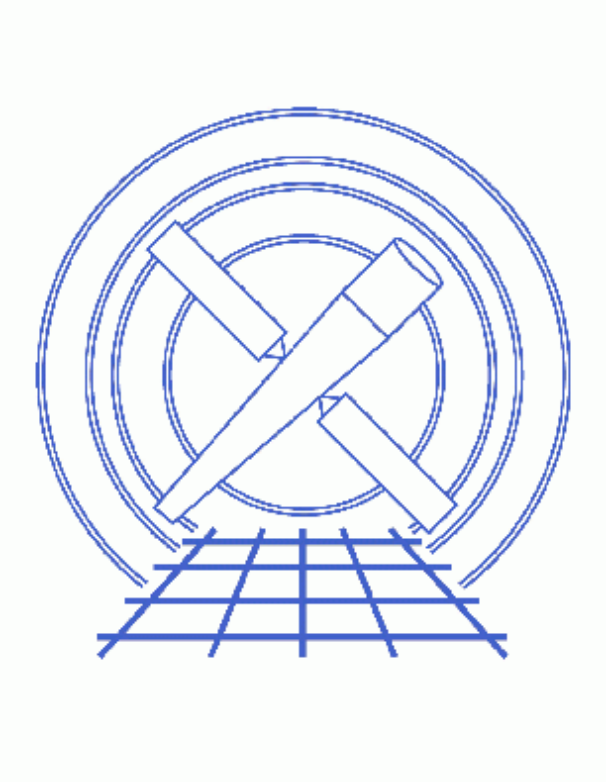
Using Parameter Files



CIAO 3.4 Science Threads

Table of Contents

URL: http://cxc.harvard.edu/ciao/threads/param_files/ Last modified: 1 Dec 2006

Using Parameter Files

CIAO 3.4 Science Threads

Overview

Last Update: 1 Dec 2006 − reviewed for CIAO 3.4: no changes

Synopsis:

The CIAO tools use ASCII parameter files to get and store processing parameters. The parameter file interface provides great flexibility in specifying parameters to programs, since their values can be obtained either from the command line or from a parameter file. This thread expands upon the basic overview of using parameter files available from the Parameter Files section of the Introduction to CIAO thread; please read that information if you have not already done so.

Read this thread if:

you are any level of CIAO user who wishes to learn more about the parameter file interface. Beginners will get a good introduction to setting and displaying parameters, while advanced users may learn new ways to reduce the amount of typing required when running a tool. There are tips and caveats included for everyone's benefit.

Related Links:

- Introduction to Peg: the Parameter Editor (peg) is a GUI application which allows the user to easily view and modify the values in a parameter file.
- `ahelp parameter`: describes the parameter interface used by CIAO. The ahelp also discusses parameter tool that aren't covered here, such as `pdump` and `pquery`.

Proceed to the HTML or hardcopy (PDF: A4 | letter) version of the thread.

Get Started

For illustration, this thread utilizes the ObsID 1843 (ACIS−I, G21.5−0.9) data that was downloaded in the How to Download Chandra Data from the Archive thread.

If this is your first time using CIAO, please read the Starting CIAO thread to ensure that your environment is configured properly. As mention in the Overview, you should also be familiar with the information in the Parameter Files section of the Introduction to CIAO thread.

Resetting Parameter Defaults

The `punlearn` command resets all the parameters to the default values. It is a good idea to `punlearn` a tool before using it for a new task, unless you are certain that you want to keep certain parameter settings.

```
unix% punlearn dmlist
unix% plist dmlist

Parameters for /home/username/cxcds_param/dmlist.par

        infile =                  Input dataset/block specification
           opt = data             Option
      (outfile = )                Output file (optional)
         (rows = )                Range of table rows to print (min:max)
        (cells = )                Range of array indices to print (min:max)
      (verbose = 0)               Debug Level(0−5)
         (mode = ql)
```

All the parameters are reset to the default values and a copy of the file is placed in the local parameter file directory (typically `$HOME/cxcds_param/`).

