Labeler and Ruleset Processing

Todd King, Steven Joy, Joe Mafi, Erin Means

Presented at the PDS Technical Session – July 2003

The Need

- Help our data providers help us.
  - Allow novice users to generate quality labels for data products.
- Convert a large number of legacy labels to current standards.
  - Augment existing labels for use in the new on-line data system.
- Consolidate and standardize the tools used by our data engineers.
- Standardize our best approaches for generating labels.

Goals

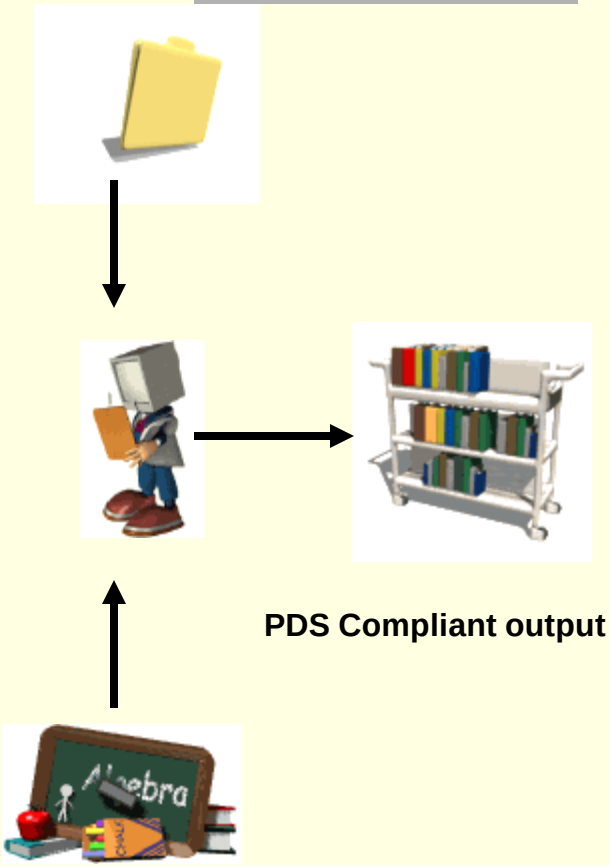
- To promote the delivery of PDS compliant products from missions and data providers.
- To be able to provide tools that data providers can use in-house and on their platform of choice to create labels for data products.
- To enable a PDS data engineer to design a label template and ruleset for the data provider.
- To have the ability to “plug-in” a service for new or unique applications.
- To be able to perform “upgrades” to existing data holdings.

The Approach

- Simple
  - Command line applications – no fancy interfaces. Mnemonic arguments.
- Portable
  - Core software written in Java. Real push to have all components written in Java.
- Extensible
  - Ability to add capabilities without modifying the core components.
- Minimal Restrictions
  - Extensions can be written in any language.

Framework

- Templates
  - A PDS label with unknown values set as variables.
  - Variables are replaced with information collected by the ruleset.
- Rulesets
  - Instructions on how to collect information about a specific data item.
  - Which template to use and where to write the resulting label.
- Plug-ins
  - Mini-applications, written in any language, that perform an external service (i.e, geometry processor, description formatter)
  - Return rulesets (values) to be processed and merged with the current ruleset.



Ruleset Language

A tag based language with flow control. Directives include:

$variable = value :: Define a ***variable*** and set to ***value***.

<RUN command> :: Run a ***command*** and processes that output of the as a ruleset.

<IF condition> <ELSEIF condition> <ELSE> </IF> :: Branching

<INCLUDE file> :: Load and process ruleset in ***file***.

<IGNORE> :: Stop processing file – do not produce output.

<TEMPLATE file> :: Use the ***file*** as the label template

<OPTION name value> :: Set ***name*** to specified ***value***.

<OUTPUT file> :: When generating output, write to ***file*** (default: *base*.lbl)

<MESSAGE text> :: Write ***text*** to display.

<ABORT> :: Stop all processing.

<COPY file dest> :: Copy ***file*** to ***dest***ination.

<DUMP [stack]> :: Output the contents of the named ***stack***.

<GLOBAL name value> :: Set persistent variable ***name*** to ***value***.

Note: Variables can be used in any argument or an assignment.

The Implementation

Written in Java.

Classes include:

- **PDSLabel**: PDS label parser.
- **PPIOption**: Command line option parser.
- **PPIRuleset**: Ruleset processor.
- **PPITable**: Delimited table parser.
- **PPITime**: Time parser and formatter.

Labeler

- An application to run the ruleset processor within a file system.
- Can walk a tree and apply ruleset to each file at each level.
- Simple command line invocation. Syntax:

```
java labeler ruleset pathname
```

where ***ruleset*** is the file containing the ruleset to process and ***pathname*** is the directory or name of the file to process. If ***pathname*** is a directory, then all files in the directory and all sub-directories are processed.

Plug-ins - Current

Current set of plug-ins:

- **FormatDescription**: Word wrap and indent text.
- **IMath**: Perform simple integer math.
- **LabelValue**: Extract a value from a label.
- **Lookup**: Find a value in an interval lookup spreadsheet.
- **SpreadSheet**: Parse files containing a spreadsheet (delimited text) and determine metrics.
- **Strings**: Determine length, change case, index, and subset strings.
- **TabStartStop**: Return a portion (column) of the first and last rows in an ASCII table.
- **TargetPhrase**: Create a properly punctuated phrase describing a list a values.
- **Time**: Parse and construct time strings in many formats.

