European Space Agency
Research and Science Support Department
Planetary Missions Division

Planetary Science Archive
PVV User Manual
SOP-RSSD-UM-004

Issue 4.1

15 May 2007

Prepared by: D. Heather, I. Ortiz

Distribution List

Recipient	Organisation	Recipient	Organisation

Change Record Sheet

Date	Iss.	Rev.	pp.	Description / Authority	CR No.
23-04-04	2	2	All	General updates and restructuring throughout [DH]	
27-04-04	2	3	3	Installation section written (AV)	
26-05-04	2	4	All	Updates following MEX test and release of V1.0	
14-07-04	2	5	4 and 7	Minor updates in preparation for SMART-1 end-to-end test and after updates to release 1.05 of PVV	
21-10-04	2	6	All	Updated document with information for the official release of PVV 2.0	
2-05-05	3	0	All	Updated document for PVV release 2.2.0 including the ‘pvv build’ and ‘pvv upload’ functionality for the release concept.	
12-05-05	3	1	All	Updated to include the new ‘pvv freeze’ functionality, split from the ‘pvv build’ command	
05-02-07	4	0	All	Updated for the PVV 2.8 with various new functionality	

Table of Contents

1. Introduction

1.1 Scope

This document describes the installation and use of the 'PSA Volume Verifier' (PVV) tool used for the validation and delivery of a scientific dataset for ingestion to the Planetary Science Archive (PSA).

1.2 Contents

This document contains instructions on how to install and use the 'PSA Volume Verifier' (PVV) tool used for the validation and delivery of a scientific dataset for ingestion to the Planetary Science Archive (PSA). After an introduction and overview of the tool in Sections 1 and 2, the installation for both Unix and Windows platforms is covered in Section 3. This is followed by a detailed users guide and list of commands and options in Section 4, and a quick step-by-step guide to validating a dataset with the PVV for more experienced users in Section 5. Section 6 describes how to use the PVV with datasets following the Release Concept [AD6]. Section 7 outlines the implementation of the PSA and Mission Specific dictionaries. Section 8 outlines the error formatting and messages thrown by the PVV, and Section 9 details some features that have yet to be completed in the current version of the PVV. Finally, Section 10 reviews the frequently asked questions and some common problems and solutions.

1.3 Readership

This document will be of use to any instrument team member wishing to use the PVV tool to validate any aspect of their dataset before delivery, or to any member of the Archive team required to test and validate data sets and labels prior to data ingestion.

1.4 Acronyms

PSA	Planetary Science Archive
PVV	PSA Volume Verifier Tool
PDS	Planetary Data System
ESA	European Space Agency
ESTEC	European Space and Technology Center in Noordwijk, The Netherlands
RSSD	Research and Scientific Support Department of ESA
PI	Principal Investigator

1.5 Applicable Documents

AD1 Planetary Data System – National Space Science Data Center Memorandum of Understanding, 13 Jan 1994.

AD2 Planetary Data System Data Preparation Workbook, JPL D-7669, Part 1, Version 3.1, 17 Feb 1995.

AD3 Planetary Data System Standards Reference, JPL D-7669, Part 2, Version 3.6, 1 Aug 2003.

AD4 Planetary Science Data Dictionary Document, JPL D-7116, Revision E, 28 Aug 2002

AD5 PVV Error and Warning List, SOP-RSSD-LI-007, Version 1.9, 1 June 2004.

AD6 Planetary Science Archive Experiment Data Release Concept Technical Proposal, SOP-RSSD-TN-015, Issue 1.15 2 May 2005.

2. Overview – What is the PVV Tool?

The PVV is a tool constructed by the PSA team to allow instrument teams from all of ESA's planetary missions to check their datasets before they are delivered to the PSA database for ingestion into the long-term archive. The tool is run from the command line, and is currently supported on Solaris, Linux and Windows XP environments.

The tool allows a user to verify PDS compliance of a label, and validates all aspects of the data set structure / content prior to delivery to the PSA. The PVV will be systematically used by the PSA team to check data sets as part of the ingestion process to the Planetary Science Archive (PSA).

The PVV tool will allow for smooth and easy validation and ingestion of PDS compliant planetary data sets (e.g. data from Mars Express, SMART-1, Cassini-Huygens, Rosetta) into the PSA.

3. Installation Guide

3.1 Software Requirements

The PVV tool requires Java version 1.4.0 or above in order to run successfully. You must set your JAVA_HOME environment variable to the location of your Java installation. The PVV also requires a working network connection (access to an internet connection) in order to query the dictionary server.

3.2 New Installation

You will be provided with a tar.gz file for the PVV tool (e.g. PVV_1.0.tar.gz). The installation of the PVV then requires three basic steps:

- The unpacking of the PVV tar.gz archive on your local disk:
  - **UNIX**: >cd /home/<username>/<my_programs>

    >gunzip PVV_1.0.tar.gz

    >tar –xf PVV_1.0.tar.gz

    >ln –s PVV_1.0/ pvv #create a link to the installation
  - **WIN-XP**: In an explorer window, double click on the file PVV_1.0.tar.gz. Extract all files to say C:\PVV or any other disk/directory that you want (if you do make sure you change all further reference to C:\PVV below to the path you selected). Please do not choose a directory path with spaces in its name.
- Create an environment variable PVV_HOME that points to the directory where you installed the PVV and modify your PATH variable to point to the PVV_HOME/bin directory:
  - **UNIX**: open your ~/.tcshrc (or which ever shell start-up script you have) and add at the bottom the lines:

    setenv PVV_HOME /home/<username>/<my_programs>/pvv

    setenv PATH $PATH:$PVV_HOME/bin

- **WIN**-**XP**: Click 'Start' (bottom left hand corner of screen) -> 'Control Panel' -> System -> Advanced (Tab) -> 'Environment Variables'

  In the window under the user variables panel click the 'New' button and enter:

  Variable Name= PVV_HOME

  Variable Value=C:\ PVV\PVV_1.0

  Click 'OK'. Click the 'New' button under the user variables panel again and enter:

  Variable Name= PATH

  Variable Value=%PVV_HOME%\bin

  Note if you already have a PATH variable then just edit it and append the value above preceded with a ';'.

- Change the permissions of files $PVV_HOME/bin/* to executable.
  - **UNIX**: > chmod +x $PVV_HOME/bin/*
  - **WIN**-**XP**: In an explorer window select all files in the directory C:\PVV\PVV_1.0\bin and right click on them. Select 'properties' on the menu that pops up. Select your user name in the top panel and make sure you have the 'Read & execute' permissions box checked in the 'Allow' column.

N.B. For Windows XP users, please ensure that the datasets you are testing are in directories with NO SPACES in their pathnames, otherwise the PVV will fail.

You are now ready to start using the PVV.

3.3 Update to the latest version.

If you installed the PVV the first time following the above 'New Installation' procedure, then all you have to do is run 'pvv update PVV_X.X.tar.gz' (where 'X.X' denotes a future release number of the PVV) in a shell terminal (DOS terminal in windows) and the PVV will automatically install itself in a new directory (in the same directory where the previous installation was done) and reset your environment variables so that you are ready to use it straight away. Make sure you place the new tar.gz file in the installation directory for the previous version (i.e. one directory above your 'PVV_HOME' directory). Windows users should then restart their DOS window for the environment variables to take effect.