Viewing Parameters and Values

plist, pline, and pget

There are several different tools that may be used to list the contents of a parameter file. The one most often used is `plist`:

```
unix% plist dmlist

Parameters for /home/username/cxcds_param/dmlist.par

        infile =                  Input dataset/block specification
           opt = data             Option
      (outfile = )                Output file (optional)
         (rows = )                Range of table rows to print (min:max)
        (cells = )                Range of array indices to print (min:max)
      (verbose = 0)               Debug Level(0−5)
         (mode = ql)
```

Those parameters not within parentheses are called "positional parameters;" the parameters that are within parentheses are called "hidden parameters."

Using `pline` instead returns all of the parameters on a single line:

```
unix% pline dmlist
infile='' opt='data' outfile='' rows='' cells='' verbose='0' mode='ql'
```

This can be useful if you wish to cut−and−paste command lines into a log or script. Notice that the information on whether parameters are positional or hidden is lost.

Finally, to find the current value of a single parameter in the file, `pget` is used:

```
unix% pget dmlist opt
data
```

The default value for many parameters is the empty string. In this case, `pget` returns a blank line:

```
unix% pget dmlist infile

unix%
```

Tricks for viewing files

There are a couple tricks hiding in the parameter interface that may be useful to regular CIAO users.

1. ***tool +***

   Using "+" (the plus sign) as the one and only argument to a tool will cause it to `plist` itself:

   ```
   unix%  dmlist +

   Parameters for /home/username/cxcds_param/dmlist.par

          infile =                  Input dataset/block specification
             opt = data             Option
        (outfile = )                Output file (optional)
           (rows = )                Range of table rows to print (min:max)
          (cells = )                Range of array indices to print (min:max)
        (verbose = 0)               Debug Level(0−5)
           (mode = ql)
   ```

2. ***plist tool1 tool2 ... toolN***

   It is possible to list more than one tool's parameter file at a time:

   ```
   uinx% plist dmlist dmcopy

   Parameters for /home/username/cxcds_param/dmlist.par

          infile =                  Input dataset/block specification
             opt = data             Option
        (outfile = )                Output file (optional)
           (rows = )                Range of table rows to print (min:max)
          (cells = )                Range of array indices to print (min:max)
        (verbose = 0)               Debug Level(0−5)
           (mode = ql)


   Parameters for /home/username/cxcds_param/dmcopy.par

          infile =                  Input dataset/block specification
         outfile =                  Output dataset name
         (kernel = default)         Output file format type
         (option = )                Option − force output type
        (verbose = 0)               Debug Level
        (clobber = no)              Clobber existing file
           (mode = ql)
   ```

Setting Parameters with pset

Parameters may be set with the `pset` command in several different ways.

Without any parameters

Issuing the command "`pset <tool>`" prompts you to set the value of each parameter as they are ordered in the file. This method is helpful if you aren't sure of a parameter name or need to set many parameters at once:

```
unix% pset dmlist
Input dataset/block specification (): acisf01843N001_evt2.fits
Option (data): blocks
Output file (optional) ():
Range of table rows to print (min:max) ():
Range of array indices to print (min:max) ():
Debug Level(0−5) (0:5) (0): 1
mode (ql):
```

If a parameter is already set (e.g. the input dataset name) and you don't wish to change it, a carriage return (<RETURN> key) will keep the current value and move on:

```
unix% pset dmlist
Input dataset/block specification (acisf01843N001_evt2.fits):
Option (blocks): cols
Output file (optional) ():
Range of table rows to print (min:max) ():
Range of array indices to print (min:max) ():
Debug Level(0−5) (0:5) (1): 0
mode (ql):
```