Plug-ins under development

- p-chronos: A Plug-in which will call the SPICE chronos utility and format its output for use in a ruleset.

How it Works

Ruleset

```
<MESSAGE "This is a very simple example">

<TEMPLATE template.lbl>
<INCLUDE constant.rul>
<IF $FILE_EXT = "FFH">
      $DESCRIPTION = "This is a test"
<ELSEIF $FILE_EXT == "TXT">
      <IF $FILE_BASE = "README">
                <MESSAGE "This is the readme file.">
      <ELSE>
                <MESSAGE "This is another type of text
      file.">
      </IF>
      <IGNORE>
<ELSE>
      <MESSAGE "Skipping all others: $PATH_NAME
      ($FILE_EXT)">
      <IGNORE>
</IF>
```

constant.rul

```
$PDS_VERSION = PDS3
$DSID = DSID_1_0
$STD_PROD_ID = DATA
$PROD_TYPE = DATA
$REC_TYPE = FIXED
      •
      •
      •
$COL_DESCR = "What?"
$HDR_BYTES = 80
$HDR_TPYE = FIXED
$HDR_DESCR = "This is the header file"
```

Template

```
PDS_VERSION_ID                = $PDS_VERSION
DATA_SET_ID                   = "$DSID"
STANDARD_DATA_PRODUCT_ID      = "$STD_PROD_ID"
PRODUCT_ID                    = "$FILE_BASE"
PRODUCT_TYPE                  = "$PROD_TYPE"
PRODUCT_CREATION_TIME         = $FILE_TIME

RECORD_TYPE                   = $REC_TYPE
RECORD_BYTES                  = $RECL
FILE_RECORDS                  = $RECS

START_TIME                    = $START_TIME
STOP_TIME                     = $STOP_TIME
SPACECRAFT_CLOCK_START_COUNT  = "$START_SCLK"
SPACECRAFT_CLOCK_STOP_COUNT   = "$STOP_SCLK"

INSTRUMENT_HOST_NAME          = "$HOST_NAME"
INSTRUMENT_HOST_ID            = "$HOST_ID"
ORBIT_NUMBER                  = $ORBIT
TARGET_NAME                   = $TARGET_LIST
INSTRUMENT_NAME               = "$INST_NAME"
INSTRUMENT_ID                 = "$INST_ID"
DESCRIPTION                   = "
$STD_PROD_DESCR"

NOTE                          = "
$FF_ABSTRACT"

^TABLE                        = "$FILE_BASE.FFD"
OBJECT                        = TABLE
    INTERCHANGE_FORMAT        = "$INTERCHANGE"
    ROWS                      = $RECS
    COLUMNS                   = $COLS
    ROW_BYTES                 = $RECL
    ^STRUCTURE                = "$FMT"
    DESCRIPTION               = "
      $COL_DESCR"
END_OBJECT                    = TABLE

^HEADER                       = "$FILE_BASE.FFH"
OBJECT                        = HEADER
    BYTES                     = $HDR_BYTES
    HEADER_TYPE               = "$HDR_TPYE"
    DESCRIPTION               = "$HDR_DESCR"
END_OBJECT                    = HEADER
END
```

How it Works

Label

```
PDS_VERSION_ID                = PDS3
DATA_SET_ID                   = "DSID_1_0"
STANDARD_DATA_PRODUCT_ID      = "DATA"
PRODUCT_ID                    = "EXAMPLE"
PRODUCT_TYPE                  = "DATA"
PRODUCT_CREATION_TIME         = 2003-04-17T11:05:02

RECORD_TYPE                   = FIXED
RECORD_BYTES                  = 64
FILE_RECORDS                  = 10

START_TIME                    = 2002-10-6
STOP_TIME                     = 2003-01-12
SPACECRAFT_CLOCK_START_COUNT  = "2400:0"
SPACECRAFT_CLOCK_STOP_COUNT   = "2500:0"

INSTRUMENT_HOST_NAME          = "Galileo"
INSTRUMENT_HOST_ID            = "GLL"
ORBIT_NUMBER                  = 1024
TARGET_NAME                   = JUPITER
INSTRUMENT_NAME               = "MAG"
INSTRUMENT_ID                 = "MAG"
DESCRIPTION                   = "
    This is a short description"

NOTE                          = "
    This is a much longer multi-line type description which
      spans multiple
    lines."

^TABLE                        = "EXAMPLE.FFD"
OBJECT                        = TABLE
  INTERCHANGE_FORMAT            = "ASCII"
  ROWS                          = 10
  COLUMNS                       = 4
  ROW_BYTES                     = 64
  ^STRUCTURE                    = "Unknown"
  DESCRIPTION                   = "
    This the the description of a column from setvars.bat"
END_OBJECT                    = TABLE

^HEADER                       = "EXAMPLE.FFH"
OBJECT                        = HEADER
  BYTES                         = 80
  HEADER_TYPE                   = "FIXED"
  DESCRIPTION                   = "This is the header file"
END_OBJECT                    = HEADER
END
```

How We Are Using Labeler

- Add keywords to existing labels.
- Upgrade labels to current standards.
- Generate labels for new data products.
- Update keyword values (i.e., improved ephemeris or pointing information)

Where to get it…

- http://www.igpp.ucla.edu/pds/