URL: http://cxc.harvard.edu/ciao3.4/toolagent.html
Last modified: December 2006

AHELP for CIAO 3.4

analysis−menu

Context: gui

Jump to: Description Examples CHANGES IN CIAO 3.0.2 CHANGES IN CIAO 3.0 Bugs See Also

Synopsis

The "Analysis" menu in CIAO GUIs allows users to run command−line tools.

Description

CIAO GUI applications − such as peg and prism − contain an "Analysis" menu from which it is possible to run command−line tools. Whilst CIAO comes with a default arrangement for the contents of this menu, it is easy to define your own menu, since it is stored as a simple text file that uses ds9's Analysis Menu Definition Language. This means that you can have the same menu in both CIAO GUIs and ds9, or you can have many menus, each tailored to a specific task.

The easiest way to learn how to customise the menu is to copy over the default file ($ASCDS_INSTALL/bin/ciao.ans), edit it, and load it up into either a CIAO GUI or ds9. If the edited file is called my.ans in the current directory then this is achieved by either:

```
unix% prism mydata.fits −toolmenu my.ans
```

or

```
unix% ds9 −analysis my.ans
```

To make this your default analysis menu file, copy it to $HOME/ciao.ans. This means that any GUIs launched from the analysis menu will also contain your customized menu.

Where is the Analysis Menu File?

The loading precedence for the menu definition is as follows (the process stops once a file is found and successfully loaded):

- If the −toolmenu command−line option is specified, use the listed file
- Check for the file "ciao.ans" in the user's home directory (i.e. $HOME/ciao.ans)
- Check for the file "CXCDSToolTable" in the user's home directory (i.e. $HOME/CXCDSToolTable)
- Look for $ASCDS_INSTALL/bin/ciao.ans
- Look for $ASCDS_INSTALL/bin/CXCDSToolTable

Writing your own menu

Since the format for the configuration file uses ds9's Analysis Menu Definition Language, the ds9 documentation can be used as well as this help file when writing the menu description file. Please note that there are several minor differences to how the analysis menu is handled between CIAO GUIs and ds9:

- The menu can only be loaded into CIAO GUIs when the application is started.
- Some of the macros specified in the ds9 format are not applicable to CIAO GUIs; unsupported entries appear as grayed out in the menus.
- ds9 parameter file specifications and bind entries are not supported.

The remainder of this section will describe the format of the menu. Note that lines beginning with the '#' character are considered to be a comment line.

Commands

A menu entry that calls a given program is specified by four lines of text arranged as:

```
<command name> [# tip <tooltip>]
<template>
menu
<command>
```

"<command name>" is the name that appears in the menu entry. "<tooltip>" is the optional tool tip for the entry; this option may not be interpreted correctly by ds9. "<template>" is the file format template used to decide whether the menu item is active for the current file (e.g., "*", "*.fits", "*.txt", etc). The word "menu" is used to signify that the entry is a menu item; ds9 also has a bind option which is not supported by the CIAO GUIs. Finally, "<command>" gives the command to be executed, which can include any supported macro expansion.

As an example, the following file defines two menu items − "Prism" and "Edit Parameter File" − that call the prism and peg tools when selected. The prism entry is active when the application containing the menu is viewing a file whose name ends in ".fits"; the entry will be grayed out for other file names whereas the "Edit Parameter File" option is available for all file names. Prism does not have a tooltip but the edit entry does. The "$filename", "$null", and "$entry()" items are macros which get expanded before the command is executed. Their meanings are described in the "Macros" section below.

```
Prism
*.fits
menu
prism $filename | $null

Edit Parameter File # tip invoke Parameter Editor Gui
*
menu
peg "$entry(Parameter file name)" infile=$filename | $null
```

What is a `tooltip'?

The text that appears to the right of "# tip" in the menu line is used to set the tooltip in CIAO GUIs. This is the message that appears when you hover on a menu item for a short time.

Macros

A powerful feature of the menu format is its ability to expand and process command−line macros before the command is executed. For example, the "Edit Parameter File" command entry specified above contains the command:

```
peg "$entry(Parameter file name)" infile=$filename | $null
```

This command contains 3 macros that are expanded before the command is executed. The "$entry(Parameter file name)" macro invokes a text dialog with the prompt 'Parameter file name'. If the user enters a name and selects the "Ok" button then the command continues, using the specified value as the filename to pass to peg. The $filename macro is replaced with the name of the current file in the application. $null is ignored by the CIAO GUIs but is included to allow the menu item to also work in ds9.