3.4 The Configuration File – Proxy Server Setup and Memory

In order to allow connection to the dictionary server via a proxy server, the PVV has a configuration file which allows the user to input their proxy server details. The configuration file is a simple text file found in:

$PVV_HOME/ext/pvv.properties

If a proxy server is to be used, the user should use their preferred text editor to edit this 'pvv.properties' configuration file and include details of their proxy server in the relevant section:

##########################

```
# PROXY SERVER SETTINGS #
############################
# Set the following properties if you are behind a firewall
# and cannot access the internet

# uncomment the next line and replace the url with your proxy's url
# if your proxy is on a different port than 80, then also set the
# the port property accordingly

#proxyHost=www.yourproxy.com
#proxyPort=8080
```

If your proxy server requires authentication, you should also use the following two keywords in the pvv.properties file:

```
proxyPassword = my password
proxyUserName = my login name
```

Similarly, within this configuration file is an option for the user to increase the amount of memory for the PVV to run. It is recommended that users initially leave this value set to the default 64Mb, but if they experience any problems with the error java.lang.OutOfMemoryError then they should try to increase this value:

```
##############################
# RUN-TIME MEMORY SETTINGS #
##############################
#
# this is the default memory allocated to the pvv at run time
# if you get errors such as java.lang.OutOfMemoryError, then
# try to increase this value to 128m, or more if you still
# get the error (note it is important to follow the value with
# the letter 'm' ie for MegaBytes)

maxMemory=64m
```

The final use of the configuration file is to set any default options you wish to run with the commands. This is described within the pvv.properties file and in Section 4.2.

4. Detailed User Guide and List of User Commands / Options

The PVV tool is run from the command line with the following general syntax:

pvv {command} *-D[command options] [argument]*

Where:

{command} is a single word that instructs the PVV to process the dataset in a specific way. The {command} must be the first argument on the line. There are several different {commands} that can be run, depending on the type of validation to be carried out.

[command options] are optional. The command options vary depending on the {command} being used. Each option must have the '-D' prefix, and be limited with the '=' sign.

[argument] is by default the current directory in most cases. However, it is sometimes required to be given as a file, depending on the command and options being used.

An example command line could therefore be:

pvv label *-DdataSetID=GIO-C-GRE-3-RDR-HALLEY-V1.0 DATA/PRODUCT_1.LBL*

In this case the **{command}** is 'label', which instructs the PVV to verify a label file.

The label file to be validated is specified in the *[argument]* field, and is given as 'PRODUCT_1.LBL' in the DATA directory.

The *[command option]* datasetID is also set in this example. This checks that the DATASET_ID in the label matches the name provided in the command line (GIO-C-GRE-3-RDR-HALLEY-V1.0 in this case).

A full list of commands and command options is provided in Section 4.2, along with details on what each command is used for.

4.1 PVV scan – The Dataset Image

Before you can do any dataset validation using the PVV, the software has to build an internal image of the dataset being tested. This is done by running the '**pvv scan**' command from the root directory of the dataset. The image created is stored as a hidden xml file called '.PDSVOLUME.XML' in the root directory, and provides a structural breakdown of the dataset being tested. This is then used by the PVV to verify the dataset.

*N.B. if any additions or changes are made to the dataset structure (e.g. altering the directory structure or changing directory or file names), you **MUST** re-run the 'scan' command in order to rebuild an image of the new dataset before you start verifying it. This must be done every time you change your dataset structure.*

4.2 Commands and Command Options

Below is a full list of the commands available in the PVV in alphabetical order, and the various command options and arguments permitted for use with each command.

N.B. If at any time you need to see a list of commands and options, use:
'**pvv help**'

---

browse - verifies label files in the BROWSE/QUICKLOOK directory.

*[command option]*s for this command are:

warn=[on/off] - turns warning messages on or off. The default is 'on'.

info=[on/off] - turns information messages on or off. The default is 'off'.

release=[0001-9999] - sets the release number, 0001 is the default (only applicable if you are using the Release Concept [AD6]. See Section 6 for details)

revision=[0000-9999] - sets the revision number, 0000 is the default (only applicable if you are using the Release Concept [AD6]. See Section 6 for details)

stopAfter=[xxx] - number of errors to report before stopping, default 50

dictionary=[x.y] – version of the dictionary to use when validating keywords. See Section 7.3 for details.

{argument} the dataset directory path. Current directory by default.

Example: To verify the contents of the BROWSE directory, without seeing any warnings thrown by the PVV, one would type:

pvv browse –Dwarn=off

---

build - This command builds a previously validated delta release or revision and packs the delta into a tar.gz archive. You will be prompted for a directory in which to place the 'tar.gz' file, and for a filename. By default, the directory will be one above the root directory of the data set, and the filename for the tar.gz file will be [DATASET_ID]_RX_RY.tar.gz where the RX is the Release number and RY is the Revision number being built. The default values will be accepted simply by pressing <enter> without typing a new value. The filename you provide does not need the 'tar.gz' suffix (see examples below).

To complete the transfer of the release or revision to the PSA, the tar.gz file must be uploaded using either the 'pvv upload' command, or manually via any ftp client (recommended).

This command is only applicable if you are using the Release Concept [AD6]. See Section 6 for details.

*[command option]*s for this command are:

warn=[on/off] - turns warning messages on or off. The default is 'on'.

info=[on/off] - turns information messages on or off. The default is 'off'.

release=[0001-9999] - sets the release number, 0001 is the default

revision=[0000-9999] - sets the revision number, 0000 is the default

stopAfter=[xxx] - number of errors to report before stopping, default 50

{argument} the dataset directory path. Current directory by default.

Example 1: To freeze release 0002 of your dataset in the directory /local2/data/psa/MY-DATASET-V1.0 after a successful verify, and build a delta archive from it ready for delivery, one would go to the directory /local2/data/psa/MY-DATASET-V1.0 and type:

pvv build –Drelease=0002

You will then be prompted to enter a directory in which to place the tar.gz file, and a filename. For the purposes of this example, we will <press enter> to accept the default values:

Building DataSet

Please enter the full path of the directory where you want to output the data

[e.g. /local2/data/psa/]:

<press enter to accept default above>

Please enter the file name to be saved as

[e.g. MY-DATASET-V1.0_R2_R0]:

<press enter to accept default above>

This may take a few minutes. Please wait...

MY-DATASET-V1.0_R2_R0.tar.gz built successfully in 303 seconds

Please run 'pvv upload' command to physically transfer the data to the

PSA or use your favorite FTP tool (recommended)

EXECUTION SUCCESSFUL

N.B. This command should be used to automatically freeze a release AND build a delta archive containing files from that release / revision. A user wishing to only freeze a release / revision and then MANUALLY construct a delta archive from this should use the 'pvv freeze' command (see Section 6.2 for details of how to manually create a delta archive).

N.B. The file produced by the 'pvv build' command will contain just those files from the specified release or revision in the data set. For example, if you have a release=0002 which contains just 10 new data files, only those 10 files will be packed into the tar.gz archive when you use the 'pvv build –Drelease=0002' command.

---

calib - verifies label files in the CALIBRATION directory.

*[command option]*s for this command are:

warn=[on/off] - turns warning messages on or off. The default is 'on'.

info=[on/off] - turns information messages on or off. The default is 'off'.

release=[0001-9999] - sets the release number, 0001 is the default (Currently not fully implemented. See Section 6 and Section 9 for details)

revision=[0000-9999] - sets the revision number, 0000 is the default (Currently not fully implemented. See Section 6 and Section 9 for details)

stopAfter=[xxx] - number of errors to report before stopping, default 50

dictionary=[x.y] – version of the dictionary to use when validating keywords. See Section 7.3 for details.

{argument} the dataset directory path. Current directory by default.

Example: To verify the labels in the CALIB directory, one would type:

pvv calib

---

catalog - checks a single catalog file.

*[command option]*s for this command are:

warn=[on/off] - turns warning messages on or off. The default is 'on'.

info=[on/off] - turns information messages on or off. The default is 'off'.

stopAfter=[xxx] - number of errors to report before stopping, default 50

dictionary=[x.y] – version of the dictionary to use when validating keywords. See Section 7.3 for details.

{argument} the path to the catalog file. No default value.

Example: To check the TARGET.CAT file, move into the CATALOG directory and type:

pvv catalog TARGET.CAT

---

catalogs - verifies all the catalog files and their integrity.

*[command option]*s for this command are:

warn=[on/off] - turns warning messages on or off. The default is 'on'.

info=[on/off] - turns information messages on or off. The default is 'off'.

release=[*0001-9999*] - sets the release number, 0001 is the default (only applicable if you are using the Release Concept [AD6]. See Section 6 for details)

revision=[0000-9999] - sets the revision number, 0000 is the default (only applicable if you are using the Release Concept [AD6]. See Section 6 for details)

stopAfter=[xxx] - number of errors to report before stopping, default 50

dictionary=[x.y] – version of the dictionary to use when validating keywords. See Section 7.3 for details.

