



## **Product Tools v.1.0.0**

**for the Planetary Data System**



# Table of Contents

---

|          |  |    |
|----------|--|----|
| <b>1</b> | <b>Product Tools Guide</b>                         |    |
| 1.1      | Overview .....                                     | 1  |
| 1.2      | Installation .....                                 | 2  |
| 1.3      | Use and Operation .....                            | 5  |
| 1.4      | Development .....                                  | 26 |
| 1.5      | Appendix A - Using the Windows Control Panel ..... | 28 |
| 1.6      | Appendix B - Report Examples .....                 | 30 |
| 1.6.1    | Full Report .....                                  | 31 |
| 1.6.2    | Summary Report .....                               | 35 |
| 1.6.3    | Minimal Report .....                               | 39 |



## 1.1 Overview

---

### About Product Tools

The next generation of Planetary Data System (PDS) Product Tools will support design/generation, validation and submission of archival products to the PDS. These tools are intended to replace the current generation of tools (e.g., lvtool, kwvtool, tbtool, etc.) used by the Discipline Nodes, Engineering Node and the PDS user community as a whole.

The current scope of the task is limited to PDS label and product validation, as directed by the PDS Management Council on October 5, 2005 at the Management Council face-to-face meeting. Other aspects of validation (e.g., catalog, volume and data set validation) will be covered in future phases.

## 1.2 Installation

---

### Installing the Tools

This section describes how to install the tools contained in the *product-tools* package. The following topics can be found in this section:

- [System Requirements](#)
- [Unpacking the Tools Package](#)
- [Modifying Shell Scripts](#)

### System Requirements

#### Java Requirement

The tools contained in the *product-tools* package were developed using Java and will run on any platform with a supported Java Runtime Environment (JRE). The tools were specifically developed under Sun Java version 1.4, so the tool will execute correctly under versions 1.4 or 1.5.

Since the tools are 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. Testing has been performed with GNU's Java environment. The result is that release 4.1.1 at a minimum is required for nominal operations.

#### Data Dictionary Requirement

With this release of VTool, the requirement of using the provided PDS data dictionary is no longer necessary. Through the efforts of many, the anomalies found in past versions of the data dictionary have been corrected in release *1r64*. So, this release or later of the data dictionary is required for VTool. Release *1r66* of the data dictionary supports the validation of explicit FILE objects. The latest version of the PDS data dictionary can be retrieved from the [PDS Data Dictionary Lookup](#) web page.

### Unpacking the Tools Package

Download the *product-tools* package from the [PDS Software Download](#) web page. The binary and source distributions are available in identical zip or tar/gzip packages. Unpack the selected file with one of the following commands (where X.X.X is the current version):

```
[node: ~] unzip product-tools-X.X.X.zip  
or  
[node: ~] tar -xzf product-tools-X.X.X.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 above commands result in the creation of the *product-tools-X.X.X* directory with the following directory structure:

- **LICENSE.txt**

The copyright notice from the California Institute of Technology detailing the restrictions regarding the use and distribution of this software.

- **bin/**

This directory contains the executable jar file containing the Validation Tool software along with a batch and shell script for executing the tool.

- **docs/**

This document directory contains a local web site with the Product Tools 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 jar file (*product-tools-X.X.X.jar*) containing the Validation Tool software.

## Modifying Shell Scripts

A batch file, *VTool.bat*, and shell script, *VTool*, are included in the *product-tools* package to provide an easy way of running the Validation Tool on Windows and Unix-based environments, respectively.

In both the batch file and shell script, there is a variable named *jarpath* that needs to be modified before the scripts can be executed. The variable needs to be set to the correct location of the executable *product-tools* jar file.

The following demonstrates the *jarpath* being set to the product-tools jar file location. The example contains X.X.X, which should be replaced with the current version of the *product-tools* package.

Note: The example has been broken into multiple lines for readability.

```
set jarpath = \  
"/home/user/product-tools-X.X.X/bin/product-tools-X.X.X-app.jar"
```



## 1.3 Use and Operation

---

### Overview

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](#)
- [Common Errors](#)

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

### Tool 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 providing four different methods:

- Specify the Shell Script on the Command-Line
- Set the CLASSPATH Environment Variable
- Specify the CLASSPATH on the Command-Line
- Specify the Jar on the Command-Line

#### *Specify the Shell Script on the Command-Line*

The preferred method is to specify the shell script, *VTool*, on the command-line. After modifying the script as described in the [Modifying Shell Scripts](#) section, set the *PATH* environment variable to the location of the script. This enables the shell script to be executed from any location in the user machine.

The following command demonstrates how to set the *PATH* environment variable, by appending to its current setting. The example contains *X.X.X*, which represents the current version of the *product-tools* package.

