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

AHELP for CIAO 3.4

session

Context: concept

Synopsis

A session is a conceptual notion of how various CIAO applications interact with each other to form an integrated data analysis environment.

Description

While all of the CIAO data analysis tools may be run independently, many of them work together to form a coherent data analysis session. For instance, filtwin may load information from prism or ChIPS, filter the data, and display the results in the imager. It can then start a second copy of prism listing the filtered data.

A major benefit of this concept is that it is possible to save the state of an analysis session so that it can be resumed at a later date or shared with a colleague. For instance, if two scientists are collaborating on a research project and have access to the same data, it is possible for one scientist to save the analysis session and send a copy of the save file to the other who may then load the session to check progress, view results, and so forth.

While much of the underlying session code is transparent to users, a "Session" menu is available in GUIs such as prism and filtwin to allow users to take full advantage of the functionality. This menu contains options to allow a user to save the current state of their analysis session, restore a previous data analysis session from a save file (only available in filtwin), or exit out of all of the applications in the data session with a single action.

The parameter file CIAO.par controls the user configurable preferences of the data analysis session; see "ahelp ciao.par" for more information.

What is a CIAO analysis session?

In simplest terms, a CIAO analysis session is any of the various tools and their data. It could be a single tool examining a data file or a collection of tools used in conjuction with each other to analyze one or more observations. Not all CIAO tools are equipped with the session functionality; at present, only filtwin, prism, and ChIPS utilize this functionality.

In actuality, the session concept is governed by a combination of the username, hostname, and display name on which applications are run. Two applications with identical values for these three factors are considered part of the same session. If any of the three values differ between two applications, they are considered separate sessions. For instance, two users logged into a remote host with the same userid but directing output to their individual terminals would be considered as two different sessions. Additionally, a constraint on the

number of instances of any given application is also imposed. A session may only contain a single instance of filtwin but may have a user−configurable number of instances of prism or ChIPS in it.

What benefit do I get from the analysis session?

While a user generally will not notice the interactions within the session, they will still benefit from it. The biggest benefit they may notice is in regards to conflict reduction. Without the session, two users running on the same host may affect each other's applications without even realizing it. For instance, its possible for one of them to attempt to display their data only to have the data appear in the other user's imager. The session also prevents a user from conflicts they may incur when running multiple instances of ChIPS in an iterative fashion. Additionally, as mentioned before, the ability to save and restore the current data session is a great asset.

How can I turn session support off?

The "−nosession" command−line option allows a GUI to be run outside of the session support. This can be used with prism, chips, filtwin, and taskmonitor. Session support is automatically turned off if the maximum number of instances of an application are already running (the "maxinstances" parameter of ciao.par) or if the application is unable to access the shared memory resources.

How do I use the XPA access points?

To find out what XPA access points are available you can use either the XPA command−line tools provided with CIAO (see the XPA documentation for details) or the XPA S−Lang module. Here we use the command−line tool xpaget to list all the available nameservers after starting chips and two Prism's:

```
unix% xpaget xpans
v2.0.ciaouser.chandra.chandra prism gs /tmp/.xpa/v2.0.ciaouser.chandra.chandra_prism.28187 ciaouser
v2.0.ciaouser.chandra.chandra prism2 gs /tmp/.xpa/v2.0.ciaouser.chandra.chandra_prism2.28188 ciaouser
v2.0.ciaouser.chandra.chandra chips gs /tmp/.xpa/v2.0.ciaouser.chandra.chandra_chips.28190 ciaouser
```

This shows that the access points are called prism, prism2, and chips and that they all have the same session key. An alternative way to find out if a specific access point exists is to use the xpaaccess tool:

```
unix% xpaaccess prism2
yes
```

To find out what commands are recognised by a particular access point you just use xpaget with the name of the access point and a list of the known commands will be printed to the standard output. As an example, when ChIPS is running you get:

```
unix% xpaget chips
cmd:            Any valid ChIPS command
                options: see ChIPS documentation for each command
exit:           Exit the current application
                options: None
quit:           Synonym for "exit"
```

and so to close down this instance of ChIPS you can say − at the command line −

```
unix% xpaset −p chips quit
```

The XPA documentation provides a much fuller description of how to use these tools. In particular if you want to send or receive data via XPA you should read the descriptions of the xpaset and xpaget routines.