{argument} the dataset directory path. Current directory by default.

Example: To check all catalogs in your dataset, from the root directory of the dataset type:

pvv catalogs

---

check - this will do a preliminary check on the internal image constructed from the initial scan. This verifies the dataset structure.

*[command option]*s for this command are:

warn=[on/off] - turns warning messages on or off. The default is 'on'.

info=[on/off] - turns information messages on or off. The default is 'off'.

stopAfter=[xxx] - number of errors to report before stopping, default 50

[argument] the dataset directory path. Current directory by default.

Example: To check a dataset with warnings switched off, one would type:

pvv check –Dwarn=off

---

docs - verifies label files in the DOCUMENT directory.

*[command option]*s for this command are:

warn=[on/off] - turns warning messages on or off. The default is 'on'.

info=[on/off] - turns information messages on or off. The default is 'off'.

release=[0001-9999] - sets the release number, 0001 is the default (Currently not fully implemented. See Section 6 and Section 9 for details)

revision=[0000-9999] - sets the revision number, 0000 is the default (Currently not fully implemented. See Section 6 and Section 9 for details)

stopAfter=[xxx] - number of errors to report before stopping, default 50

dictionary=[x.y] – version of the dictionary to use when validating keywords. See Section 7.3 for details.

{argument} the dataset directory path. Current directory by default.

Example: To check the DOCUMENTS directory without seeing any warnings, type:

pvv docs –Dwarn=off

---

freeze - This command allows a user to 'freeze' a dataset in a static 'transferred' state, allowing work to be carried on a new release or a revision of a 'transferred' release.

This command is only applicable if you are using the Release Concept [AD6]. See Section 6 for details.

*[command option]*s for this command are:

warn=[on/off] - turns warning messages on or off. The default is 'on'.

info=[on/off] - turns information messages on or off. The default is 'off'.

release=[0001-9999] - sets the release number, 0001 is the default

revision=[0000-9999] - sets the revision number, 0000 is the default

stopAfter=[xxx] - number of errors to report before stopping, default 50

{argument} the dataset directory path. Current directory by default.

Example: To freeze release 0002 of your dataset after a successful verify, but NOT build a delta archive, one would type:

pvv freeze –Drelease=0002

N.B. Once you have run the 'pvv freeze' command, you will NOT be able to run the 'pvv build' or 'pvv upload' commands on your data set as the XML file will be frozen. You will therefore have to manually build the delta archive containing all files in that release / revision, and then manually transfer that archive to the PSA (see Section 6.2 for details of how to manually create a delta archive).

To automatically create the delta archive using the PVV, you can use the 'pvv build' command described below.

---

gazet - verifies label files in the GAZETTER directory.

*[command option]*s for this command are:

warn=[on/off] - turns warning messages on or off. The default is 'on'.

info=[on/off] - turns information messages on or off. The default is 'off'.

release=[0001-9999] - sets the release number, 0001 is the default (Currently not fully implemented. See Section 6 and Section 9 for details)

revision=[0000-9999] - sets the revision number, 0000 is the default (Currently not fully implemented. See Section 6 and Section 9 for details)

stopAfter=[xxx] - number of errors to report before stopping, default 50

dictionary=[x.y] – version of the dictionary to use when validating keywords. See Section 7.3 for details.

{argument} the dataset directory path. Current directory by default.

Example: To check the gazetter directory, viewing a maximum of 10 errors, one would move to the dataset root directory and type:

pvv gazet –DstopAfter=10

---

help – Returns a list of the possible commands and options.

*[command option]*s for this command are:

dictionary=[x.y] – version of the dictionary to use when validating keywords. See Section 7.3 for details.

{argument} [keyword | object name] will print a description of that odl object as it is defined in the PSA dictionary.

Example 1: To find the definition and description of the INSTRUMENT_NAME keyword in the PSA dictionary, one would type:

pvv help instrument_name

Example 2: To get a full listing of all of the possible PVV commands and their corresponding options and arguments, one would simply type:

pvv help

---

index - constructs the main index files (INDEX.TAB, INDEX.LBL and INDXINFO.TXT) from the 'scan'ned and 'check'ed datasets.

*[command option]*s for this command are:

warn=[on/off] - turns warning messages on or off. The default is 'on'.

info=[on/off] - turns information messages on or off. The default is 'off'.

stopAfter=[xxx] - number of errors to report before stopping, default 50

[argument] the dataset directory path. Current directory by default.

Example: To construct an INDEX.TAB, INDEX.LBL and INDXINFO.TXT file for a dataset, one would type:

pvv index

---

indexes - constructs both the main and browse index files from the 'scan'ned and 'check'ed datasets.

*[command option]*s for this command are:

warn=[on/off] - turns warning messages on or off. The default is 'on'.

info=[on/off] - turns information messages on or off. The default is 'off'.

stopAfter=[xxx] - number of errors to report before stopping, default 50

[argument] the dataset directory path. Current directory by default.

Example: To construct main and browse index files for a dataset, without seeing any warnings thrown by the PVV, one would type:

pvv indexes –Dwarn=off

---

label - checks a single label file. [command option] for this command are:

*[command option]*s for this command are:

warn=[on/off] - turns warning messages on or off. The default is 'on'.

info=[on/off] - turns information messages on or off. The default is 'off'.

dataSetID=[NAME] - set this if you want the dataset ID to be validated in the label file

productID=[NAME] - set this if you want the product ID to be validated in the label file

isData=[true/false] - set this to false if the label file is not a data product label. True by default

stopAfter=[xxx] - number of errors to report before stopping, default 50

dictionary=[x.y] – version of the dictionary to use when validating keywords. See Section 7.3 for details.

[argument] the path to the label file. No default value.

PLEASE NOTE:

- If your data set contains 'FMT' files in a LABEL directory, these will need to be copied into the current directory when testing an individual label file. This is NOT the case when testing *all* label files using the 'pvv labels' command, only if you are testing an individual file using 'pvv label'.
- The PVV automatically detects whether the label you are testing is attached or detached.

Example 1: To check a file MYLABEL.LBL which is not a data product label (e.g. a document label), one would go to the directory containing the label and type:

pvv label MYLABEL.LBL –DisData=false

Example 2: To check an attached label on a product called MYPRODUCT.DAT, and to limit the number of errors shown to 20, one would go to the directory containing the product and type:

pvv label MYPRODUCT.DAT –DstopAfter=20

Example 3: To check a label on a product called MYPRODUCT.DAT, which points to a format file 'MY_FORMAT.FMT' in the LABEL directory, first copy the 'MY_FORMAT.FMT' file into the same directory as the label being tested. Then run the label command as normal:

pvv label MYPRODUCT.DAT

---

labels - verifies all the product label files. It requires that the index file be built first. It will also verify any user created index files that have been 'scan'ned.

*[command option]*s for this command are:

warn=[on/off] - turns warning messages on or off. The default is 'on'.

info=[on/off] - turns information messages on or off. The default is 'off'.

release=[0001-9999] - sets the release number, 0001 is the default (only applicable if you are using the Release Concept [AD6]. See Section 6 for details)

revision=[0000-9999] - sets the revision number, 0000 is the default (only applicable if you are using the Release Concept [AD6]. See Section 6 for details)

stopAfter=[xxx] - number of errors to report before stopping, default 50

dictionary=[x.y] – version of the dictionary to use when validating keywords. See Section 7.3 for details.

{argument} the dataset directory path. Current directory by default.

Example: To check all labels in a dataset, listing up to 100 errors and turning off warnings, one would type:

pvv labels –DstopAfter=100 –Dwarn=off

---

others - verifies label files in the DOCUMENT, CALIBRATION, LABEL, GAZETTER, and BROWSE/QUICKLOOK directories.