```
[node:~] setenv PATH ${PATH}:\
$HOME/product-tools-X.X.X/bin
```

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

```
[node:~] VTool <command-line arguments>
```

### ***Set the CLASSPATH Environment Variable***

An alternative method is to set the *CLASSPATH* environment variable. The following commands demonstrate how to set this variable, by appending to its current setting. The example contains *X.X.X*, which represents the current version of the *product-tools* package.

The first example and preferred method for setting the variable, appends the executable jar file found in the *bin* directory:

```
[node:~] setenv CLASSPATH ${CLASSPATH}:\
$HOME/product-tools-X.X.X/bin/product-tools-X.X.X-app.jar

[node:~] echo $CLASSPATH
```

The second example separately appends the dependent jar files, found in the *lib* directory:

```
[node:~] setenv CLASSPATH ${CLASSPATH}:\
$HOME/product-tools-X.X.X/lib/antlr-2.7.6.jar:\
$HOME/product-tools-X.X.X/lib/commons-cli-1.0.jar:\
$HOME/product-tools-X.X.X/lib/commons-collections-3.1.jar:\
$HOME/product-tools-X.X.X/lib/commons-configuration-1.2.jar:\
$HOME/product-tools-X.X.X/lib/commons-io-1.2.jar:\
$HOME/product-tools-X.X.X/lib/commons-lang-2.1.jar:\
$HOME/product-tools-X.X.X/lib/commons-logging-1.0.3.jar:\
$HOME/product-tools-X.X.X/lib/product-tools-X.X.X.jar

[node:~] echo $CLASSPATH
```

The second command in both of the examples above, will display the current value of the *CLASSPATH* variable. Please note that the value for the *CLASSPATH* variable may not contain space characters. Once the *CLASSPATH* is set, the tool can be executed with the following command:

```
[node:~] java gov.nasa.pds.tools.VTool <command-line arguments>
```

### ***Specify the CLASSPATH on the Command-Line***

An alternative method to setting the *CLASSPATH* variable 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:

```
[node:~] java -Djava.ext.dirs=$HOME/product-tools-X.X.X/lib \
gov.nasa.pds.tools.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:

```
[node:~] java -jar \
$HOME/product-tools-X.X.X/bin/product-tools-X.X.X-app.jar \
<command-line arguments>
```

## **Windows Setup**

This section details the environment setup for Windows machines providing four different methods:

- Specify the Batch File on the Command-Line
- Set the CLASSPATH Environment Variable
- Specify the CLASSPATH on the Command-Line
- Specify the Jar on the Command-Line

### ***Specify the Batch File on the Command-Line***

The preferred method is to specify the batch file, *VTool.bat*, on the command-line. After modifying the batch file as described in the [Modifying Shell Scripts](#) section, set the *PATH* environment variable to the location of

the file. This enables the batch to be executed from any location in the user machine.

The following command demonstrates how to set the *PATH* environment variable, by appending to its current setting. The example contains *X.X.X*, which represents the current version of the *product-tools* package.

```
C:\> set PATH = %PATH%;\  
C:\product-tools-X.X.X\bin
```

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

```
C:\> VTool <command-line arguments>
```

### ***Set the CLASSPATH Environment Variable***

An alternative method is to set the *CLASSPATH* environment variable. The following commands demonstrate how to set this variable, by appending to its current setting. The example contains *X.X.X*, which represents the current version of the *product-tools* package.