One parameter at a time

You may also give `pset` the name and value of the desired parameter on the command line. This method is most commonly used in the CIAO threads to highlight syntax.

```
unix% pset dmlist infile=pcadf084271087N001_asol1.fits
unix% pset dmlist opt=cols
unix% plist dmlist

Parameters for /home/username/cxcds_param/dmlist.par

        infile = pcadf084271087N001_asol1.fits Input dataset/block specification
           opt = cols             Option
     (outfile = )                 Output file (optional)
        (rows = )                 Range of table rows to print (min:max)
       (cells = )                 Range of array indices to print (min:max)
     (verbose = 0)                Debug Level(0−5)
        (mode = ql)
```

Several parameters at once

The single parameter syntax may be expanded to set several parameters at once on the command line:

```
unix% pset dmlist infile=acisf01843N001_evt2.fits opt=data rows=1:5
unix% plist dmlist

Parameters for /home/username/cxcds_param/dmlist.par

        infile = acisf01843N001_evt2.fits Input dataset/block specification
           opt = data             Option
      (outfile = )                Output file (optional)
          (rows = 1:5)            Range of table rows to print (min:max)
         (cells = )               Range of array indices to print (min:max)
      (verbose = 0)               Debug Level(0−5)
          (mode = ql)
```

Setting a parameter to an empty string

It is natural to think that hitting return at a blank parameter prompt would set the value to the empty string. However, it was shown in the Setting Parameters with pset: without any parameters section that this keeps the current value instead. To set the parameter to an empty string, supply empty quotes as the value:

```
unix% pget dmlist infile outfile
acisf01843N001_evt2.fits
outfile.txt
unix% pset dmlist
Input dataset/block specification (acisf01843N001_evt2.fits): ""
Option (blocks):
Output file (optional) (outfile.txt): ""
Range of table rows to print (min:max) ():
Range of array indices to print (min:max) ():
Debug Level(0−5) (0:5) (0):
mode (ql):
unix% pget dmlist infile outfile


unix%
```

Two blank lines are printed as each parameter now contains an empty string.

Similarly, this can be done by issuing `pset` command with empty quotes as the value:

```
unix% pset dmlist in=acisf01843N001_evt2.fits
unix% pget dmlist infile
acisf01843N001_evt2.fits
unix% pset dmlist infile=""
unix% pget dmlist infile

unix%
```

There is related information on setting the empty string in the Setting Parameters without pset: Setting a parameter to an empty string section.

Setting Parameters without pset

There are also ways of specifying parameters without explicitly using `pset`. These approaches can help reduce the amount of typing needed to execute the run of a tool.

Supplying parameters on the command line

It is possible to run a tool by supplying all the required (i.e. positional) parameters when the tool is run:

```
unix% punlearn dmlist
unix% dmlist infile=pcadf084271087N001_asol1.fits opt=cols

--------------------------------------------------------------------------------
Columns for Table Block ASPSOL
--------------------------------------------------------------------------------

ColNo  Name                 Unit        Type             Range
   1   time                 s            Real8          84271087.6932067424: 84280448.2498081326 Time
   2   ra                   deg          Real8          −Inf:+Inf            RA of MNC frame (x−axis)
   3   dec                  deg          Real8          −Inf:+Inf            DEC of MNC frame (x−axis)
   4   roll                 deg          Real8          −Inf:+Inf            ROLL of MNC frame
   5   ra_err               deg          Real4          −Inf:+Inf            Uncertainty in  RA
   6   dec_err              deg          Real4          −Inf:+Inf            Uncertainty in  DEC
(etc.)
```

Notice that the parameter file ***is updated*** when you use this method:

```
unix% plist dmlist

Parameters for /home/username/cxcds_param/dmlist.par

        infile = pcadf084271087N001_asol1.fits Input dataset/block specification
           opt = cols             Option
      (outfile = )                Output file (optional)
         (rows = )                Range of table rows to print (min:max)
        (cells = )                Range of array indices to print (min:max)
      (verbose = 0)               Debug Level(0−5)
         (mode = ql)
```

