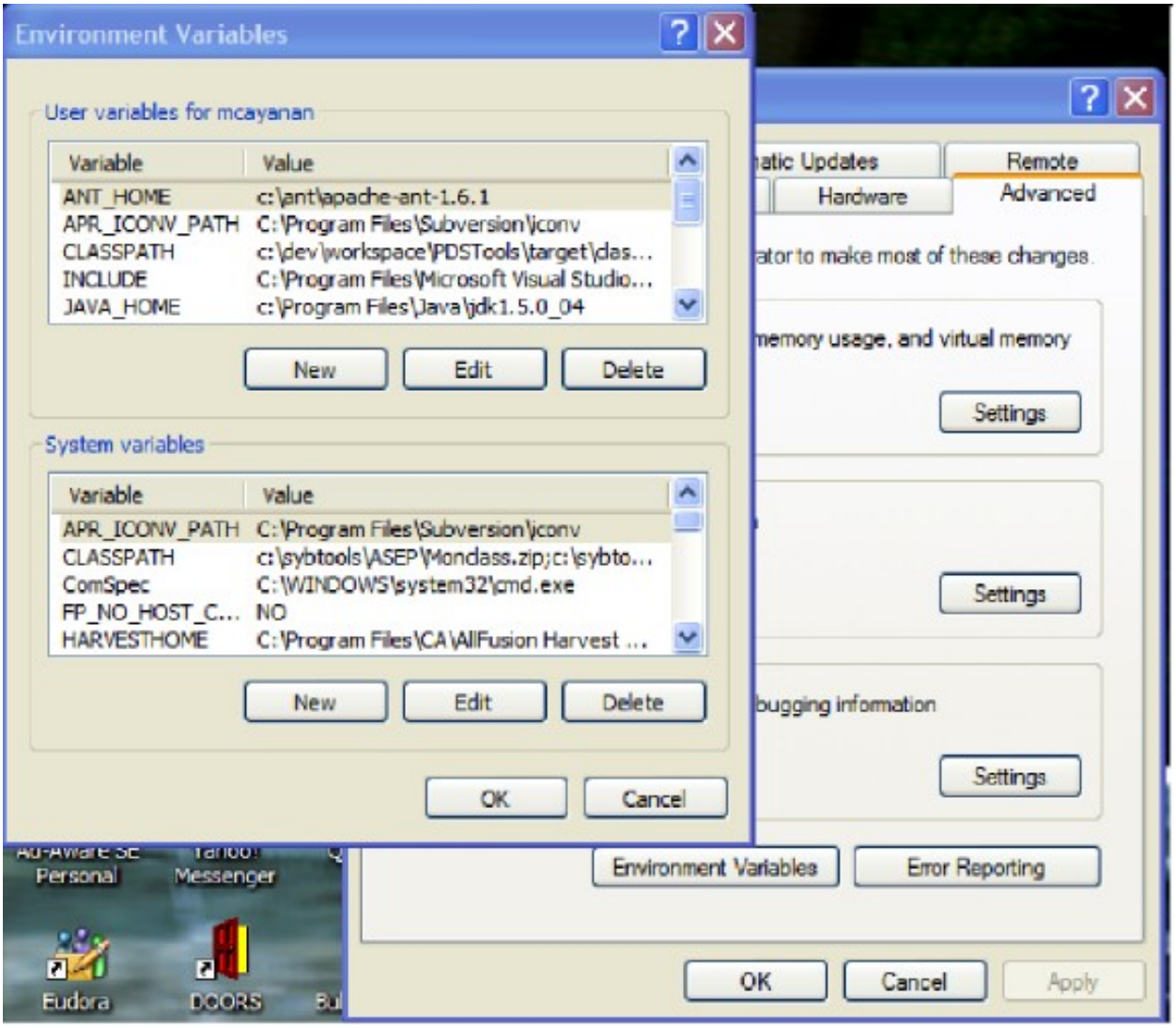
Another alternative method is to specify the executable JAR file on the command-line when running the tool each time. This is done by passing the JAR file specification via the Java "-jar" flag as demonstrated in the following example:

```
C:\> java -jar c:\vtool-2.2.0\lib\vtool-2.2.0.jar <command-line arguments>
```

Specify the Path in the Control Panel

The method for setting the executable path permanently for VTool 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 *vtool* 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 - Report Examples

Report Examples

This section details the various report formats available from the Validation Tool.