The first example and preferred method for setting the variable, appends the executable jar file found in the *bin* directory:

```
C:\> set CLASSPATH=%CLASSPATH%;\  
c:\product-tools-X.X.X\bin\product-tools-X.X.X-app.jar  
  
C:\> echo %CLASSPATH%
```

The second example separately appends the dependent jar files, found in the *lib* directory:

```
C:\> set CLASSPATH=%CLASSPATH%;\  
c:\product-tools-X.X.X\lib\antlr-2.7.6.jar;\  
c:\product-tools-X.X.X\lib\commons-cli-1.0.jar;\  
c:\product-tools-X.X.X\lib\commons-collections-3.1.jar;\  
c:\product-tools-X.X.X\lib\commons-configuration-1.2.jar;\  
c:\product-tools-X.X.X\lib\commons-io-1.2.jar;\  
c:\product-tools-X.X.X\lib\commons-lang-2.1.jar;\  
c:\product-tools-X.X.X\lib\commons-logging-1.0.3.jar;\  
c:\product-tools-X.X.X\lib\product-tools-X.X.X.jar  
  
C:\> echo %CLASSPATH%
```

The second command in both of the examples above, will display the current value of the *CLASSPATH* variable. Please note that the value for the *CLASSPATH* variable may not contain space characters. Once the *CLASSPATH* is set, the tool can be executed with the following command:

```
C:\> java gov.nasa.pds.tools.VTool <command-line arguments>
```

Another way of setting the *CLASSPATH* is via the [Windows Control Panel](#) . If viewing this document in PDF form, see the appendix for details on this method.

### ***Specify the CLASSPATH on the Command-Line***

An alternative method to setting the *CLASSPATH* variable 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:\product-tools-X.X.X\lib \
gov.nasa.pds.tools.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:\product-tools-X.X.X\bin\product-tools-X.X.X-app.jar \
<command-line arguments>
```

## **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:

## VTool Command-Line Options

|                                  |  |
|----------------------------------|--|
| -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 <a href="#">Specifying Targets</a> section.   |
| -d, --dict <.full files>         | Specify the Planetary Science Data Dictionary full file name and any local dictionaries.   |
| -l, --log-file <file (optional)> | 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 <a href="#">Report and Log Generation</a> section.  |
| -r, --report-file <file>         | Specify the file name for the human-readable report. Default is to write to the standard out if this flag is not specified. This report, however, will not print to standard out if this flag is missing AND the log file flag is specified with no file name. For more details on how to create reports, see the <a href="#">Report and Log Generation</a> 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 flag is not specified. For more details on these report styles, see the <a href="#">Report Formats</a> 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 space. 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 space. Patterns should be surrounded in quotes ("*.LBL *.FMT" or "*.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 space. Patterns should be surrounded in quotes ("*.TXT *.TAB" or "*.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 EXTRAS" or "LABEL" "EXTRAS") 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.                    |

**VTool Command-Line Options**

|                |   |
|----------------|---|
| -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.  |

**Running VTool**

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

- 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

Note: The command-line examples below are executing VTool via the batch/shell script. Alternatively, the `java -jar` or `java gov.nasa.pds.tools.VTool` command can be used to execute the tool as mentioned in the [Tool Setup](#) section. Make sure the user environment is configured properly prior to calling the tool with your preferred method.

***Validating Against a Single Dictionary***

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

```
[node:~] 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:

```
[node:~] 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":

```
[node:~] 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 flag.

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

```
[node:~] VTool $HOME/DIR \  
-d pdsdd.full -D "EXTRAS" "LABEL"
```

The local flag 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:

```
[node:~] VTool $HOME/DIR \  
-d pdsdd.full -L
```

### ***Ignoring Files with a Specific File Pattern***



The ignore files flag 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":

```
[node:~] 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.

```
[node:~] 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.

```
[node:~] 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:

```
[node:~] VTool LABEL.LBL \  
-d pdsdd.full -F
```

### ***Standalone Label Fragment Validation***

The force flag is used to force standalone label fragment validation. When this flag 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:

```
[node:~] VTool $HOME/DIR \  
-d pdsdd.full -f
```

### ***Progress Reporting***

Validation progress reporting is enabled with the progress flag. 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.

```
[node:~] 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 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:

```
[node:~] 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 option flags.

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

```
[node:~] VTool LABEL.LBL \  
-d pdsdd.full
```

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

```
[node:~] VTool LABEL.LBL $HOME/DIR \  
-d pdsdd.full
```

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

```
[node:~] 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 flag.