Option: omitting the parameter name

When supplying values on the command line, the parameter names may be omitted, *as long as the parameters are given in order*. For example, to repeat the last example:

```
unix% dmlist pcadf084271087N001_asol1.fits cols

--------------------------------------------------------------------------------
Columns for Table Block ASPSOL
--------------------------------------------------------------------------------

ColNo  Name                 Unit        Type             Range
   1   time                 s            Real8          84271087.6932067424: 84280448.2498081326 Time
(etc.)
```

This method only works for positional (non−hidden) parameters. If you wish to send the output to a file, this command ***will not work***, as evidenced by the error:

```
unix% dmlist pcadf084271087N001_asol1.fits cols output.txt
Problem opening parameter file: too many positional arguments
```

Instead, the name of the hidden parameter must be given:

```
unix% dmlist pcadf084271087N001_asol1.fits cols outfile=output.txt
unix% more output.txt

--------------------------------------------------------------------------------
Columns for Table Block ASPSOL
--------------------------------------------------------------------------------
ColNo  Name                 Unit        Type             Range
   1   time                 s            Real8          84271087.6932067424: 84280448.
2498081326 Time
(etc.)
```

For other restrictions on using hidden parameters, see the Caveat: setting hidden parameters section.

Important: while it is valid to start off specifying parameter values by position and then switch to the "key=value" method, the reverse does not hold true. If you begin by using "key=value" pairs, you must do so for the entire command.

Caveat: switching between parameter name and positional argument

While it is valid to start off specifying parameter values by position and then switch to the "key=value" method, as shown in the "Option: omitting the parameter name" seubsection:

```
unix% dmlist pcadf084271087N001_asol1.fits cols outfile=output.txt
```

the reverse does not hold true. ***If you begin by using "key=value" pairs, you must do so for the entire command.*** The following syntax *is not valid*:

```
unix% dmlist infile=pcadf084271087N001_asol1.fits blocks
Sorry, Arguments not ok, try again.
Problem opening parameter file: parameter error?
```

Allowing the tool to prompt for them

If a positional parameter is not specified on the command line, the tool will prompt you for the value:

```
unix% dmlist acisf01843N001_evt2.fits
Option (cols): blocks

--------------------------------------------------------------------------------
Dataset: acisf01843N001_evt2.fits
--------------------------------------------------------------------------------

     Block Name                          Type         Dimensions
--------------------------------------------------------------------------------
Block    1: PRIMARY                       Null
Block    2: EVENTS                        Table        14 cols x 475869   rows
Block    3: GTI7                          Table         2 cols x 1        rows
Block    4: GTI0                          Table         2 cols x 1        rows
Block    5: GTI1                          Table         2 cols x 1        rows
Block    6: GTI2                          Table         2 cols x 1        rows
Block    7: GTI3                          Table         2 cols x 2        rows
Block    8: GTI6                          Table         2 cols x 1        rows
```

The parameter value that's given in parentheses could have been accepted by hitting the <RETURN> key. Again, the parameter file is updated when using this method.

Caveat: setting hidden parameters

The interface will not prompt you for hidden parameters; they must be specified by `pset` or given on the command line:

```
unix% punlearn dmlist
unix% dmlist infile=acisf01843N001_evt2.fits opt=data rows=1:2

--------------------------------------------------------------------------------
Data for Table Block EVENTS
--------------------------------------------------------------------------------

ROW    time                 ccd_id node_id expno      chip(chipx,chipy) tdet(tdetx,tdety) det(detx,dety)
     1  84272488.5504292250    6    3          3    (861,148)  (3736,1850) (  3697.3767089844, ...
     2  84272488.5504292250    6    3          3    (962,609)  (3837,2311) (  3798.8344726562, ...
```

Values specified on the command line for hidden parameters will be used *only* when the tool is run; they ***are not recorded*** in the parameter file.

