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

AHELP for CIAO 3.4

xpaset

Context: xpa

Jump to: Description Examples CHANGES IN CIAO 3.2 See Also

Synopsis

Send a command or data to one or more XPA servers.

Syntax

```
Integer_Type xpaset( String_Type dest, String_Type cmd, Any_Type data,
... )

(names, msgs) = XPASet( XPA_Type xpaHdl, String_Type dest, String_Type
cmd )
(names, msgs) = XPASet( XPA_Type xpaHdl, String_Type dest, String_Type
cmd, Integer_Type max_rec )
(names, msgs) = XPASet( XPA_Type xpaHdl, String_Type dest, String_Type
cmd, Integer_Type max_rec, Any_Type data )
(names, msgs) = XPASet( XPA_Type xpaHdl, String_Type dest, String_Type
cmd, Integer_Type max_rec, Any_Type data, String_Type mode )

where names and msgs are both String_Type arrays.
```

Description

The xpaset() and XPASet() functions are used to send commands or data to one or more XPA servers. This allows a user to remotely control an application which contains an XPA server; an example of this is to tell ds9 to load a given file. The xpaset() function provides a simple, easy to use, interface which is similar to the xpaset command−line tool from the XPA package, whilst the XPASet() function gives the user more control.

The xpaset() function

The function automatically appends all scalar parameters (char, short, integer, long, float, or string) to the XPA command (the value of the cmd variable) before transmission. If a non−scalar parameter argument is present (e.g., an array of floats) it will be treated as "data" (see the XPA documentation) for the command, and any subsequent arguments will be ignored. The return value gives the number of application servers that were contacted.

The XPASet() function

This function is similar to xpaset(), but provides more user control. It uses an XPA_Type handle returned by XPAOpen(), and permits the user to explicitly limit the number of applications to contact (the max_rec parameter). The mode parameter is currently unused.

The return values from XPASet() are the names and messages values for each contacted server. The length of either array gives the number of servers that were sent the message.

Example 1

```
chips> require( "xpa" )
chips> xpaset( "prism", "file hrc_evt2.fits" )
1
```

Here we send the command

```
file hrc_evt2.fits
```

to all XPA servers that match the name

```
prism
```

The require() call is only needed if the XPA module has not already been loaded.

Example 2

```
chips> system("ds9 &")
chips> img = _reshape( [1:65536], [256,256] )
chips> fits_bitpix( img )
32
chips> xpaset("ds9","array new [dim=256,bitpix=32]", img )
1
```

In this example we created an image (256 x 256 pixels with values from 1 to 65536) and sent it to ds9 using the "array" XPA command. The system() call starts up a ds9 to send the data to, and the fits_bitpix() function is used to find out what value to use for the bitpix parameter of the XPA command. See the DS9 XPA Access Points page for more information on how to control DS9 using XPA.

Example 3

```
chips> xpaset("prism","file hrc_evt2.fits")
1
```

The return value of 1 means that 1 prism process was sent the message. It does not indicate that the command was successful, just that it was sent.

CHANGES IN CIAO 3.2

The return values of xpaset() and XPASet() have changed in CIAO 3.2. See the "Backwards Compatability" section below for a way of loading the module so that the old behavior is retained. The changes are:

- xpaset() now returns the number of servers that were sent the message, rather than using the slxpa_errno variable;
- XPASet() now returns two arrays − names and messages − rather than an integer value listing the number of contacted servers.

Backwards Compatability

The old behavior − xpaset() has no return vallue and XPASet() only returns an Integer_Type variable − can be retained by declaring the variable

```
_CIAO3_XPA_COMPAT_
```

in the Global namespace prior to the first loading of the module. As an example, after:

```
if (0 != _featurep("xpa"))
  error("The XPA module has already been loaded");
public variable _CIAO3_XPA_COMPAT_;
require("xpa");
```

then the CIAO 3.1 versions of the commands will be used. The first statement, fearuting the _featurep() function, is not necessary, but added as a precaution to check that the XPA module has not previously been loaded.

See Also

modules

xpa

xpa

_slxpa_version, slxpa_errno, xpa_maxhosts, xpa_version, xpaaccess, xpaclose, xpaget, xpagetb, xpagettofile, xpaopen

---

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