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

```
[node:~] VTool LABEL1.LBL \  
LABEL2.LBL -d psdd.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. This is done in the following ways:

- Writing a Human-Readable Report to File
- Writing a Human-Readable Report to Standard Out
- Validating Files with a Specific File Pattern
- 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 flag to write the human-readable report to file. The report style flag is used to specify the report format. If the style flag 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*:

```
[node:~] VTool LABEL.LBL \  
-d psdd.full -r report.txt
```

#### ***Writing a Human-Readable Report to Standard Out***

Do not specify the report file flag 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:

```
[node:~] VTool LABEL.LBL \  
-d pdsdd.full
```

In the above examples, the machine-readable log is written to memory. The examples below will demonstrate how to see the machine-readable log.

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

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

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

```
[node:~] 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 flag and the log file flag *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.

```
[node:~] 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 flag and the log file flag 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*

```
[node:~] 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 flag and the log file flag 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*

```
[node:~] 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" flag. 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 definition found in the dictionary file specified LAST on the command-line will be retained.

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

```

If the PSDD is passed in first followed by the local dictionary as follows:

```
... -d pdsdd.full local-dd1.full ...
```

VTool will retain the DATA\_SET\_ID definition found in the local dictionary.

If the local dictionary is passed in first followed by the PSDD as follows:

```
... -d local-dd1.full pdsdd.full ...
```

VTool will retain the DATA\_SET\_ID definition found in the PSDD.

The following example demonstrates the VTool behavior when passing in three dictionaries:

Suppose the PSDD and the local dictionary, local-dd1.full, file have the DATA\_SET\_ID definition as stated above. In addition, there is another local dictionary, local-dd2.full, that will be passed into VTool also, which 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 = {
    "MARS"
    "JUPITER"
}
END_OBJECT = ELEMENT_DEFINITION
END
```

If these three dictionaries get passed into VTool as follows:

```
... -d pdsdd.full local-dd1.full local-dd2.full ...
```



VTool will retain the DATA\_SET\_ID definition found in the last dictionary specified. In this case, it is the definition found in local-dd2.full.

If these dictionaries get passed into VTool in some other order like the following:

```
... -d local-dd2.full local-dd1.full pdsdd.full ...
```

VTool will retain the DATA\_SET\_ID definition found in pdsdd.full.

### 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.

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 VTool 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 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:

| KEYWORD      | ASSOCIATED FLAG | valid value(s)   |
|--------------|-----------------|--|
| vtool.target | -t              | Specify labels, directories, and URLs  |
| vtool.dict   | -d              | Specify dictionary files   |
| vtool.log    | -l              | 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)   |

| KEYWORD            | ASSOCIATED FLAG | valid value(s)  |
|--------------------|-----------------|---|
| 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 | -l              | Specify paths to search for files referenced 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.   |
| 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 flags:

```
-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 flags:

```
-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:

```
[node:~] 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 flag is used to change the formatting. When this flag 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.

## Common Errors

At this point in development there is a single common error that several users have encountered. The error is as follows:

```
[node:~] java gov.nasa.pds.tools.VTool ...  
Exception in thread "main" java.lang.NoClassDefFoundError:  
gov/nasa/pds/tools/VTool
```

The actual class name may vary but the above error is the result of the *CLASSPATH* environment variable not being set correctly. Verify that the variable contains all of the specified jar files and that there are no space characters in the value. See either the [UNIX-Based Setup](#) or [Windows Setup](#) section for details.

## 1.4 Development

---

### Developing Tools

This section describes how a user can utilize the underlying Application Program Interface (API) of the *product-tools* package to develop their own tool. This section also covers how to interpret the exit status values returned by the tool if the user calls the *VTool* class from a script.

#### API Entry Points

We have not had a chance to document the entry points into the API as of the current release. The best source for this information currently is to view the *VTool.java* file from the source package.

#### 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 what exit status values to expect when the tool is validating a single data product label and when it is validating multiple data product labels in a single run.

#### Validation Of A Single Data Product

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

| Exit Status Value | Meaning  |
|-------------------|--|
| 0                 | The data product label has passed the PDS validation step.                                     |
| 1                 | The data product label has failed or skipped the PDS validation step.                          |
| -1                | A tool application error has occurred (invalid command-line option, file cannot be read, etc.) |

#### Validation Of Multiple Data Products

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

| Exit Status Value | Meaning   |
|-------------------|---|
| 0                 | One of the following has occurred: <ul style="list-style-type: none"><li>• All data product labels have passed the PDS validation step.</li><li>• Data product labels have passed and skipped the PDS validation step.</li></ul>    |
| 1                 | One of the following has occurred: <ul style="list-style-type: none"><li>• One or more of the data product labels has failed the PDS validation step.</li><li>• All data product labels have skipped the validation step.</li></ul> |
| -1                | A tool application error has occurred (invalid command-line option, file cannot be read, etc.)  |