*[command option]*s for this command are:

warn=[on/off] - turns warning messages on or off. The default is 'on'.

info=[on/off] - turns information messages on or off. The default is 'off'.

release=[0001-9999] - sets the release number, 0001 is the default (currently checks releases and revisions in the BROWSE directory only – see Sections 6 and 9)

revision=[0000-9999] - sets the revision number, 0000 is the default (currently checks releases and revisions in the BROWSE directory only – see Sections 6 and 9)

stopAfter=[xxx] - number of errors to report before stopping, default 50

dictionary=[x.y] – version of the dictionary to use when validating keywords. See Section 7.3 for details.

{argument} the dataset directory path. Current directory by default.

Example: To validate the label files in the DOCUMENT, CALIBRATION, LABEL, GAZETTER and BROWSE directories, without seeing any warnings thrown by the PVV and limiting the errors to 15, one would type from the root directory:

pvv others –Dwarn=off –DstopAfter=15

---

pack - packs an entire dataset into a tar.gz archive. N.B. You must be in the directory ONE ABOVE YOUR ROOT DIRECTORY for this command to run successfully.

*[command option]*s for this command are:

warn=[on/off] - turns warning messages on or off. The default is 'on'.

info=[on/off] - turns information messages on or off. The default is 'off'.

stopAfter=[xxx] - number of errors to report before stopping, default 50

{argument} the dataset directory path. Current directory by default.

Example: To construct a 'tar.gz' file of your dataset, from one directory above the root directory type:

pvv pack ROOT_DIRECTORY

---

qlindex - constructs the browse index files (BROWSE_INDEX.TAB and BROWSE_INDEX.LBL) from the 'scan'ned and 'check'ed datasets. These files are required if you have browse products in your dataset.

*[command option]*s for this command are:

warn=[on/off] - turns warning messages on or off. The default is 'on'.

info=[on/off] - turns information messages on or off. The default is 'off'.

stopAfter=[xxx] - number of errors to report before stopping, default 50

[argument] the dataset directory path. Current directory by default.

Example: To construct a BROWSE_INDEX.TAB and BROWSE_INDEX.LBL for a dataset, one would type:

pvv qlindex

---

scan - the PVV will scan the dataset and build an internal image which it will use for further verification (see Section 4.1).

*[command option]*s for this command are:

warn=[on/off] - turns warning messages on or off. The default is 'on'.

stopAfter=[xxx] - number of errors to report before stopping, default 50

release=[*0001-9999*] - sets the release number, 0001 is the default (only applicable if you are using the Release Concept [AD6]. See Section 6 for details)

revision=[0000-9999] - sets the revision number, 0000 is the default (only applicable if you are using the Release Concept [AD6]. See Section 6 for details)

dictionary=[x.y] – version of the dictionary to use when validating keywords. See Section 7.3 for details.

[argument] the dataset directory path. Current directory by default.

Example: To scan a dataset and show the first 100 warning or error messages, one would type:

pvv scan –DstopAfter=100

---

status – This command compiles a brief report of the release/revision status of the dataset. By using the –Drelease and/or –Drevision options you can retrieve detailed information about the products in each delivery, and the status of the testing of each release or revision.

*[command option]*s for this command are:

warn=[on/off] - turns warning messages on or off. The default is 'on'.

info=[on/off] - turns information messages on or off. The default is 'off'.

release=[0001-9999] - sets the release number, 0001 is the default (only applicable if you are using the Release Concept [AD6]. See Section 6 for details)

revision=[0000-9999] - sets the revision number, 0000 is the default (only applicable if you are using the Release Concept [AD6]. See Section 6 for details)

{argument} the dataset directory path. Current directory by default.

Example: To check the status of your release 0002, revision0001, and to see if products are successfully scanned, verified or even transferred, you would type:

pvv status –Drelease=0002 –Drevision=0001

---

unpack - unpacks a tar.gz dataset archive.

*[command option]*s for this command are:

warn=[on/off] - turns warning messages on or off. The default is 'on'.

info=[on/off] - turns information messages on or off. The default is 'off'.

stopAfter=[xxx] - number of errors to report before stopping, default 50

{argument} the dataset archive file (tar.gz). No default value.

Example: To unpack a packed 'tar.gz' dataset called 'MY_GLORIOUS_DATASET.tar.gz', from the directory containing the file, one would type:

pvv unpack MY_GLORIOUS_DATASET.tar.gz

---

update - This command automatically updates the PVV software to a newer version, unpacking the new 'tar.gz' file and updating the environment variables.

*[command option]*s for this command are:

warn=[on/off] - turns warning messages on or off. The default is 'on'.

stopAfter=[xxx] - number of errors to report before stopping, default 50

{argument} the installation directory path. Current directory by default.

Example: To update your PVV installation, place the new 'PVV_XX.tar.gz' file in the pvv installation directory and from there, type:

pvv update PVV_XX.tar.gz

---

upload – This command uploads a given file to the PSA upload ftp area.

*[command option]*s for this command are:

no options are available for this command

{argument} The full path and filename to upload.

Example: To upload a tar.gz archive produced using the 'pvv build' command or any tar.gz file containing your data set ready for delivery, type:

pvv upload FULL_PATH/MY_FILENAME.tar.gz

N.B. It is not recommended to use this command to transfer your data to the PSA. Instead, it is strongly recommended you use your preferred FTP client and transfer the file(s) manually.
You will NOT be able to run 'pvv upload' if you have used the 'pvv freeze' command on a release or revision. Please use your preferred ftp client to upload your release / revision instead.

---

verify - This is the absolute PVV test on a dataset. It will execute sequentially the commands 'check', 'catalogs', 'labels', and 'others', as well as carrying out an integrity check between label and catalog files.

*[command option]*s for this command are:

warn=[on/off] - turns warning messages on or off. The default is 'on'.

info=[on/off] - turns information messages on or off. The default is 'off'.

release=[0001-9999] - sets the release number, 0001 is the default (only applicable if you are using the Release Concept [AD6]. See Section 6 for details)

revision=[0000-9999] - sets the revision number, 0000 is the default (only applicable if you are using the Release Concept [AD6]. See Section 6 for details)

stopAfter=[xxx] - number of errors to report before stopping, default 50

dictionary=[x.y] – version of the dictionary to use when validating keywords. See Section 7.3 for details.

{argument} the dataset directory path. Current directory by default.

Example: To verify your dataset ready for delivery but limiting the errors thrown to 30, one would type:

pvv verify –DstopAfter=30

---

version – This simply displays the version of the PVV software currently running.

*[command option]*s for this command are:

no options are available for this command

{argument} no arguments are available for this command.

Example: To display the version of the PVV being used, type:

pvv version

---

xlabels - verifies label files in the LABEL directory.

*[command option]*s for this command are:

warn=[on/off] - turns warning messages on or off. The default is 'on'.

info=[on/off] - turns information messages on or off. The default is 'off'.

release=[0001-9999] - sets the release number, 0001 is the default (Currently not fully implemented. See Section 6 and Section 9 for details).

revision=[0000-9999] - sets the revision number, 0000 is the default (Currently not fully implemented. See Section 6 and Section 9 for details).

stopAfter=[xxx] - number of errors to report before stopping, default 50

dictionary=[x.y] – version of the dictionary to use when validating keywords. See Section 7.3 for details.

{argument} the dataset directory path. Current directory by default.

Example: To verify all labels in the LABEL directory, from the root directory type:

pvv xlabels

N.B. You can use the $PVV_HOME/ext/pvv.properties file to set properties permanently and thus configure your PVV tool without having to constantly flag them on the command line. These take the form of 'propertyName = propertyValue', e.g. adding 'stopAfter=100' to the file would run all commands to a maximum of 100 errors.

5. Quick Step-by-Step Guide to Validating a Dataset

The steps required to check one dataset and transfer it to the PSA are:

- Go to the dataset root directory.
- Run 'pvv scan'. This creates an xml file in the directory, with an image of the dataset.
- If your dataset has no tables in the INDEX directories, pvv scan will complain. Run 'pvv index', and then re-run 'pvv scan' to build the dataset image.
- Run 'pvv catalogs' to check all of the catalogues.
- Run 'pvv labels' to check all of the labels.
- Run 'pvv others' to check the DOCUMENT, SOFTWARE, etc. directories.
- Run 'pvv verify', which will complete all the procedures listed above again, but will additionally check for consistency between labels, catalogues, etc.
- If you are using the Release Concept [AD6] and are preparing a new release or revision:

  Either:

  Run the 'pvv build' command to create the delta archive.