For example, running `dmlist` with hidden parameters specified on the command line, as with the last example, results in the following recorded parameter file:

```
unix% plist dmlist

Parameters for /home/username/cxcds_param/dmlist.par

        infile = acisf01843N001_evt2.fits Input dataset/block specification
           opt = data             Option
      (outfile = )                Output file (optional)
          (rows = )               Range of table rows to print (min:max)
         (cells = )               Range of array indices to print (min:max)
      (verbose = 0)               Debug Level(0−5)
           (mode = ql)
```

However, running the tool with hidden parameters specified using `pset` results in the following recorded parameter file:

```
unix% pset dmlist rows=1:2
unix% dmlist acisf01843N001_evt2.fits data
(output omitted)

unix% plist dmlist

Parameters for /home/username/cxcds_param/dmlist.par

        infile = acisf01843N001_evt2.fits Input dataset/block specification
           opt = data             Option
      (outfile = )                Output file (optional)
          (rows = 1:2)            Range of table rows to print (min:max)
         (cells = )               Range of array indices to print (min:max)
      (verbose = 0)               Debug Level(0−5)
           (mode = ql)
```

It is therefore recommended that hidden parameters be specified using `pset` so that the parameter file accurately indicates all values used the last time that a tool was run.

Setting a parameter to an empty string

To set the value of a parameter to an empty string without using `pset`, specify the parameter on the command line with empty quotes as the value.

For instance, if you have an `outfile` set in `dmlist` and don't want to use it for the next run:

```
unix% pget dmlist outfile
out.txt
unix% dmlist acisf01843N001_evt2.fits blocks outfile=""

--------------------------------------------------------------------------------
Dataset: acisf01843N001_evt2.fits
--------------------------------------------------------------------------------

     Block Name                          Type         Dimensions
--------------------------------------------------------------------------------
Block    1: PRIMARY                        Null
Block    2: EVENTS                         Table        14 cols x 475869   rows
Block    3: GTI7                           Table         2 cols x 1        rows
Block    4: GTI0                           Table         2 cols x 1        rows
Block    5: GTI1                           Table         2 cols x 1        rows
Block    6: GTI2                           Table         2 cols x 1        rows
Block    7: GTI3                           Table         2 cols x 2        rows
Block    8: GTI6                           Table         2 cols x 1        rows
```

The output filename is ignored, so the information is printed to the screen.

There is related information on setting the empty string in the Setting Parameters with pset: Setting a parameter to an empty string section.

Abbreviating Parameter Names and Values

The parameter file interface allows you to shorten the parameter names to the smallest unique string, whether you are using `pset`:

```
unix% pget dmcopy clobber
no
unix% pset dmcopy cl+
unix% pget dmcopy clobber
yes

unix% pset dmcopy outfile=one.fits
unix% pget dmcopy outfile
one.fits
unix% pset dmcopy out=two.fits
unix% pget dmcopy outfile
two.fits
unix% pset dmcopy o=three.fits
parammatch : parameter name o not uniq (outfile, option)
unix% pset dmcopy ou=three.fits
unix% pget dmcopy outfile
three.fits
```

or not:

```
unix% dmlist in=radial.fits out=list.txt opt=blocks
unix% cat list.txt

--------------------------------------------------------------------------------
Dataset: radial.fits
--------------------------------------------------------------------------------

     Block Name                          Type         Dimensions
--------------------------------------------------------------------------------
Block    1: PRIMARY                        Null
Block    2: HISTOGRAM                      Table        24 cols x 85       rows
```

This feature is especially useful when changing the `clobber` setting for a tool. For all tools, adding "`clob+`" to the end of the command will overwrite the old output; for most tools, this can be shortened further to "`cl+`" or even "`c+`".