## 1.5 Appendix A - Using the Windows Control Panel

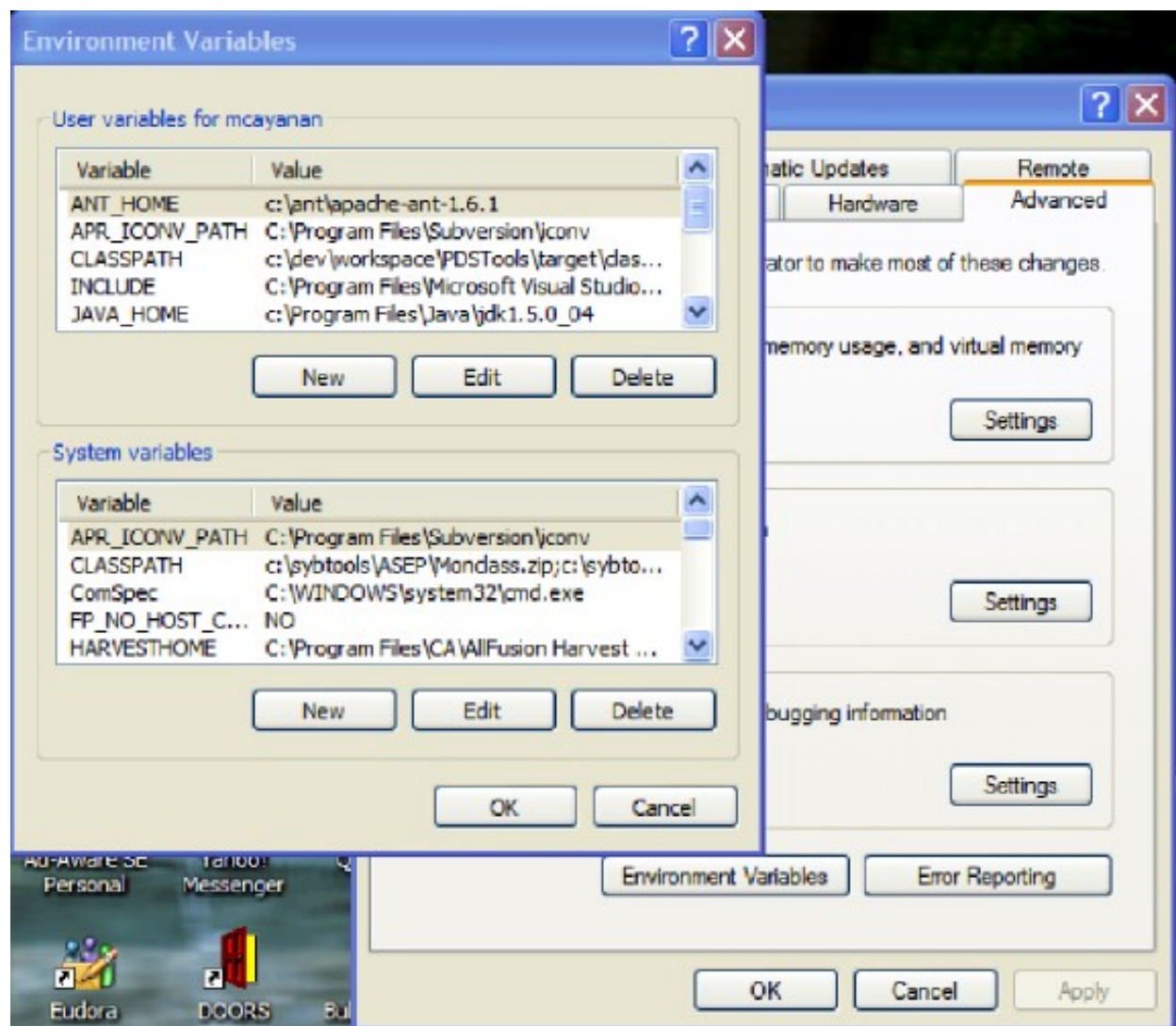
---

### Windows Environment Setup

For those attempting to run the Validation Tool in a Windows environment, here are the steps for setting the CLASSPATH environment variable via the control panel:

- Right-click on My Computer 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 *CLASSPATH* variable in the User Variables list and select the Edit button.
- Append to the current contents of the variable, the path(s) to the jar files that came with the *product-tools* package. See the *Windows Setup* section for actual jar file paths for the current release. Separate each path with a semicolon.
- Select the *OK* button when you are finished editing the *CLASSPATH* 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 *CLASSPATH* changes to take effect.