  Or:

  Run the 'pvv freeze' command to freeze the release and then manually construct your delta archive for delivery (see Section 6.2 for details of how to manually create a delta archive)
- If you are not using the Release Concept, and want to pack the entire data set to deliver to the PSA:

  cd .. and then type 'pvv pack ROOT_DIRECTORY_NAME'. This will pack your dataset into a tar.gz file ready to transfer to the PSA
- Use your preferred ftp client or 'pvv upload' to transfer your dataset to the PSA.

N.B. If at any time you need to see a list of commands and options, use: '**pvv help**'

TIP: When running PVV, a lot of warnings could be sent to the terminal, and the user can get confused. You can switch the warnings off by using the '-Dwarn=off' option at the command line. For example, you can check the labels running 'pvv labels –Dwarn=off'.

For checking an individual label file, the easiest way is go to the DATA directory where the label resides, and run 'pvv label [label.lbl] –Dwarn=off', where [label.lbl] is the label filename. The same applies for catalogs: 'pvv catalog [catalog.cat] –Dwarn=off'.

6. Guide to Using the PVV with the Release Concept

This Section describes how to build your dataset and use the PVV to test your dataset if you are using the Release Concept described in [AD6]. If you do not use this concept for your deliveries, then this section is not applicable to you.

There are some important rules you must follow in order for the PVV to work successfully while you build your dataset using the Release Concept (for full details, see the Release Concept document [AD6]):

- You must construct your dataset *incrementally*. In other words, if you already start testing using data from several Releases, you **must** first construct the dataset for Release=0001 and PVV it, then add Release=002 and PVV it etc., until the complete dataset is tested, up to the current Release.
- All Revisions made to a single Release are cumulative. In other words:
    i. In Release0001, Revision0000 of a dataset: Product X is updated and delivered as Release0001 Revision0001
    ii. In Release0001, Revision0001 of the dataset: later updates to different data (e.g Product_Y or Document_Z). Although this is the first revision of these specific products, it is the second revision to this Release, and so these should be delivered as Release0001, Revision0002.
- If a new document or 'non-DATA' file is to be delivered (i.e. a file not in the DATA, CALIB or BROWSE directory), it should be delivered as part of the latest Release. For example, if you are ready to deliver Release 0004 of a dataset, and you have also decided to include a new document to the DOCUMENT directory, this document should be delivered as Release 0004 as well, even though it is the first delivery of this document. This way, the document can be checked as part of the overall release using the PVV.
- If you need to deliver a new 'non-DATA' file *between* the routine release deliveries, it should be delivered as a Revision of the most recent Release. For example: you have delivered Release 0004 Revision 0000, and you now want to add a new file to the dataset. You can do this by delivering this product as Release 0004 Revision 0001.

  [N.B. The DOCUMENTS directory on the online database will not take account of release or revision IDs in the labels. The latest copies of ALL documents will always be delivered to the end user from the online PSA archive. Using the Release ID in the above example simply allows the PVV to check the file as part of the overall delivery].

IMPORTANT NOTE:
The XML data set image file contains the creation time of the individual files of your dataset as time stamps after the 'pvv freeze' or 'pvv build' command (see below). In case of moving data products in and out of a file system, the user therefore has to maintain the creation times of all the files involved.
On Solaris e.g. the 'mv' command preserves the creation time of a file and should be used instead of making a copy using the 'cp' command.

6.1 PVV commands to use when using the Release Concept

The following step-by-step procedure should be followed when testing your Releases using the PVV.

The PVV is designed to test a dataset as it grows from release to release. *It is therefore very important that you run the PVV on **each and every** release and that you **build up your dataset one release at a time***.

Please ensure that you deliver us ALL releases and revisions. If you are testing release 1 revision 1 and then release 1 revision 2 before your next delivery, please make sure that BOTH revisions are sent to us as separate files.

6.1.1 Checking your first Release [Release 0001, Revision 0000]

The steps required to check the first release [Release0001 Revision0000] and transfer it to the PSA are:

- Go to the dataset root directory.
- Run 'pvv scan –Drelease=1'. This creates an xml file in the directory, with an image of the entire dataset.
- If your dataset has no tables in the INDEX directories, pvv scan will complain. Run 'pvv index', and then re-run 'pvv scan –Drelease=1' to build the dataset image again.
- Run 'pvv catalogs' to check all of the catalogues.
- Run 'pvv labels –Drelease=1' to check all of the labels.
- Run 'pvv others' to check the DOCUMENT, SOFTWARE, etc. directories.
- Run 'pvv verify –Drelease=1', which will complete all the procedures listed above again, but will additionally check for consistency between labels, catalogues, etc.
- ***If you want to automatically create a delta archive of your release for delivery:***
  - Run 'pvv build –Drelease=1'. This will freeze your dataset and generate a 'tar.gz' file containing all data in release1 ready for delivery to the PSA. Freezing your XML file for Release1 will allow you to build new releases on top of this.
- ***If you want to freeze your release and then MANUALLY create a delta archive of your release for delivery (Section 6.2):***
  - Run 'pvv freeze –Drelease=1'. This will freeze release 1 of your dataset. After this you will have to manually construct a delta archive containing those files relevant to the release and manually transfer it to the PSA. Freezing your XML file for Release1 will allow you to build new releases on top of this.
- Run 'pvv status –Drelease=1' to get a report on the status of your first release.
- Either run 'pvv upload' or use your preferred ftp client to transfer your dataset to the PSA.

6.1.2 Checking your second Release [Release 0002, Revision 0000]

The steps required to check the second release [Release0002 Revision0000] and transfer it to the PSA are:

- Go to the dataset root directory.
- Run 'pvv scan –Drelease=2'. This builds upon the existing xml file from release1, adding those files that have been added to the dataset in this release.

- If you have not already done so, you need to re-build the index tables in the INDEX directory. Remove the old files and run 'pvv index', then re-run 'pvv scan –Drelease=2' to build the dataset image again.
- Run 'pvv labels –Drelease=2' to check all of the labels added in Release2.
- If you have included any new non-DATA files (e.g. software, documents etc) then you should also run 'pvv others –Drelease=2'.
- Run 'pvv verify –Drelease=2', which will complete all the procedures listed above again, but will additionally check for consistency between labels, catalogues, etc. for all products added in Release2.
- ***If you want to automatically create a delta archive of your release for delivery:***
  - Run 'pvv build –Drelease=2'. This will freeze your dataset and will pack ***only those products / labels that are part of release2*** into a 'tar.gz' file ready for delivery to the PSA.
- ***If you want to freeze your release and then MANUALLY create a delta archive of your release for delivery (Section 6.2):***
  - Run 'pvv freeze –Drelease=2'. This will freeze release2 of your dataset. After this you will have to manually construct a delta archive containing ***only those files relevant to release2*** and manually transfer it to the PSA.
- Run 'pvv status –Drelease=2' to get a report on the status of your second release.
- Run 'pvv upload' or use your preferred ftp client to transfer your dataset to the PSA.

6.1.3 Making a Revision to a Release [e.g. Release0001 Revision0001]

The steps required to check a revision to a release [e.g. Release0001 Revision0001] and transfer it to the PSA are:

- Go to the dataset root directory.
- Run 'pvv scan –Drelease=1 –Drevision=1'. This builds upon the existing xml file from all releases, changing the details of the file(s) that have been revised from Release1.
- If you have not already done so, you need to re-build the index tables in the INDEX directory. Remove the old files and run 'pvv index', then re-run 'pvv scan –Drelease=1 –Drevision=1' to build the dataset image again.
- Run 'pvv labels –Drelease=1 –Drevision=1' to check all of the labels in this revision.
- Run 'pvv verify –Drelease=1 –Drevision=1', which will complete all the procedures listed above again, but will additionally check for consistency between labels, catalogues, etc. for all products in this revision.
- ***If you want to automatically create a delta archive of your revision for delivery:***
  - Run 'pvv build –Drelease=1 –Drevision=1'. This will freeze your your XML file and will build a 'tar.gz' file containing only those files that are part of your release1 revision1, ready for transfer to the PSA.