Related to this is the ability to use abbreviated names for the enumerated values in a parameter file as well. For example, the `grouptype` parameter in `dmgroup` will accept only a finite set of values:

```
unix% grep grouptype /soft/ciao/param/dmgroup.par
grouptype,s,a,"NONE",NONE|BIN|SNR|NUM_BINS|NUM_CTS|ADAPTIVE|ADAPTIVE_SNR|BIN_WIDTH|MIN_SLOPE|MAX_SLOPE|BIN
grouptypeval,r,a,0,,,"Grouping type value"
```

To set the `grouptype` to `MAX_SLOPE`, you can say:

```
unix% pget dmgroup grouptype
NUM_CTS
unix% pset dmgroup grouptype=MA
unix% pget dmgroup grouptype
MAX_SLOPE
```

Specifying a non−unique value returns an error and prompts you for the correct choice:

```
unix% pset dmgroup grouptype=NUM
pquery: enumerated value not unique : grouptype
Grouping type (NONE|BIN|SNR|NUM_BINS|NUM_CTS|ADAPTIVE|ADAPTIVE_SNR|BIN_WIDTH|MIN_SLOPE|MAX_SLOPE|BIN_FILE)
unix% pget dmgroup grouptype
NUM_BINS
```

Running Multiple Instances of a Tool

If you wish to have several runs of the same CIAO tool going at the same time, it is necessary to set up different parameter files for each run (since it is unlikely that the parameters will be unchanged for the different runs). This situation can also apply if you are analyzing multiple datasets and wish to maintain separate `ardlib.par` and other parameter files.

In the following we consider the example of running `aconvolve` to smooth an image with a two−dimensional gaussian with sigma values of 3 and 5 pixels along each dimension.

Using the '@@' syntax

If this is a once−only situation, then you can set up the parameter file, copy it to a separate location, and then

point the tool to that copy.

Here we set up two parameter files − one using a gaussian with a sigma of 3 pixels in each direction (`/tmp/aconvolve.g3.par`) and the other with the sigma value set to 5 pixels (`/tmp/aconvolve.g5.par`). We then run two copies of `aconvolve` with these individual parameter files, setting the mode to "h" to avoid any queries of automatic parameters.

```
unix% punlearn aconvolve
unix% pset aconvolve infile=img.fits method=fft
unix% pset aconvolve outfile=img.sm3.fits
unix% pset aconvolve kernelspec="lib:gauss(2,5,1,3,3)"
unix% cp `paccess aconvolve` /tmp/aconvolve.g3.par

unix% pset aconvolve outfile=img.sm5.fits
unix% pset aconvolve kernelspec="lib:gauss(2,5,1,5,5)"
unix% cp `paccess aconvolve` /tmp/aconvolve.g5.par

unix% aconvolve @@/tmp/aconvolve.g3.par mode=h &
unix% aconvolve @@/tmp/aconvolve.g5.par mode=h &
```

Caveat: the @@ syntax may not work with scripts such as `psextract`, `dmgti`, and `wavdetect`. Since scripts usually run multiple CIAO tools, there still could be race conditions unless the environment variable is set.

Changing the PFILES environment variable

A more robust solution is to run each copy of the tool with the PFILES environment variable set to a different local directory. Assuming that the default PFILES setting after starting CIAO looks like:

```
unix% echo $PFILES
/home/username/cxcds_param;/soft/ciao/param
```

and you want to use the directories "`/home/username/cxcds_param1/`" and "`/home/username/cxcds_param2/`", then you could say:

```
unix% setenv PFILES "/home/username/cxcds_param1;/soft/ciao/param"
unix% punlearn aconvolve
unix% pset aconvolve infile=img.fits method=fft
unix% pset aconvolve outfile=img.sm3.fits
unix% pset aconvolve kernelspec="lib:gauss(2,5,1,3,3)"
unix% aconvolve mode=h &

unix% setenv PFILES "/home/username/cxcds_param2;/soft/ciao/param"
unix% punlearn aconvolve
unix% pset aconvolve infile=img.fits method=fft
unix% pset aconvolve outfile=img.sm5.fits
unix% pset aconvolve kernelspec="lib:gauss(2,5,1,5,5)"
unix% aconvolve mode=h &
```

While this is in general a better approach than the previous suggestion, it brings the consequence of having to remember which parameter file directory you are currently using.

