pyds9 Documentation

Release 1.8

Eric Mandel

September 24, 2015

CONTENTS

A Python Connection to DS9 via XPA

The XPA messaging system (http://hea-www.harvard.edu/saord/xpa) provides seamless communication between many kinds of Unix programs, including Tcl/Tk programs such as ds9. The pyds9 module uses a Python interface to XPA to communicate with ds9. It supports communication with all of ds9's XPA access points. See http://hea-www.harvard.edu/saord/ds9/ref/xpa.html for more info on DS9's access points.

pyds9 is available from GitHub:

> https://github.com/ericmandel/pyds9

To install in the default directory:

```
# install in default system directory
> python setup.py install
```

or to install in a user-specified directory:

```
# install in specified directory
> python setup.py install --prefix=<install-dir>
> setenv PYTHONPATH <install-dir>lib/python2.x/site-packages
```

To run:

```
# start up python
> python
... (startup messages) ...
>>> from pyds9 import *
>>> print ds9_targets()
>>> d = Ds9()
```

The setup.py install will build and install both the XPA shared library and the xpans name server. By default, the code generated for the shared object will match the address size of the host machine, i.e. 32-bit or 64-bit as the case may be. But on 64-bit Intel machines, the XPA build also will check whether python itself is 64-bit. If not, it will add the "-m32" switch to the compile options to build a 32-bit shared object. This check can be overridden by defining the CFLAGS environment variable (which can be anything sensible, including an empty string).

Contact saord@cfa.harvard.edu for help.

The DS9 Class:

class pyds9.**DS9**(*target=u'DS9:*'*, *start=True*, *wait=10*, *verify=True*)

The DS9 class supports communication with a running ds9 program via the xpa messaging system. All of ds9's xpa access points are available via the DS9.get() and DS9.set() methods:

- str = get(paramlist): get data or info from ds9
- n = set(paramlist, [buf, [blen]]): send data or commands to ds9

The get method returns the data as a string, while the set method returns the number of targets successfully processed (i.e., 1 means success, while 0 probably means the ds9 is no longer running).

DS9's xpa access points are documented in the reference manual:

- http://hea-www.harvard.edu/saord/ds9/ref/xpa.html

In addition, a number of special methods are implemented to facilitate data access to/from python objects:

- get_arr2np: retrieve a FITS image or an array into a numpy array
- get_np2arr: send a numpy array to ds9 for display
- get_pyfits: retrieve a FITS image into a pyfits (or astropy) hdu list
- set_pyfits: send a pyfits (or astropy) hdu list to ds9 for display

__init__ (*target=u'DS9:*'*, *start=True*, *wait=10*, *verify=True*)

Parameters

- **target** – the ds9 target name or id (default is all ds9 instances)
- **start** – start ds9 if its not already running (optional: instead of True, you can specify a string or a list of ds9 command line args)
- **wait** – seconds to wait for ds9 to start
- **verify** – perform xpaaccess check before each set or get?

Return type DS9 object connected to a single instance of ds9

The DS9() contructor takes a ds9 target as its main argument. If start is True (default), the ds9 program will be started automatically if its not already running.

The default target matches all ds9 instances. (Note that ds9 instances are given unique names using the -title switch on the command line). In general, this is the correct way to find ds9 if only one instance of the program is running. However, this default target will throw an error if more than one ds9 instance is running. In this case, you will be shown a list of the actively running programs and will be asked to use one of them to specify which ds9 is wanted:

```
>>> DS9()
More than one ds9 is running for target DS9:*:
DS9:foo1 838e29d4:42873
DS9:foo2 838e29d4:35739
Use a specific name or id to construct a ds9 object, e.g.:
d = ds9('foo1')
d = ds9('DS9:foo1')
d = ds9('838e29d4:42873')
The 'ip:port' id (3rd example) will always be unique.
...
ValueError: too many ds9 instances running for target: DS9:*
```

You then can choose one of these to pass to the contructor:

```
d = DS9('838e29d4:35739')
```

Of course, you can always specify a name for this instance of ds9. A unique target name is especially appropriate if you want to start up ds9 with a specified command line. This is because pyds9 will start up ds9 only if a ds9 with the target name is not already running.

If the verify flag is turned on, each ds9 method call will check whether ds9 is still running, and will throw an exception if this is not the case. Otherwise, the method return value can be used to detect failure. Using verification allows ds9 methods to used in try/except constructs, at the expense of a slight decrease in performance.

access ()

Return type xpa target name and id

The 'access' method returns the xpa id of the current instance of ds9, by making a direct contact with ds9 itself.

get (*paramlist=None*, *decode=None*)

Parameters

- **paramlist** – command parameters (documented in the ds9 ref manual)
- **decode** – decode the output; if `None` decodes the output if `paramlist` is not present in the list `ds9Globals['bin_cmd']`

Return type returned data or info (as a string or byte string)

Once a DS9 object has been initialized, use 'get' to retrieve data from ds9 by specifying the standard xpa paramlist:

```
>>> d.get("file")
'/home/eric/python/ds9/test.fits'
>>> d.get("fits height")
'15'
>>> d.get("fits width")
'15'
>>> d.get("fits bitpix")
'32'
```

Note that all access points return data as python strings.

get_arr2np()

