Parameter Handling System

June 2, 2019

Abstract

The parameter handling system allows tasks to retrieve parameters which are specified on the command line. The parameters are defined in a Parameter Specification File. The current system utilises:

- default values
- complex expressions for allowable values
- parameters that can have child-parameters, depending on its value
- implicit setting of controling parameters, if a child is activated on the command line

1 Introduction

Tasks may be configured by a set of parameters, which are entered on the UNIX command line. Every task has a Parameter Specification File which defines the name and type of each parameter and possibly a default value, allowable range, etc. An Application Programmers Interface (API) allows parameters to be retrieved by tasks written in either C++ or Fortran-90.

Parameters may also be entered using the Graphical User Interface (GUI). This reads the parameter specification file and provides a dialog window into which the user can enter parameters. The GUI then launches the task by generating a command line. For further information, see the documentation of the 'gui' package.

2 Command line syntax

Parameters are specified on the command-line in either of the following formats:

- GNU long option style: `- -<param>=<value>`
- FTOOLS XPI parameter interface style: `<param>=<value>`

The command line may only specify parameters which are defined in the parameter file.

Examples:

```
foo --ival=42 --rval=1.23          # Simple parameters
foo --sval=hello                   # Simple string
foo --sval='hello world'           # String with space
foo --slist='"one two" three'      # List of two strings
```

Arguments which contain spaces must be quoted. This also applies if the string has a leading or trailing space which must be preserved. For example:

```
foo --sval=' hello world'
```

The shell's file-name expansion mechanism cannot be used directly to specify a list of filenames. File-name expansion may be implemented within the parameter package, at a later date. In the meanwhile, the following example shows a work-around solution:

```
foo --flist="'echo *.fits *.out'"
```

The single-quote character may be used in an argument, provided that it is correctly quoted or escaped. The following examples are both valid:

```
foo --sval="can't wait"
foo --sval='can'\''t wait'
```

If the parameters does not contain allowable values, as defined by the parameter specification file, an error message is given and the task is not run. The user should then correct the command-line, using the shell's history mechanism.

Command-line options with a single '-' are used for certain generic functions and are not considered as task parameters. See the documentation of the **taskmain** package for further details.

3 Parameter types

The following parameter types are supported:

bool type that can have two values: true or false

int a member of the set of whole numbers

real a member of the set of rational and irrational numbers

string sequence of characters

directory name of the location of a list of files on a storage device

file name of any file

dataset name of a SAS data file

block general type for a section of data within a dataset

table a block in a dataset, interpreted as a table

array a block in a dataset, interpreted as an array

column a column of data within a table of a dataset

angle type that allows different formats of angle specification

time type that allows different formats of time specification

Each parameter may be specified to contain either a scalar (single) value or a list of values:

list<bool>

list<int>

list<real>

list<string>

list<directory>

list<file>

list<dataset>

list<block>

list<table>

list<array>

list<column>

list<angle>

list<time>

The use of different types allows the parameter interface to check that the value is syntactically correct and, where appropriate, within an allowable range. It also allows the GUI to provide appropriate widgets for entering the values.

For example, a file-name is simply a string, as far as the task is concerned. However, using a separate type allows the GUI to present the user with a file dialog, instead of a simple string-edit field.

3.1 bool

Boolean values may be represented by any of the following symbols:

```
0, n, no,  f, false
1, y, yes, t, true
```

For example:

```
taskname --foo=n --bar=yes
```

3.2 int

Example:

```
taskname --foo=42 --bar=-123
```

3.3 real

Real parameters may be specified in simple decimal values or using exponent notation. For example:

```
taskname --foo=1 --bar=1.23 --x=-1.23e-5
```

3.4 string

String parameters are used to represent text strings. Strings containing spaces should be enclosed in quotes. For example:

```
taskname --foo=hello --bar='hello world'
```

3.5 directory

Directory parameters are used to specify a directory.

The directory may be expressed as a relative or absolute path. For example:

```
bar
../foo/bar
/data/foo/bar
```

3.6 file

File name parameters are used to express the name of any type of for file. To specify a SAS data file, the dataset type is used. The SAS GUI uses a file dialog so that the user can select a file.

A file name parameter may specify an absolute or relative path. For example:

```
bar.dat
../foo/bar.dat
/data/foo/bar.dat
```

3.7 dataset

Data-set parameters are used to specify SAS datasets (a.o. FITS files).

The parameter may specify an absolute or relative path. For example:

```
bar.FIT
../foo/bar.FIT
/data/foo/bar.FIT
```

There are additional parameter types which allow components of a dataset, such as blocks,tables, arrays and columns to be specified (see below).

The SAS GUI provides a special browser for examining and selecting datasets and their components.

3.8 block, table and array

Table parameters and array parameters allow a block (generic name of a table or array) and its associated dataset to be specified by a single parameter, using a colon-separated pair of the form `set:table` or `set:array`.

Block parameters are used where the value may be either a table or an array. This is appropriate for tasks which can perform generic operations, such as deleting a block.

Examples:

```
foo.FIT:R1SPE1           # Table R1SPE1 in set foo.FIT
/data/bar.FIT:FILTER_U   # Array FILTER_U in array bar.FIT
```

The SAS GUI allows a block, table or array to be selected in a single operation, using a dataset browser. This is simpler than entering separate parameters for the dataset and block-name.

3.9 column

Column parameters allow a column and its associated table and dataset to be specified as a single parameter, using a colon-separated triple, of the form *set:table:column*.