- ***If you want to freeze your release and then MANUALLY create a delta archive of your release for delivery (Section 6.2):***
  - Run 'pvv freeze –Drelease=1 –Drevision=1'. This will freeze your XML file. After this you will have to manually construct a delta archive containing the release1 revision1 data and manually transfer it to the PSA.
- Run 'pvv status –Drelease=1 –Drevision=1' to get a report on this revision.
- Run 'pvv upload' or use your preferred ftp client to transfer your dataset to the PSA.

N.B. At any time, you can run 'pvv status' to get a report on the release and revision status of your dataset. To get an update on the status of a given release or revision, you can type 'pvv status –Drelease=x –Drevision=y' where 'x' and 'y' represent the release and revision numbers you are interested in.

IMPORTANT NOTE:
The XML data set image file contains the creation time of the individual files of your dataset as time stamps after the 'pvv freeze' or 'pvv build' command (see below). In case of moving data products in and out of a file system, the user therefore has to maintain the creation times of all the files involved.
On Solaris e.g. the 'mv' command preserves the creation time of a file and should be used instead of making a copy using the 'cp' command.

6.2 Manually constructing a delta archive containing a given release / revision

If you are using the release concept [AD6] and have decided not to use the 'pvv build' command to automatically generate a delta archive, then you will have to freeze your archive using the 'pvv freeze' command and then manually construct your own delta archive to transfer to the PSA. In order for the PSA to successfully ingest this into your existing data set structure, you will need to ensure that the correct files are in the delta archive you generate. The following sections describe the requirements for this.

6.2.1 Manually constructing your first Release [Release0001 Revision0000]

For your first release, you will need to pack the entire data set together into a single file (e.g. tar.gz or tar.bz2). This will then have to be manually transferred to the PSA using your preferred ftp client.

6.2.2 Manually constructing your second Release [Release0002 Revision0000]

For your second release, you will need to pack the following files together into a single file (e.g. tar.gz or tar.bz2):

- All products and product labels from Release2
- **All** *.TXT files (e.g. CALINFO.TXT, DOCINFO.TXT etc)
- The **entire** INDEX directory (even if nothing has changed)
- The **entire** EXTRAS directory (even if nothing has changed)

You should maintain the directory structure for all files packed together. This delta archive will then have to be manually transferred to the PSA using your preferred ftp client.

6.2.3 Manually constructing a Revision [e.g. Release0001 Revision0001]

For any revision to a release, you will need to pack the following files together into a single file (e.g. tar.gz or tar.bz2):

- All products and product labels from Release1 Revision1
- **All** *.TXT files (e.g. CALINFO.TXT, DOCINFO.TXT etc)
- The **entire** INDEX directory (even if nothing has changed)
- The **entire** EXTRAS directory (even if nothing has changed)

You should maintain the directory structure for all files packed together. This delta archive will then have to be manually transferred to the PSA using your preferred ftp client.

7. Implementation of the PSA and Mission Specific Dictionaries

The PVV now includes a check through both a 'PSA Data Dictionary' and new Mission Specific dictionaries, all of which are controlled by the PSA archive team. The new dictionaries allow full validation of datasets complying with the PDS standards, which is not possible using the PDS Dictionary alone.

Where appropriate, PVV will now check keyword values against a list of permitted values in the dictionary (e.g. values for INSRUMENT_ID will be checked against a list in the PSA Dictionary). If a value is used that is not in the dictionary, an error will be thrown. In order to solve this, the given value will have to be added to the list in the PSA dictionary.

Changes to the dictionaries can only be made by the PSA representatives, so all requests for updates / additions must be made via your Mission Archive Manager (see below). Once agreed, changes to the dictionaries can be implemented immediately, allowing for testing of datasets and labels to continue without delay.

7.1 The 'PSA Dictionary'

The PSA Dictionary is an extension of the PDS Data Dictionary that allows the PVV to fully validate keywords, and (where applicable) their values and their type (i.e. whether or not enumeration is permitted). There are several reasons for this new dictionary implementation:

- All keywords have an additional parameter that will tell the PVV if enumerated values are permitted.

  [N.B. If you come across any keyword during testing that you feel should be enumerated, but is currently not permitted, please contact your Mission Archive Manager at the PSA to discuss and, if appropriate, he/she will update this keyword]

- Objects in the PDS Dictionary that have the optional parameter of 'PSDD' permit the use of any keyword in the PDS Dictionary. Rather than permitting all values, the PVV will now check against a list of optional values in the PSA dictionary. Therefore, if any situations arise

where keywords are included in objects due to the 'PSDD' option being present in the PDS Dictionary, the corresponding keyword must be added to the optional list of the object in the PSA Dictionary.

[N.B. If you are using any keywords in an object as a result of the 'PSDD' option being present in the PDS Dictionary, please consult your Mission Archive Manager at the PSA with a list of these, and, if appropriate, the PSA Dictionary will be updated]

- The SAMPLE, BAND and LINE prefixes to keywords that are used in the qube object are not currently in the PDS Data Dictionary. These keywords have been added to the PSA Data Dictionary to allow successful checking of these products.
- Some keywords are only allowed to have values that are listed in the dictionary (these keywords have a list of permitted values in the PDS Dictionary). If a value that is not in the dictionary is used, you will get an 'Illegal keyword value' error from the PVV. To remove the error, this value will have to be added to the list in the dictionary.

  [N.B. Should you come across these errors, first check that you are using a sensible value, and that no value already exists in the PDS Dictionary that would fit your needs. If there is not a suitable value available, then consult with your Mission Archive Manager to ask for the value to be added to the PSA Dictionary].

7.2 Mission Specific Dictionaries

The PSA have now implemented Mission Specific Dictionaries to the PVV. These are dictionaries containing new keywords that are deemed only of interest to the specific mission / instrument for which they are used. All mission specific keywords will be added to a mission specific dictionary, and any occurrences of these keywords in a label will then be validated against this.

N.B. Mission specific keywords should all have a 'MISSION_ID:' prefix. For example 'MEX: KEYWORD = VALUE' or 'S1: KEYWORD = VALUE' for Mars Express and SMART-1 respectively.

You should communicate with your Mission Archive Manager to decide upon any Mission Specific Keywords you want, and the agreed keywords and definitions / values will be added to the applicable dictionary.

7.3 Dictionary Versioning

The PSA have now implemented Dictionary versioning in the PVV in order to allow data sets ingested with old dictionary values to remain compliant should we require ire-ingestion. In addition, this allows us to keep the PSA dictionary in line with the most recent version of the PDS Dictionary without forcing older data sets to become non-compliant. The PSA Dictionary will be merged with the PDS dictionary whenever significant updates have been made. This document will be updated with details of new dictionary versions whenever they are made available.