Return type numpy array

To read a FITS file or an array from ds9 into a numpy array, use the 'get_arr2np' method. It takes no arguments and returns the np array:

```
>>> d.get("file")
'/home/eric/data/casa.fits[EVENTS]'
>>> arr = d.get_arr2np()
>>> arr.shape
(1024, 1024)
>>> arr.dtype
dtype('float32')
>>> arr.max()
51.0
```

get_pyfits()

Return type pyfits hdulist

To read FITS data or a raw array from ds9 into pyfits, use the 'get_pyfits' method. It takes no args and returns an hdu list:

```
>>> hdul = d.get_pyfits()
>>> hdul.info()
Filename: StringIO
No.    Name         Type      Cards   Dimensions   Format
0    PRIMARY     PrimaryHDU      24  (1024, 1024)  float32
>>> data = hdul[0].data
>>> data.shape
(1024, 1024)
```

info(*paramlist*)

Return type 1 for success, 0 for failure

Once a DS9 object has been initialized, use 'info' to send xpa info messages to ds9. (NB: ds9 currently does not support info messages.)

set(*paramlist*, *buf=None*, *blen=-1*)

Parameters **paramlist** – command parameters (documented in the ds9 ref manual)

Return type 1 for success, 0 for failure

Once a DS9 object has been initialized, use 'set' to send data and commands to ds9:

```
>>> d.set("file /home/eric/data/casa.fits")
1
```

A return value of 1 indicates that ds9 was contacted successfully, while a return value of 0 indicates a failure.

To send data (as well as the paramlist) to ds9, specify the data buffer in the argument list. The data buffer must either be a string, a numpy.ndarray, or an array.array:

```
>>> d.set("array [xdim=1024 ydim=1024 bitpix=-32]", arr)
```

Sending both a paramlist and data is the canonical way to send a region to ds9:

```
>>> d.set('regions', 'fk5; circle(345.29,58.87,212.58")');
1
```

This is equivalent to the Unix xpaset command:

> echo 'fk5; circle(345.29,58.87,212.58")' | xpaset ds9 regions

Indeed, if you are having problems with ds9.set() or ds9.get(), it often is helpful to try the equivalent command using the Unix xpaset and xpaget programs.

set_np2arr (*arr*, *dtype=None*)

Parameters

- **arr** – numpy array
- **dtype** – data type into which to convert array before sending

Return type 1 for success, 0 for failure

After manipulating or otherwise modifying a numpy array (or making a new one), you can display it in ds9 using the 'set_np2arr' method, which takes the array as its first argument:

```
>>> d.set_np2arr(arr)
1
```

A return value of 1 indicates that ds9 was contacted successfully, while a return value of 0 indicates a failure.

An optional second argument specifies a datatype into which the array will be converted before being sent to ds9. This is important in the case where the array has datatype np.uint64, which is not recognized by ds9:

```
>>> d.set_np2arr(arru64)
...
ValueError: uint64 is unsupported by DS9 (or FITS)
>>> d.set_np2arr(arru64,dtype=np.float64)
1
```

Also note that np.int8 is sent to ds9 as int16 data, np.uint32 is sent as int64 data, and np.float16 is sent as float32 data.

set_pyfits (*hdul*)

Parameters **hdul** – pyfits hdulist

Return type 1 for success, 0 for failure

After manipulating or otherwise modifying a pyfits hdulist (or making a new one), you can display it in ds9 using the 'set_pyfits' method, which takes the hdulist as its sole argument:

```
>>> d.set_pyfits(nhdul)
1
```

A return value of 1 indicates that ds9 was contacted successfully, while a return value of 0 indicates a failure.

Auxiliary Routines:

pyds9.**ds9_targets**(*target=u'DS9:*'*)

Parameters **target** – ds9 target template (default: all ds9 instances)

Return type list of available targets matching template (name and id)

To see all actively running ds9 instances for a given target, use the ds9_targets() routine:

```
>>> ds9_targets()
['DS9:foo1 838e29d4:42873', 'DS9:foo2 838e29d4:35739']
```

You then can pass one of the ids (or names) to the DS9() constructor.

pyds9.**ds9_openlist**(*target=u'DS9:*', n=1024*)

Parameters

- **target** – the ds9 target template (default: all ds9 instances)
- **n** – maximum number of targets to connect to (default: 1024)

Return type list of connected ds9 objects

To open multiple instances of ds9, use the ds9_openlist() routine. Specify the target template and an (optional) max target count, and the routine returns a list of ds9 objects. For example, assuming 3 instances of ds9 are running with names foo1, foo2, foo3:

```
>>> ds9list = ds9_openlist("foo*")
>>> for d in ds9list:
...     print d.target, d.id
...
DS9:foo1 a000104:56249
DS9:foo2 a000104:56254
DS9:foo3 a000104:56256
>>> ds9list[1].set("file test.fits")
```

pyds9.**ds9_xpans**()

Return type 0 => xpans already running, 1 => xpans started by this routine

ds9_xpans() starts the xpans name server, if its not already running. If xpans was not running (and so was started by this routine) while ds9 was already running, an explanation on how to connect to that instance of ds9 is displayed.

CHAPTER

ONE

INDICES AND TABLES

- *genindex*
- *modindex*
- *search*

p

Symbols

A

D

G

I

P

S