CHANGES IN CIAO 3.0

Simplfied names for XPA access points

As of CIAO 3.0 the name of the default XPA access point of a GUI is set to the application's name, with a number appended to it in the case that that access point already exists. So instead of having to say something like (the exact name depends on your user and machine names):

```
unix% xpaset −p "prism:ciaouser.lagado.lagado.1" quit
unix% xpaset −p "prism:ciaouser.lagado.lagado.2" quit
```

you can just say

```
unix% xpaset −p prism quit
unix% xpaset −p prism2 quit
```

Renaming the XPA access point

As well as a simpler default value, you can now also change the name used by a GUI using the "−xpa" command−line option.

```
unix% prism &
unix% xpaset −p prism quit
unix% prism −xpa newname &
unix% xpaset −p newname quit
```

The CXCDS_SESSION_QUALIFIER environment variable

If set, the contents of the CXCDS_SESSION_QUALIFIER environment variable are used as a prefix when creating the name of the XPA access point of tools. So if CXCDS_SESSION_QUALIFIER were set to "cxc_" and then prism were started, its XPA acccess point would be called "cxc_prism" rather than "prism".

THE SESSION KEY

The "Session key" is used to allow tools to communicate with each other and is defined as a string with the following format,

```
v2.0.logname.hostname.display
```

where

- logname is the user's login name ($LOGNAME)
- hostname is the machine benig used ($HOST)
- display is the $DISPLAY environment variable after removing any text after (and including) the colon.

If the CXCDS_SESSION_QUALIFIER environment variable is set then this value is appended to the end of the key (separated by a "."), which allows multiple sessions to be defined on the same machine.

The value of the key for the current session can be found using:

```
unix% ciaoshmem −i
v2.0.ciaouser.chandra.chandra
```

This example assumes that $LOGNAME is set to "ciaouser", that the machine (and hence the $HOST variable) is called "chandra" and $DISPLAY is set to "chandra:0.0".

HOUSE KEEPING FILES

When new resources are allocated for a session, a small file is created in your $HOME/.ciao/ directory and used to store pertinent information about the resource. The file will be automatically deleted when the resource is no longer in use, and should not be changed by the user. It is mainly useful when trying to track down problems when resources have not been cleanly released.

Configuration Error: Unable to create or access shared memory segment

The cause of the above error message − seen when starting an application − is that the program was unable to create or use a system resource it required. This mostly occurs when a previous application has not exited cleanly, such as after a program has crashed.

The first thing to try is "ciaoshmem −c", or "ciaoshmem −c all", to clear up the resources. If this does not work then look in your $HOME/.ciao/ directory for any left over files which begin with "session_" and end with the session key; it is best to do this after shutting down all the CIAO GUIs to ensure that any files in this directory are truly due to un−freed resources. If any files do exist then:

Cat the file

The files are short ASCII files that list the allocated resource (the name of the file will depend on the value of the session key):

```
unix% cat $HOME/.ciao/session_v2.0.ciaouser.chandra.chandra
 Session Key      : 0xf75bff1
 Session Prefix   :
 Shared Memory id : 72201
 Max instances    : 10
 Buffer size      : 14764
```

Check the allocated resources

The UNIX ipcs command can be used to list allocated resources used by the system (by both CIAO and other tools). The output depends on what system you are using; that below is an example of the Solaris 2.8 output:

```
unix% ipcs
IPC status from <running system> as of Wed Jun  4 19:20:12 EDT 2003
T         ID      KEY        MODE        OWNER    GROUP
Message Queues:
Shared Memory:
m          0   0x500002db --rw-r--r--     root     root
m      72201   0xf75bff1  --rw-rw-rw- ciaouser     head
Semaphores:
s   47710208   0xf75bff1  --ra-ra-ra- ciaouser     head
```

The important lines are those whose KEY values match that of the "Session Key" line from the $HOME/.ciao/ file. In this example that is 0xf75bff1 which means the lines

```
m      72201   0xf75bff1  --rw-rw-rw- ciaouser     head
s   47710208   0xf75bff1  --ra-ra-ra- ciaouser     head
```

The UNIX ipcrm command can be used to remove these resources; the exact syntax depends on what system you are using, but for Solaris 2.8 you would say:

```
unix% ipcrm −m 72201 −s 47710208
```

See Also

gui

analysis−menu, ciao.par, ciaoshmem, filtwin, firstlook, gui, peg, prism, taskmonitor

tools

mkoif

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