For example:

```
foo.FITS:R1SPE1:CCDNODE  # Column CCDNODE in table R1SPE1 in set foo.FITS
```

The SAS GUI allows a column to be selected in a single operation, using a dataset browser. This is simpler than entering 3 separate parameters for the dataset, table and column.

3.10 angle

Angle parameters are intended for right ascensions and declinations, where conversion from hours/minutes/seconds or degrees/minutes/seconds may be needed. For other angles, such as small offset angles, a real parameter is more appropriate. Note that the maximum range of the angle type is [-360,360] degrees.

The angle can be entered in one of the following formats:

```
(+|-)xxdxxmxx[.xxx]s      (degrees, minutes and seconds)
xxhxxmxx[.xxx]s           (hours, minutes and seconds)
[(+|-)]xxx[.xxx]          (decimal degrees)
```

where 'x' is a single digit, 'd','m','s' and 'h' are literal characters and square brackets denote optional items. The number of digits shown is not significant.

Note: Angle parameters are accessed by the task as a real value, in degrees.

3.11 time

Time parameters allow times to be entered in a variety of different formats:

```
xxxx-xx-xxTxx:xx[:xx.xxx]  (FITS date and time)
jdxxxxxx.xxx               (Julian day)
mjdxx.xxx                  (modified Julian day)
yxxxx.xxx                  (decimal year - Not yet implemented)
[-]xxxx[.xxx]              (seconds since XMM reference time)
xxx xxx xx xx:xx:xx xxxx   (calendar format)
```

where 'x' is a single digit, 'm','j','d' and 'T' are literal characters and square brackets denote optional items. The number of digits shown is not significant.

Examples:

```
1997-10-13T00:00:00.000    (FITS date and time)
jd2450734.5                (Julian day)
mjd50734                   (modified Julian day)
-6.912e+06                 (seconds since XMM reference time)
Mon Oct 13 00:00:00 1997   (calendar format)
```

Note: Times are accessed by the task as real values which represents the number of seconds since the XMM mission reference time.

3.12 list

Special types are provided for lists of each of the simple parameter types. List items are separated by spaces. For example:

```
taskname --group='xmm sas user developer'
```

String lists may use double-quotes to allow items which contain spaces or empty strings. For example, the following list contains 5 items ("ab cd", "ef", "", ",,,", "String list"):

```
taskname --list='"ab cd" ef "" ,,, "String list"'
```

4 Optional and mandatory parameters

Each parameter has a *default* value, which is used if no parameter is specified on the command line. A parameter may be either *mandatory* or *optional*. The default value and mandatory attribute are specified in the parameter file.

A mandatory parameter must be explicitly given on the command-line, otherwise an error message will be printed to prompt the user to re-enter the command. If an optional parameter is not specified, the default is used.

Note: Mandatory parameters have a default value for use in the GUI. The GUI assigns values to all parameters and does not treat mandatory parameters specially.

5 Constraints on values

If you specify a parameter on the command-line, the value of the parameter must be of the appropriate type.

Optionally the parameter file may define constraints on accepted values of that type. For example, for integer types this may be all *positive* whole numbers.

6 Errors

This section documents warnings and errors generated by this task (if any). Note that warnings and errors can also be generated in the SAS infrastructure libraries, in which case they would not be documented here. Refer to the index of all errors and warnings available in the HTML version of the SAS documentation.

ParamCorrupted *(error)*
: Parameter space got corrupted. Accessing any parameter may stop your program working.

ParamFileOpen *(error)*
: Could not open the parameter file for reading. Either the parameter does not exist or it is not readable.

ParamFileInvalid *(error)*
: The contents of the parameter file is invalid.

ParamFileNotXml *(error)*
: The contents of the parameter file violates the XML syntax

ParamIndexRange *(error)*
: The list parameter is accessed to return an item at a position, that is out of range

ParamInvalidConstraints *(error)*
: The syntax or the contents of the constraints expression is wrong

ParamMandatory *(error)*
: Lists all parameters that still need to be specified on the command-line

ParamNotFound *(error)*
Requested a parameter that is not found in the parameter space of this task

ParamNotInRange *(error)*
The value of this parameter is outside the range as expressed in the constraints

ParamNotList *(error)*
The parameter is incorrectly accessed as a list

ParamWrongQuotes *(error)*
Mismatch of quotes in the value of the parameter

ParamWrongType *(error)*
The value of this parameter has the wrong type

7 Environment variables

The parameter system uses the following environment variables:

`SAS_PATH` Parameter specification files are located by searching the 'config' directory of each path specified by the SAS_PATH environment variable.

`SAS_CLMODE` This environment variable may be used to select the command-line syntax:

- **SAS** - Normal SAS mode
- **PCS** - Special mode for PCS tools

The default mode is **SAS**.

7.1 PCS mode

The parameter system includes support for an alternative command-line syntax, required by the PCS tools. This mode is selected using the following environment variable:

```
setenv SAS_CLMODE PCS
```

In this mode, only the parameter names CCF, SF, COEFF and ODF should be specified. These options should have a single minus sign and be followed by a space-separated list of arguments. Arguments before the first of these flags are assumed to be ODF items.

For example, the following PCS command line:

```
foo odf1 odf2 -COEFF co1 co2 -CCF ccf1 ccf2 -SP res
```

is equivalent to the following SAS command line:

```
foo --ODF="odf1 odf2" --COEFF="co1 co2" --CCF="ccf1 ccf2" --SP=res
```

PCS mode should not be used for normal SAS tools. PCS mode is a deprecated feature, which may disappear if later releases of the PCS tools use the normal SAS format.

References