During a 'scan' the PVV will stamp the XML file with the version of the PSA Dictionary that is being used to validate the data set. A user can force a specific dictionary version to be used using the '-Ddictionary=x.y' flag when running the PVV, where x.y is the version number of the dictionary to use. If no dictionary flag is given for a new data set, the latest version will be used by default. If a non-existing dictionary version is given, then the most recent will be used by default. At the time of writing, the following Dictionary versions are implemented:

1.0 – Released 10-02-2005

1.1 – Released 27-07-2005

1.5 – Released 19-10-2005

In some cases, when a data set has been initially tested using an old Dictionary version, a user can be presented with an 'Illegal value for keyword' error even after the value listed has been entered in the dictionary. For example:

```
[VERIFY ][E][07.020][0100] Illegal value for keyword MISSION_PHASE_NAME
 (DATA/NPI_EDR_L1B_AUG2005/NPINORM20052252023C_ACC01.LBL, 20)
   Keyword MISSION_PHASE_NAME cannot have value MR Phase 7
```

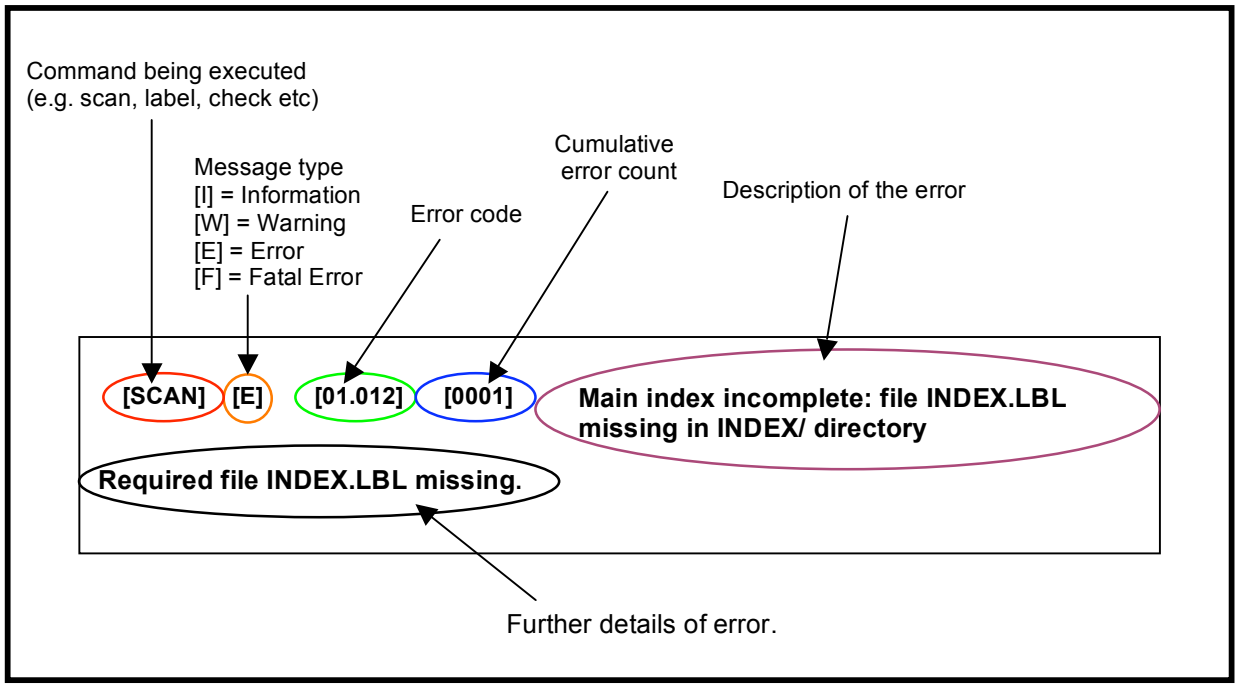
The above error occurred even though the "MR Phase 7" value was available for use in the dictionary. This is because the dictionary version in the PDSVOLUME.XML file is the older version and any updates made since the latest dictionary was released are not found during validation. In these cases, the user should run the PVV with the flag –Ddictionary=x.y where x.y represents the latest version of the PSA Dictionary.

8. Errors – Formatting and Types

The PVV throws messages with separate fields for each of the following:

- Command: The command being executed by the user
- Message Type: There are four types of messages thrown by the PVV
  - [I] Information – the user does not need to act on these messages – they are for information only. By default, information messages are switched off when running the PVV. They can be switched on using the command option '–Dinfo=on'.
  - [W] Warning – these are warnings to the user of potential problems detected by the PVV. Warnings will not cause a command to fail. By default, warnings are switched on when running the PVV. They can be switched off using the command option '-Dwarn=off'.
  - [E] Error – These are errors detected by the PVV and require fixes to your dataset. Errors are always on when running the PVV. Individual errors will not cause the PVV to fail. However, if a large number of errors are accumulated, the execution will fail, and some errors will need to be fixed before running the command again.
  - [F] Fatal Error – This is a critical error that has forced the PVV to fail immediately. The error will need to be fixed before the command can be executed successfully. Fatal errors are always switched on.
- Error Code: The error number. Definitions of the errors and their corresponding codes are listed in [AD5]
- Number of Messages: This is simply a count of the messages thrown during an execution
- Description of Error: This is a text field that provides a simple description of the error and where it was detected (e.g. file name and line number). If no explicit file name is given, the error is most likely to be found at the line number given in the '.PDSVOLUME.XML' file generated by the PVV during the scan.
- Further Details: Additional details of the error as required.

An example of a message from the PVV is shown below, with each of the above fields highlighted.



Some further examples of real errors from the PVV are shown below:

[LABELS][W][09.005][0006] File referenced by pointer TABLE not being checked
(DATA/MARS1/MARS.LBL, 14)
External file(s) of pointer TABLE is(are) not being checked by the PVV
in this operation

EXECUTION SUCCESSFUL

In this example, a warning is thrown when running the 'pvv labels' command. The warning states that the pointer to the table is not being checked by this command, so this would have to be checked another way. Warnings inform the user of potential problems, and do not always need to be acted upon (note that the execution was successful despite this message being thrown). If the number of warnings thrown becomes cumbersome, they can be switched off using the '-Dwarn=off' option.

[CATALOGS][E][02.031][0002] Required object SOFTWARE is missing in parent
CATALOG (./VOLDESC.CAT, 34)
Value of required object VOLUME:CATALOG:SOFTWARE is missing

EXECUTION FAILED

In this example, the 'pvv catalogs' command was run, and the PVV has detected a missing SOFT.CAT file pointer, which is required by the PDS Standards. The PVV identifies that this file pointer is missing, and should be present on line 34 of this particular VOLDESC.CAT. If no software catalog is present, the 'NULL' value can be given.

[SCAN][E][01.027][0038] PRODUCT_ID keyword in label file
DOCUMENT/MY_LABEL.LBL missing
PRODUCT_ID missing in label file DOCUMENT/MY_LABEL.LBL

EXECUTION FAILED

In this example, the 'pvv scan' command was run, and the PVV has detected a missing PRODUCT_ID keyword in the label file 'MY_LABEL.LBL' in the DOCUMENT directory.

[CHECK][F][03.005][0001] PDS IDs must be 40 characters long and formed of
upper case letters, numbers, '_', '-', '/', or '.'.
Error on line 2: cvc-pattern-valid: Value '' is not facet-valid with
respect to pattern '[A-Zv0-9\-_\./]{1,40}' for type 'IDType'.

EXECUTION FAILED

In this example, the 'pvv check' command was run, and the PVV detected a fatal error in line 2 of the '.PDSVOLUME.XML' file (N.B. If a line number is given with no file specified, you should always look in the XML file). Looking in the XML file shows that this line contains the various IDs from the datset (e.g. DATSET_ID, INSTRUMENT_ID etc.). One of these values is missing or incorrect. The user should locate the error in the .PDSVOLUME.XML file, and correct the relevant ID in the corresponding catalog file (the PVV reads the IDs from their corresponing catalog files e.g. DATASET_ID is read from the DATASET.CAT etc.). Remember that once the correction is made, a new 'pvv scan' is required to generate a new XML file before testing can continue.

[VERIFY][E][07.020][0006] Illegal value for keyword INSTRUMENT_ID
(CATALOG/DATASET.CAT, 82)
Keyword INSTRUMENT_ID cannot have value MY_INSTRUMENT

EXECUTION FAILED

In this example, an error is thrown when running the 'pvv verify' command. The error states that the value 'MY_INSTRUMENT' is not permitted for the INSTRUMENT_ID keyword. To solve this, the new value will have to be added to the PSA Dictionary (see Section 7).

A complete list of the errors thrown by the PVV and the corresponding error codes can be found in [AD5]. In case of problems or queries with the errors thrown, please refer to your mission archive manager.

9. Future Releases – Features Yet To Be Implemented

Below are listed some of the features that have yet to be implemented in the PVV, and will be present in future releases of the tool. Users should be aware of these limitations when testing their datasets.