## 1.6 **Appendix B - Report Examples**

---

### **Report Examples**

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

## 1.6.1 Full Report

---

### Full Report Example

The following is an example of a full report:

```
PDS Validation Tool Report

VTool Version           1.0.0

Execution Date           Thu, Apr 26 2007 at 14:19:28 PM

Dictionary version
/* Planetary Science Data Dictionary database dump */
/* Start of alias definitions */
/* Version: OPS */
/* Online Database: pdscat1r65 */
/* Generated: Thu Feb 22 14:39:35 2007 */


Parameter Settings:

Target(s)                [VTT_EN_19-1\target\DATA]

Dictionary File(s)       [UTIL\pdsdd.full]

Aliasing                 false

Directory Recursion      true

Follow Pointers          true

Validate Standalone Fragments false

Report File              rpt-full.txt

Report Style             full

Severity Level           Warning

Include Path(s)          [VTT_EN_19-1\target\LABEL, VTT_EN_19-1\target\DOCUMENT]

Progress Reporting       false
```

## Summary:

8 of 8 validated, 9 skipped

7 of 8 passed

## Validation Details:

FAIL: file:/C:/tool-tests-0.4.0/tests/VTT\_EN\_19-1/target/DATA/COMBINED\_DETACHED.LBL

ERROR line 20: Undefined Element: CASSINI:ADC\_TIMING\_SETTINGS

ERROR line 21: Undefined Element: CASSINI:BARREL\_BAFFLE\_TEMPERATURE

ERROR line 22: Undefined Element: CASSINI:CPMM\_NUMBER

ERROR line 23: Undefined Element: CASSINI:POWERED\_CPMM\_FLAG

ERROR line 24: Undefined Element: CASSINI:BINNING

ERROR line 26: Undefined Element: MRO:ADC\_TIMING\_SETTINGS

ERROR line 27: Undefined Element: MRO:BARREL\_BAFFLE\_TEMPERATURE

ERROR line 28: Undefined Element: MRO:CPMM\_NUMBER

ERROR line 29: Undefined Element: MRO:POWERED\_CPMM\_FLAG

ERROR line 30: Undefined Element: MRO:BINNING

PASS: file:/C:/tool-tests-0.4.0/tests/VTT\_EN\_19-1/target/DATA/COMP\_ARRAY-1.LBL

SKIP: file:/C:/tool-tests-0.4.0/tests/VTT\_EN\_19-1/target/DATA/COMP\_ARRAY.DAT

WARNING Not a label. Could not find the PDS\_VERSION\_ID in the first line.

PASS: file:/C:/tool-tests-0.4.0/tests/VTT\_EN\_19-1/target/DATA/COMP\_HISTORY-1.LBL

SKIP: file:/C:/tool-tests-0.4.0/tests/VTT\_EN\_19-1/target/DATA/COMP\_IMG-1.IMG

WARNING Not a label. Could not find the PDS\_VERSION\_ID in the first line.

```
PASS: file:/C:/tool-tests-0.4.0/tests/VTT_EN_19-1/target/DATA/COMP_IMG-1.LBL

PASS: file:/C:/tool-tests-0.4.0/tests/VTT_EN_19-1/target/DATA/COMP_PALETTE-1.LBL

SKIP: file:/C:/tool-tests-0.4.0/tests/VTT_EN_19-1/target/DATA/COMP_PALETTE.TAB
      WARNING Not a label. Could not find the PDS_VERSION_ID in the first line.

SKIP: file:/C:/tool-tests-0.4.0/tests/VTT_EN_19-1/target/DATA/COMP_SPECTRUM.DAT
      WARNING Not a label. Could not find the PDS_VERSION_ID in the first line.

PASS: file:/C:/tool-tests-0.4.0/tests/VTT_EN_19-1/target/DATA/COMP_SPECTRUM.LBL

PASS: file:/C:/tool-tests-0.4.0/tests/VTT_EN_19-1/target/DATA/COMP_SPREADSHEET-1.LBL

SKIP: file:/C:/tool-tests-0.4.0/tests/VTT_EN_19-1/target/DATA/HISTORY.TXT
      WARNING Not a label. Could not find the PDS_VERSION_ID in the first line.

SKIP: file:/C:/tool-tests-0.4.0/tests/VTT_EN_19-1/target/DATA/LEAPSECS.KER
      WARNING Not a label. Could not find the PDS_VERSION_ID in the first line.

SKIP: file:/C:/tool-tests-0.4.0/tests/VTT_EN_19-1/target/DATA/MYDATA.CSV
      WARNING Not a label. Could not find the PDS_VERSION_ID in the first line.

PASS: file:/C:/tool-tests-0.4.0/tests/VTT_EN_19-1/target/DATA/NONCOMP_ATTACHED.LBL

SKIP: file:/C:/tool-tests-0.4.0/tests/VTT_EN_19-1/target/DATA/T92025.TAB
      WARNING Not a label. Could not find the PDS_VERSION_ID in the first line.

SKIP: file:/C:/tool-tests-0.4.0/tests/VTT_EN_19-1/target/DATA/W1477654052.IMG
```