Listed below are the subset of macro expansions which are currently supported by both toolagent and ds9:

Macro	Explanation
$filename	substitutes the application's current filename
$filename(root)	as with $filename but it also removes any specified path, leaving only the root of the filename
$entry(<text>)	displays a text entry dialog with the given text message. If the "Ok" button is selected then the rest of the command will be executed with the contents of the entered text.
$message([ok|okcancel|yesno],<text>)	displays a message dialog box with the specified option buttons and text. The remainder of the command will only be executed if "ok/yes" is selected.
$text	the output of the specified command will be displayed in a window (taskmonitor when run from a CIAO GUI). It should be the last macro of a command.
$null	included for compatability with ds9. The macro tells ds9 that it should not wait for the completion status or output of the command. It is ignored by the CIAO GUIs.

Help

In addition to command entries, it is also possible to create menu items that invoke a help dialog. The synax is just:

```
help <help label>
  <helptext>
endhelp
```

where "<help label>" is the label for the menu entry, and "<helptext>" is the text to be displayed in the dialog box. This text can extend over multiple lines. The following example creates a menu item called "ACIS tools" which, when selected, displays the text within the help/endhelp delimeters.

```
help ACIS tools
   The following tools are designed to be run on level 1
   ACIS event files:
   acis_process_events
   acis_detect_afterglow
   destreak
endhelp
```

Hierarchical menus

In order to split items up into sub−menus, you can place the items within a set of hmenu/endhmenu keywords. Nesting of these menus is allowed up to a depth of 10. The syntax for the hierarchical menus is:

```
  hmenu <name> [# tip <tooltip>]
    ...
  endhmenu
```

where "<name>" is the text to appear in the menu, the optional "<tooltip>" gives the text for the tool tip, and the contents of the menu are given by whatever lies between the hmenu/endhmenu keywords (here represented by "..."). Note that indentation is not important and that hmenu/endhmenu pairs can appear within a hmenu section. For example:

```
  hmenu One
    hmenu Two
      hmenu Three
      endhmenu
    endhmenu
    hmenu Four
    endhmenu
  endhmenu
```

defines one menu item ("One") which contains two items, "Two" and "Four", and "Two" contains one item ("Three").

Separators

Separators may be added to menus by inserting a series of three consecutive hyphens on an otherwise empty line. For example, the following text will create a separator between the "Top" and "Bottom" hierarchical menu items.

```
  hmenu Top
  endhmenu
  ---
  hmenu Bottom
  endhmenu
```

Example 1

```
Subspace
*.fits
menu
dmlist infile=$filename outfile="" opt="subspace" mode="hl" | $text
```

The preceeding four lines creates a menu item called "Subspace" which does not have a tool tip. It is only available for use if the application has a file whose name ends in ".fits". When the menu item is selected, the dmlist command (with opt set to subspace) is run on the file loaded into the application and the output is sent to taskmonitor (if run from a CIAO GUI) or a terminal window (if called from ds9).

Example 2

```
Run Command # tip allows user to run whatever
*
menu
"'$entry(Enter command to execute)'" | $text
```

This entry allows a user to run whatever command−line tool they want. When the "Run Command" menu item − which has a tooltip of "allows user to run whatever" − is selected, a dialog window appears with the message "Enter command to execute". If the "Ok" button is selected then the text entered into the widget is executed, and the output displayed in taskmonitor (CIAO GUI) or a terminal (ds9).

CHANGES IN CIAO 3.0.2

Displaying long menus

Prior to CIAO 3.0.2 items at the end of a menu would not be displayed if there were too many entries to fit vertically on the screen. The menu will now be split horizontally to try and accommodate all the entries.

CHANGES IN CIAO 3.0

The format used to determine the layout of the Analysis menu has been changed to match that of ds9's Analysis Menu Definition Language, as described above. The old CXCDSToolTable format is still supported, but will be removed in a later release. As the new format is much more flexible and lets you use the same menu in CIAO GUIs and ds9 it is strongly suggested that you use the new format.

To use the old format menu description, you can either run a CIAO GUI with the "−toolmenu" option set to the location of the menu − e.g.

```
unix% prism −toolmenu /path/to/old/file
```

or store it in your home directory as the file CXCDSToolTable and make sure there is no file called ciao.ans in that directory.

Bugs

See the bugs page for the Analysis Menu on the CIAO website for an up−to−date listing of known bugs.

See Also

chips
chips
concept
configure, session
gui
ciao.par, ciaoshmem, filtwin, firstlook, gui, peg, prism, taskmonitor
modules
varmm
tools
mkoif

URL: http://cxc.harvard.edu/ciao3.4/toolagent.html
Last modified: December 2006