- Currently, the PVV does not handle the use of units correctly. Full validation of units used in a keyword value will be implemented in a future version.
- The number of errors and warnings thrown by the PVV can be large if there are identical errors in all products and there are hundreds of products in a data set. It is possible that a more controlled error manager will be looked into in the long term to try and smooth this.

  N.B. It is still vital that you use the correct release / revision IDs in the labels for these directories, even though they are not separated out by these pvv commands – the 'pvv verify' command requires the correct IDs and ***will*** *check these values for each release / revision*.

10. Typical Problems and Solutions (FAQ)

- The PVV in Windows XP is crashing without finding my directory.
  - Ensure that your dataset is in a directory with NO SPACES in the pathname, and run the command again.
- The PVV in Windows XP always throws an error 'Unable to locate tools.jar. Expected to find it in c:/program files… etc.'
  - This error is thrown if a user has only got the 'JRE' (Java Runtime Environment) installed, and not the full 'JDK' (Java Development Kit). *For all normal use, the PVV will work perfectly with just the JRE, so this error can be ignored and the PVV will still run*. The full JDK installation is only required if debugging is necessary, and does not need to be installed for normal day-to-day PVV usage.
- The PVV cannot connect to the Dictionary server.
  - Are you using a proxy server? If so, you need to edit the $PVV_HOME/ext/pvv.properties file and include the details of your proxy server.
- The PVV is crashing with a java.lang.OutOfMemoryError message.
  - You need to edit the $PVV_HOME/ext/pvv.properties file and increase the amount of memory allocated to the PVV in the relevant section called 'Run-Time Memory Settings'. By default, this is set to 64 MB, but

you can increase it to 128Mb or more as required. Although this can slow the operations down, it should help solve any memory problems.

- I am running 'pvv label' on one of my files, and the PVV complains about a missing 'FMT' file, but I am using a structure pointer to this file in my label, and it is in the LABEL directory.
    - Because 'pvv label' is only looking at a single file, it does not use the XML file to look at the mapping of your directory structure, and therefore cannot find the FMT file in the LABEL directory. For this operation you will have to temporarily copy a version of the FMT file into the same directory as the label being tested. Please note that this is not the case for 'pvv labels' as this operation then uses the XML image to look at the full structure of your data set.
- The PVV is throwing so many warnings that I cannot filter out the errors easily
    - Run the command again with the option '-Dwarn=off' to switch the warnings off
- The PVV has thrown several errors which refer to 'Error on line x' but it does not specify which file to look in. Where is the error?
    - The line is in the '.PDSVOLUME.XML' file. You need to look in the XML file to locate the error, fix it in the dataset and re-run the scan to build a new corrected XML file.
- The PVV is complaining about my mission specific keywords, which I have placed in a GROUP=MISSION.
    - Mission specific keywords should all have a 'MISSION_ID:' prefix. For example 'MEX: KEYWORD = VALUE' or 'S1: KEYWORD = VALUE' for Mars Express and SMART-1 respectively. These keywords should NOT be placed in a 'GROUP=MISSION' object. If the PVV complains about an 'illegal keyword' for one of your mission specific keywords, please consult your mission archive manager and, if reasonable, he or she will add it to the mission specific dictionary.
- The PVV does not accept one of my keywords in a parent object, although the 'PSDD' option is present in the PDS Dictionary, which implies I am free to use any keyword.
    - The PVV uses two dictionaries, the PDS Dictionary, and the 'PSA Dictionary'. If any situations arise where keywords are included in objects due to the 'PSDD' option being present in the PDS Dictionary, the corresponding keyword will be added to the optional list of the object in the PSA Dictionary, which the PVV will also check.

      N.B. All instances in which a keyword is used outside of those listed in the required or optional set because of the presence of the PSDD option, MUST be reported to your mission archive manager, who will confirm that this is ok, and then update the PSA Dictionary.
- The PVV is complaining about 'enumerated values' not being permitted for a certain keyword, which in fact requires enumeration.
    - The PVV checks against the PSA dictionary to see if a keyword can be enumerated. If you come across a keyword you feel should be enumerated, but for which this is not currently allowed, you should communicate with your Mission Archive Manager and ask for this keyword to be updated in the PSA Dictionary. .

- The PVV is complaining about the units I am using on my keywords, but I cannot see any problems.
    - The PVV does not yet correctly handle units. This will be implemented for a future version of the PVV.
- I would like to transfer my fully tested (and perfectly PDS compliant!) dataset to the PSA. How can I use the PVV to do this directly?
    - If you are not using the Release Concept [AD6] (Section 6), you can first run the 'pvv pack' command from one directory above your rott directory. This will produce a 'tar.gz' file containing your entire dataset. This can then be transferred to the PSA using your preferred ftp client, or the 'pvv upload' command.
    - If you are using the Release Concept [AD6] (Section 6), you can first execute the 'pvv build –Drelease=x –Drevision=y' command, where x and y are the release and revision number respectively. This will generate a 'tar.gz' file containing only those files relevant to the specified release and revision. This 'tar.gz' file can then be transferred to the PSA using your preferred ftp client, or the 'pvv upload' command.
- I have several errors of the same type in my dataset simply because I have hundreds of products with the same single error. I would like to move further into the testing to identify other errors, but the PVV will crash after a limited number of errors. Can I adjust this?
    - By Default the PVV will fail an executed command after 50 errors are received. To adjust this value, just run the command you want with the option '-DstopAfter=xxx' where 'xxx' is the number of errors you would like the PVV to accept before failing. Example: to run pvv scan listing 200 errors, simply type '*pvv scan –DstopAfter=200*'.
- I have got a dataset ready to deliver, but the 'pvv pack' command seems to generate an empty tar.gz file.
    - You need to run the 'pvv pack' command form the directory *above* your root directory.
- I have run through a completely successful test on my dataset, but I know there are inconsistencies and gaps in the dataset. How thorough is the PVV and what should I check manually to back up the software tests before I deliver my dataset?
    - The PVV does not yet check the content of the SOFTWARE directory, or the DOCUMENTS, CALIBRATION and GEOMETRY directory beyond the fact that every file requires a label, and that every directory requires a corresponding INFO.TXT file. You should also be aware that mission specific keywords are not yet verified.
- I am trying to re-run a test on my dataset, but I am getting the following sort of error:

  [VERIFY][F][09.018][0001] Dataset already transfered to PSA.

  Dataset MY_DATASET release/revision 0001/0000 has

  already been transfered to the PSA. If you wish to make an

  update of this dataset, please use the release concept.

    - You have already run the 'pvv build' command on this release/revision of the dataset, so the XML file is frozen to ensure that no changes can

be made to the delivery. If you want to make changes and re-test the dataset, you should make a Revision to the dataset (see Section 6).

- I am trying to upload my file to the PSA, but every time I run the 'pvv upload' command, it says that '0 files are transferred.
  - You must specify the *full* path of the file to upload, not just the filename.
- I am getting an 'Illegal value for keyword' error, even though I can see the value I am using is in the Dictionary.
  - The dictionary version you are using is old. Run the PVV with the –Ddictionary=x.y flag, where x.y is the latest version of the PSA Dictionary (see Section 7.3).

In some cases, when a data set has been initially tested using an old Dictionary version, a user can be presented with an 'Illegal value for keyword' error even after the value listed has been entered in the dictionary. For example:

[VERIFY][E][07.020][0100] Illegal value for keyword MISSION_PHASE_NAME
(DATA/NPI_EDR_L1B_AUG2005/NPINORM20052252023C_ACC01.LBL, 20)
Keyword MISSION_PHASE_NAME cannot have value MR Phase 7

The above error occurred even though the "MR Phase 7" value was available for use in the dictionary. This is because the dictionary version in the PDSVOLUME.XML file is the older version and any updates made since the latest dictionary was released are not found during validation. In these cases, the user should run the PVV with the flag –Ddictionary=x.y where x.y represents the latest version of the PSA Dictionary.

If further errors or complications arise that are not clarified above, please consult your mission archive manager for support. For problems with the software, please consult the PVV support helpline outlined below.

11. PVV User Support Helpline

In case of problems with the PVV software, please contact your mission archive scientist. For general PSA issues, contact psahelp@rssd.esa.int