1.7.1 Full Report

Full Report Example

The following is an example of a full report:

```
PDS Validation Tool Report

Configuration:
  Version                   2.1.0-dev
  Date                      Fri, Feb 05 2010 at 08:49:23 AM
  Level of Validation       Syntactic and Semantic
  Dictionary version        1r77

Parameters:
  Target(s)                       [DATA]
  Dictionary File(s)              [..\..\UTIL\pdsdd.full]
  Aliasing                        false
  Directory Recursion             true
  Follow Pointers                 true
  Validate Standalone Fragments   false
  Report File                     report-full.txt
  Report Style                    full
  Severity Level                  INFO
  Include Path(s)                 [LABEL, DOCUMENT]
  Progress Reporting              false

Validation Details:

  FAIL: file:/C:/tool-tests/VTT_EN_19-1/target/DATA/COMBINED_DETACHED.LBL
      ERROR  line 20: No definition was found for the key "CASSINI:ADC_TIMING_SETTINGS".
      ERROR  line 21: No definition was found for the key
"CASSINI:BARREL_BAFFLE_TEMPERATURE".
      ERROR  line 23: No definition was found for the key "CASSINI:CPMM_NUMBER".
      ERROR  line 25: No definition was found for the key "CASSINI:POWERED_CPMM_FLAG".
      ERROR  line 26: No definition was found for the key "CASSINI:BINNING".
      WARNING  line 28: "?" is not in the list of valid values for
"MRO:ADC_TIMING_SETTINGS". It may be that the value needs to be added to the dictionary.

  PASS: file:/C:/tool-tests/VTT_EN_19-1/target/DATA/COMP_ARRAY-1.LBL

  SKIP: file:/C:/tool-tests/VTT_EN_19-1/target/DATA/COMP_ARRAY.DAT
      ERROR  "COMP_ARRAY.DAT" is not a label. Could not find the PDS_VERSION_ID in the
first line.

  PASS: file:/C:/tool-tests/VTT_EN_19-1/target/DATA/COMP_HISTORY-1.LBL
```

```
  SKIP: file:/C:/tool-tests/VTT_EN_19-1/target/DATA/COMP_IMG-1.IMG
      ERROR  "COMP_IMG-1.IMG" is not a label. Could not find the PDS_VERSION_ID in the
first line.

  PASS: file:/C:/tool-tests/VTT_EN_19-1/target/DATA/COMP_IMG-1.LBL

  PASS: file:/C:/tool-tests/VTT_EN_19-1/target/DATA/COMP_PALETTE-1.LBL

  SKIP: file:/C:/tool-tests/VTT_EN_19-1/target/DATA/COMP_PALETTE.TAB
      ERROR  "COMP_PALETTE.TAB" is not a label. Could not find the PDS_VERSION_ID in the
first line.

  SKIP: file:/C:/tool-tests/VTT_EN_19-1/target/DATA/COMP_SPECTRUM.DAT
      ERROR  "COMP_SPECTRUM.DAT" is not a label. Could not find the PDS_VERSION_ID in
the first line.

  PASS: file:/C:/tool-tests/VTT_EN_19-1/target/DATA/COMP_SPECTRUM.LBL

  PASS: file:/C:/tool-tests/VTT_EN_19-1/target/DATA/COMP_SPREADSHEET-1.LBL

  SKIP: file:/C:/tool-tests/VTT_EN_19-1/target/DATA/HISTORY.TXT
      ERROR  "HISTORY.TXT" is not a label. Could not find the PDS_VERSION_ID in the
first line.

  SKIP: file:/C:/tool-tests/VTT_EN_19-1/target/DATA/LEAPSECS.KER
      ERROR  "LEAPSECS.KER" is not a label. Could not find the PDS_VERSION_ID in the
first line.

  SKIP: file:/C:/tool-tests/VTT_EN_19-1/target/DATA/MYDATA.CSV
      ERROR  "MYDATA.CSV" is not a label. Could not find the PDS_VERSION_ID in the first
line.

  PASS: file:/C:/tool-tests/VTT_EN_19-1/target/DATA/NONCOMP_ATTACHED.LBL

  SKIP: file:/C:/tool-tests/VTT_EN_19-1/target/DATA/T92025.TAB
      ERROR  "T92025.TAB" is not a label. Could not find the PDS_VERSION_ID in the first
line.

  SKIP: file:/C:/tool-tests/VTT_EN_19-1/target/DATA/W1477654052.IMG
      ERROR  "W1477654052.IMG" is not a label. Could not find the PDS_VERSION_ID in the
first line.

Summary:

  8 of 17 validated, 9 skipped
  7 of 8 passed

End of Report
```