It is not required that you use an absolute path when setting PFILES; it often may make sense to use a relative path. For instance, if you set PFILES to

```
./param:/home/username/cxcds_param;/soft/ciao/param
```

then the parameter library will look for the file first in the subdirectory `param/` of the working directory, then in the `cxcds_param/` subdirectory of your home directory, and finally access the system default copy in the

CIAO distribution ($ASCDS_INSTALL, here taken to be `/soft/ciao`).

Setting the parameter file directory to the current working directory, e.g. "." (dot), is a special case. If you want to put the files in the current working directory, you must include a slash in the PFILES variable:

```
./:/home/username/cxcds_param;/soft/ciao/param
```

Miscellaneous

When are quotes needed?

When entering a complex filename on the command line, quotes are required so that the parameter interface can define the various fields:

```
unix% dmlist "acisf01843N001_evt2.fits[GTI3]" cols

-------------------------------------------------------------------------------
Columns for Table Block GTI3
-------------------------------------------------------------------------------

ColNo  Name                 Unit        Type             Range
   1   START                s            Real8          −Inf:+Inf
   2   STOP                 s            Real8          −Inf:+Inf
```

However, the quotes are not necessary when responding to a prompt and will result in an error if included. The correct syntax in this case is:

```
unix% punlearn dmlist
unix% dmlist
Input dataset/block specification (): acisf01843N001_evt2.fits[GTI3]
Option (data): cols

-------------------------------------------------------------------------------
Columns for Table Block GTI3
-------------------------------------------------------------------------------

ColNo  Name                 Unit        Type             Range
   1   START                s            Real8          −Inf:+Inf
   2   STOP                 s            Real8          −Inf:+Inf
```

Whitespace in pset commands

In all of these examples, we have not left any space between the equals sign (=) and the parameter name or value, e.g.:

```
unix% pset dmlist infile=pcadf084271087N001_asol1.fits
unix% pget dmlist infile
pcadf084271087N001_asol1.fits
```

It is also possible, however, to do:

```
unix% pset dmlist infile= pcadf084271087N001_asol1.fits
unix% pget dmlist infile
```

```
pcadf084271087N001_asol1.fits
```

and

```
unix% pset dmlist infile = pcadf084271087N001_asol1.fits
unix% pget dmlist infile
pcadf084271087N001_asol1.fits
```

On the command line, the whitespace before and after the "=" is removed. This is particularly useful in the (t)csh/bash shells, since it allows one to use Command−line completion on the file names.

Of course

```
unix% pset dmlist infile =" pcadf084271087N001_asol1.fits"
unix% pget dmlist infile
 pcadf084271087N001_asol1.fits
```

retains the space since quotes were used. (This is similar to putting a leading space when prompted for a parameter value).

Using redirects

There are several kinds of redirects that can be used in setting the value of a parameter (`ahelp parameter` has more examples of using this syntax):

- "`)paramname`" : the value for one parameter is re−directed to use the value for another parameter of the same tool.

  This type of redirect is commonly used to set the `eventdef` parameter of `acis process events`:

  ```
  unix% pset acis_process_events eventdef=")stdlev1"
  unix% pget acis_process_events eventdef
  {d:time,s:ccd_id,s:node_id,i:expno,s:chip,s:tdet,f:det,f:sky,s:phas,l:pha,f:energy,
  l:pi,s:fltgrade,s:grade,x:status}
  ```

- "`)tool.paramname`" : similar to the the previous redirection, except that the value is taken from the parameter of a different tool.