WARNING Not a label. Could not find the PDS\_VERSION\_ID in the first line.

End Of Report

## 1.6.2 Summary Report

---

### Summary Report Example

The following is an example of a summary report:

```
PDS Validation Tool Report

VTool Version           1.0.0

Execution Date           Thu, Apr 26 2007 at 14:19:00 PM

Dictionary version
/* Planetary Science Data Dictionary database dump */
/* Start of alias definitions */
/* Version: OPS */
/* Online Database: pdscat1r65 */
/* Generated: Thu Feb 22 14:39:35 2007 */


Parameter Settings:

Target(s)                [VTT_EN_19-1\target\DATA]

Dictionary File(s)       [UTIL\pdsdd.full]

Aliasing                 false

Directory Recursion      true

Follow Pointers          true

Validate Standalone Fragments false

Report File              rpt-sum.txt

Report Style             sum

Severity Level           Warning

Include Path(s)          [VTT_EN_19-1\target\LABEL, VTT_EN_19-1\target\DOCUMENT]

Progress Reporting       false
```

**Summary:**

8 of 8 validated, 9 skipped

7 of 8 passed

**Errors Found:**

ERROR Undefined Element: CASSINI:ADC\_TIMING\_SETTINGS

Example: line 20 of  
file:/C:/tool-tests-0.4.0/tests/VTT\_EN\_19-1/target/DATA/COMBINED\_DETACHED.LBL

1 occurrence(s)

ERROR Undefined Element: CASSINI:BARREL\_BAFFLE\_TEMPERATURE

Example: line 21 of  
file:/C:/tool-tests-0.4.0/tests/VTT\_EN\_19-1/target/DATA/COMBINED\_DETACHED.LBL

1 occurrence(s)

ERROR Undefined Element: CASSINI:BINNING

Example: line 24 of  
file:/C:/tool-tests-0.4.0/tests/VTT\_EN\_19-1/target/DATA/COMBINED\_DETACHED.LBL

1 occurrence(s)

ERROR Undefined Element: CASSINI:CPMM\_NUMBER

Example: line 22 of  
file:/C:/tool-tests-0.4.0/tests/VTT\_EN\_19-1/target/DATA/COMBINED\_DETACHED.LBL

1 occurrence(s)

ERROR Undefined Element: CASSINI:POWERED\_CPMM\_FLAG

Example: line 23 of  
file:/C:/tool-tests-0.4.0/tests/VTT\_EN\_19-1/target/DATA/COMBINED\_DETACHED.LBL

1 occurrence(s)



ERROR Undefined Element: MRO:ADC\_TIMING\_SETTINGS

Example: line 26 of  
file:/C:/tool-tests-0.4.0/tests/VTT\_EN\_19-1/target/DATA/COMBINED\_DETACHED.LBL

1 occurrence(s)

ERROR Undefined Element: MRO:BARREL\_BAFFLE\_TEMPERATURE

Example: line 27 of  
file:/C:/tool-tests-0.4.0/tests/VTT\_EN\_19-1/target/DATA/COMBINED\_DETACHED.LBL

1 occurrence(s)

ERROR Undefined Element: MRO:BINNING

Example: line 30 of  
file:/C:/tool-tests-0.4.0/tests/VTT\_EN\_19-1/target/DATA/COMBINED\_DETACHED.LBL

1 occurrence(s)

ERROR Undefined Element: MRO:CPMM\_NUMBER

Example: line 28 of  
file:/C:/tool-tests-0.4.0/tests/VTT\_EN\_19-1/target/DATA/COMBINED\_DETACHED.LBL

1 occurrence(s)

ERROR Undefined Element: MRO:POWERED\_CPMM\_FLAG

Example: line 29 of  
file:/C:/tool-tests-0.4.0/tests/VTT\_EN\_19-1/target/DATA/COMBINED\_DETACHED.LBL

1 occurrence(s)

Warnings Found:

WARNING Not a label. Could not find the PDS\_VERSION\_ID in the first line.