1.7.2 Summary Report

Summary Report Example

The following is an example of a summary report:

```
PDS Validation Tool Report

Configuration:
  Version                   2.1.0-dev
  Date                      Fri, Feb 05 2010 at 08:49:32 AM
  Level of Validation       Syntactic and Semantic
  Dictionary version        1r77

Parameters:
  Target(s)                       [DATA]
  Dictionary File(s)              [..\..\UTIL\pdsdd.full]
  Aliasing                        false
  Directory Recursion             true
  Follow Pointers                 true
  Validate Standalone Fragments   false
  Report File                     report-summary.txt
  Report Style                    sum
  Severity Level                  INFO
  Include Path(s)                 [LABEL, DOCUMENT]
  Progress Reporting              false


Errors Found:

  ERROR No definition was found for the key "CASSINI:POWERED_CPMM_FLAG".
  Example: line 25 of file:/C:/tool-tests/VTT_EN_19-1/target/DATA/COMBINED_DETACHED.LBL
  1 occurence(s)

  ERROR No definition was found for the key "CASSINI:CPMM_NUMBER".
  Example: line 23 of file:/C:/tool-tests/VTT_EN_19-1/target/DATA/COMBINED_DETACHED.LBL
  1 occurence(s)

  ERROR No definition was found for the key "CASSINI:ADC_TIMING_SETTINGS".
  Example: line 20 of file:/C:/tool-tests/VTT_EN_19-1/target/DATA/COMBINED_DETACHED.LBL
  1 occurence(s)

  ERROR No definition was found for the key "CASSINI:BINNING".
  Example: line 26 of file:/C:/tool-tests/VTT_EN_19-1/target/DATA/COMBINED_DETACHED.LBL
  1 occurence(s)

  ERROR No definition was found for the key "CASSINI:BARREL_BAFFLE_TEMPERATURE".
```

```
  Example: line 21 of file:/C:/tool-tests/VTT_EN_19-1/target/DATA/COMBINED_DETACHED.LBL
  1 occurence(s)


Warnings Found:

  WARN "?" is not in the list of valid values for "MRO:ADC_TIMING_SETTINGS". It may be
that the value needs to be added to the dictionary.
  Example: line 28 of file:/C:/tool-tests/VTT_EN_19-1/target/DATA/COMBINED_DETACHED.LBL
  1 occurence(s)


Info Found:


Summary:

  8 of 17 validated, 9 skipped
  7 of 8 passed

End of Report
```

1.7.3 Minimal Report

Minimal Report Example

The following is an example of a minimal report:

```
PDS Validation Tool Report

Configuration:
  Version                   2.1.0-dev
  Date                      Fri, Feb 05 2010 at 08:49:39 AM
  Level of Validation       Syntactic and Semantic
  Dictionary version        1r77

Parameters:
  Target(s)                       [DATA]
  Dictionary File(s)              [..\..\UTIL\pdsdd.full]
  Aliasing                        false
  Directory Recursion             true
  Follow Pointers                 true
  Validate Standalone Fragments   false
  Report File                     report-min.txt
  Report Style                    min
  Severity Level                  INFO
  Include Path(s)                 [LABEL, DOCUMENT]
  Progress Reporting              false

Message Counts:

 ERROR    WARN    INFO    FILE
     5       1       0
file:/C:/tool-tests/VTT_EN_19-1/target/DATA/COMBINED_DETACHED.LBL
     0       0       0    file:/C:/tool-tests/VTT_EN_19-1/target/DATA/COMP_ARRAY-1.LBL
     0       0       0    file:/C:/tool-tests/VTT_EN_19-1/target/DATA/COMP_HISTORY-1.LBL
     0       0       0    file:/C:/tool-tests/VTT_EN_19-1/target/DATA/COMP_IMG-1.LBL
     0       0       0    file:/C:/tool-tests/VTT_EN_19-1/target/DATA/COMP_PALETTE-1.LBL
     0       0       0    file:/C:/tool-tests/VTT_EN_19-1/target/DATA/COMP_SPECTRUM.LBL
     0       0       0
file:/C:/tool-tests/VTT_EN_19-1/target/DATA/COMP_SPREADSHEET-1.LBL
     0       0       0
file:/C:/tool-tests/VTT_EN_19-1/target/DATA/NONCOMP_ATTACHED.LBL
------------------------
     5       1       0


Summary:
```

```
  8 of 17 validated, 9 skipped
  7 of 8 passed

End of Report
```