  Some CIAO tools have parameters redirected to other tools by default. In the case of `tgdetect`, several of the parameters point to the `celldetect` tool:

  ```
  unix% plist tgdetect | grep ")celldetect"
        (thresh = )celldetect.thresh -> 3) celldetect source threshold
  (celldetect_log = )celldetect.log -> no) make a celldetect log file?
      (psftable = )celldetect.psftable -> /soft/ciao/data/psfsize20010416.fits) table of PSF s
      (bkgvalue = )celldetect.bkgvalue -> 0) background count/pixel, for celldetect
    (bkgerrvalue = )celldetect.bkgerrvalue -> 0) background error, for celldetect
         (eband = )celldetect.eband -> 1.4967) energy band, for celldetect
       (eenergy = )celldetect.eenergy -> 0.8) encircled energy of PSF, for celldetect
      (convolve = )celldetect.convolve -> no) use convolutions for celldetect

  unix%  pget tgdetect eenergy
  0.8
  ```

- "`))command`" : in this case, the parameter value is found by executing the given command and using the value it returns (i.e. the standard output of the command).

This can be especially useful to the advanced CIAO user who would like to combine several commands. For example:

```
unix% punlearn dmcoords
unix% pset dmcoords infile=acisf01843N001_evt2.fits
unix% pset dmcoords asolfile=pcadf084271087N001_asol1.fits
unix% pset dmcoords celfmt=deg

unix% pset dmcoords ra="))dmkeypar acisf01843N001_evt2.fits ra_nom echo+"
unix% pset dmcoords dec="))dmkeypar acisf01843N001_evt2.fits dec_nom echo+"

unix% plist dmcoords

Parameters for /home/username/cxcds_param/dmcoords.par

        infile = acisf01843N001_evt2.fits Input dataset/block specification
...
            ra = ))dmkeypar acisf01843N001_evt2.fits ra_nom echo+ −> 278.04738610798 RA [deg or hh:m
           dec = ))dmkeypar acisf01843N001_evt2.fits dec_nom echo+ −> −10.571148740177 Dec [deg or d
(etc.)
```

Note that the `plist` may take a little longer than usual, since CIAO is evaluating the commands to get the `ra` and `dec` parameter values.

- "`%xpa()`" : XPA redirects allow you to get information from other CIAO applications that have XPA access points, such as ds9.

This example uses the XPA redirect to read the x and y coordinates of the crosshair mode in ds9 into the tool `dmcoords`:

```
unix% ds9 acisf01843N001_evt2.fits &

unix% punlearn dmcoords
unix% pset dmcoords asolfile=pcadf084271087N001_asol1.fits
unix% pset dmcoords infile="%xpa(ds9,file)"

unix% pset dmcoords opt=sky
unix% pset dmcoords x="%xpa(ds9,x)"
unix% pset dmcoords y="%xpa(ds9,y)"

# place the crosshairs at some location in the ds9 window

unix% dmcoords mode=h verb=1 | grep THETA
THETA,PHI          2.482'        108.18 deg

# move the crosshairs in ds9

unix% dmcoords mode=h verb=1 | grep THETA
THETA,PHI          4.808'        208.91 deg
```

Since the x and y coordinates are set in the parameter file as redirects, the theta value (and all other values) is automatically updated when the crosshairs are moved:

```
unix% plist dmcoords

Parameters for /home/username/cxcds_param/dmcoords.par

        infile = %xpa(ds9,file) −> acisf01843N001_evt2.fits[EVENTS] Input dataset/block specificatio
...
             x = %xpa(ds9,x) −> 4407.5 Sky X [pixel]
             y = %xpa(ds9,y) −> 3599.5 Sky Y [pixel]
```

. . .

History

22 Dec 2004	reviewed for CIAO 3.2: no changes
24 Aug 2005	added Caveat: switching between parameter name and positional argument section
01 Dec 2005	reviewed for CIAO 3.3: no changes
01 Dec 2006	reviewed for CIAO 3.4: no changes

URL: http://cxc.harvard.edu/ciao/threads/param_files/ Last modified: 1 Dec 2006