Example: file:/C:/tool-tests-0.4.0/tests/VTT\_EN\_19-1/target/DATA/COMP\_ARRAY.DAT

9 occurrence(s)

End Of Report

## 1.6.3 Minimal Report

---

### Minimal Report Example

The following is an example of a minimal report:

```
PDS Validation Tool Report

VTool Version          1.0.0

Execution Date          Thu, Apr 26 2007 at 14:18:49 PM

Dictionary version
/* Planetary Science Data Dictionary database dump */
/* Start of alias definitions */
/* Version: OPS */
/* Online Database: pdscat1r65 */
/* Generated: Thu Feb 22 14:39:35 2007 */

Parameter Settings:

Target(s)               [VTT_EN_19-1\target\DATA]

Dictionary File(s)       [UTIL\pdsdd.full]

Aliasing                 false

Directory Recursion      true

Follow Pointers          true

Validate Standalone Fragments false

Report File              rpt-min.txt

Report Style             min

Severity Level           Warning

Include Path(s)          [VTT_EN_19-1\target\LABEL, VTT_EN_19-1\target\DOCUMENT]

Progress Reporting       false
```

## Summary:

8 of 8 validated, 9 skipped

7 of 8 passed

## Total Messages:

ERROR WARN

10 9

## Message Counts by File:

ERROR WARN FILE

10 0

file:/C:/tool-tests-0.4.0/tests/VTT\_EN\_19-1/target/DATA/COMBINED\_DETACHED.LBL

0 0 file:/C:/tool-tests-0.4.0/tests/VTT\_EN\_19-1/target/DATA/COMP\_ARRAY-1.LBL

0 1 file:/C:/tool-tests-0.4.0/tests/VTT\_EN\_19-1/target/DATA/COMP\_ARRAY.DAT

0 0 file:/C:/tool-tests-0.4.0/tests/VTT\_EN\_19-1/target/DATA/COMP\_HISTORY-1.LBL

0 1 file:/C:/tool-tests-0.4.0/tests/VTT\_EN\_19-1/target/DATA/COMP\_IMG-1.IMG

0 0 file:/C:/tool-tests-0.4.0/tests/VTT\_EN\_19-1/target/DATA/COMP\_IMG-1.LBL

0 0 file:/C:/tool-tests-0.4.0/tests/VTT\_EN\_19-1/target/DATA/COMP\_PALETTE-1.LBL

0 1 file:/C:/tool-tests-0.4.0/tests/VTT\_EN\_19-1/target/DATA/COMP\_PALETTE.TAB

0 1 file:/C:/tool-tests-0.4.0/tests/VTT\_EN\_19-1/target/DATA/COMP\_SPECTRUM.DAT

0 0 file:/C:/tool-tests-0.4.0/tests/VTT\_EN\_19-1/target/DATA/COMP\_SPECTRUM.LBL

0 0

file:/C:/tool-tests-0.4.0/tests/VTT\_EN\_19-1/target/DATA/COMP\_SPREADSHEET-1.LBL

0 1 file:/C:/tool-tests-0.4.0/tests/VTT\_EN\_19-1/target/DATA/HISTORY.TXT

0 1 file:/C:/tool-tests-0.4.0/tests/VTT\_EN\_19-1/target/DATA/LEAPSECS.KER

0 1 file:/C:/tool-tests-0.4.0/tests/VTT\_EN\_19-1/target/DATA/MYDATA.CSV

0 0

file:/C:/tool-tests-0.4.0/tests/VTT\_EN\_19-1/target/DATA/NONCOMP\_ATTACHED.LBL

```
0      1  file:/C:/tool-tests-0.4.0/tests/VTT_EN_19-1/target/DATA/T92025.TAB
0      1  file:/C:/tool-tests-0.4.0/tests/VTT_EN_19-1/target/DATA/W1477654052.IMG
```

End Of Report