nexradaws Documentation

Release 1.0

Aaron Anderson

Jul 29, 2018

Contents

1 API Docs **3**

2 Tutorial **9**

2.1 Query methods 9

2.2 Query for available scans 10

2.3 Downloading Files 11

2.4 Working with LocalNexradFile objects 11

3 Indices and tables **15**

This module is designed to allow you to query and download Nexrad radar files from Amazon Web Services S3 Storage. The real-time feed and full historical archive of original resolution (Level II) NEXRAD data, from June 1991 to present, is now freely available on Amazon S3 for anyone to use. More information can be found here https://aws.amazon.com/public-datasets/nexrad/.

nexradaws supports Python 2.7 and Python 3.6.

Github - https://github.com/aarande/nexradaws

PyPi - https://pypi.python.org/pypi/nexradaws

Required dependencies

- boto3
- pytz
- six

Optional dependencies

- pyart

Install with pip:

```
pip install nexradaws
```

Source Code

The source code for nexradaws is available on Github here. Tickets can also be opened here for any issues or enhancements requests.

Please feel free to submit Pull Requests for any fixes or enhancements.

New in Version 1.1

- Bug fix for varying filename extensions over the years (.gz .V06 etc). Thanks Nick Guy for the PR!

Contents:

CHAPTER 1

API Docs

class nexradaws.**NexradAwsInterface**

Instantiate an instance of this class to get a connection to the Nexrad AWS bucket. This class provides methods to query for various metadata of the AWS bucket as well as download files.

```
>>> import nexradaws
>>> conn = nexradaws.NexradAwsInterface()
```

get_avail_years()

This method allows you to get the years that are currently available.

```
>>> print conn.get_avail_years()
>>> [u'1991', u'1992', u'1993', u'1994', u'1995', u'1996', u'1997', u'1998', u
↪'1999', u'2000', u'2001', u'2002', u'2003', u'2004', u'2005', u'2006', u
↪'2007', u'2008', u'2009', u'2010', u'2011', u'2012', u'2013', u'2014', u
↪'2015', u'2016', u'2017']
```

Returns A list of strings representing the years available

Rtype list

get_avail_months(*year*)

This method allows you to get the available months in a given year.

```
>>> print conn.get_avail_months('2013')
>>> [u'01', u'02', u'03', u'04', u'05', u'06', u'07', u'08', u'09', u'10', u
↪'11', u'12']
```

Parameters **year** (*str or int*) – the year we are requesting available months for (i.e. 2010)

Returns A list of strings representing the months available for that year

Rtype list

get_avail_days (*year*, *month*)
This method allows you to get the available days in a given year and month.

```
>>> print conn.get_avail_days('2013','05')
>>> [u'01', u'02', u'03', u'04', u'05', u'06', u'07', u'08', u'09', u'10', u
↪'11', u'12', u'13', u'14', u'15', u'16', u'17', u'18', u'19', u'20', u'21',␣
↪u'22', u'23', u'24', u'25', u'26', u'27', u'28', u'29', u'30', u'31']
```

Parameters

- **year** (*str or int*) – the year we are requesting available days for (i.e 2010)
- **month** (*str or int*) – the month we are requesting available days for (i.e. 05)

Returns A list of strings representing the days available in the given month and year

Rtype list

get_avail_radars (*year*, *month*, *day*)
This method allows you to get the available radars in a given year, month, and day.

```
>>> print conn.get_avail_radars('2013','05','31')
>>> [u'DAN1', u'KABR', u'KABX', u'KAKQ', u'KAMA', u'KAMX', u'KAPX', u'KARX', u
↪'KATX', u'KBBX', u'KBGM', u'KBHX', u'KBIS', u'KBLX', u'KBMX', u'KBOX', u
↪'KBRO', u'KBUF', u'KBYX', u'KCAE', u'KCBW', u'KCBX', u'KCCX', u'KCLE', u
↪'KCLX', u'KCRP', u'KCXX', u'KCYS', u'KDAX', u'KDDC', u'KDFX', u'KDGX', u
↪'KDLH', u'KDMX', u'KDOX', u'KDTX', u'KDVN', u'KEAX', u'KEMX', u'KENX', u
↪'KEOX', u'KEPZ', u'KESX', u'KEVX', u'KEWX', u'KEYX', u'KFCX', u'KFDR', u
↪'KFFC', u'KFSD', u'KFSX', u'KFTG', u'KFWS', u'KGGW', u'KGJX', u'KGLD', u
↪'KGRB', u'KGRK', u'KGRR', u'KGSP', u'KGWX', u'KGYX', u'KHDX', u'KHGX', u
↪'KHNX', u'KHPX', u'KHTX', u'KICT', u'KICX', u'KILN', u'KILX', u'KIND', u
↪'KINX', u'KIWA', u'KIWX', u'KJAX', u'KJGX', u'KJKL', u'KLBB', u'KLCH', u
↪'KLGX', u'KLIX', u'KLNX', u'KLOT', u'KLRX', u'KLSX', u'KLTX', u'KLVX', u
↪'KLWX', u'KLZK', u'KMAF', u'KMAX', u'KMBX', u'KMHX', u'KMKX', u'KMLB', u
↪'KMOB', u'KMPX', u'KMQT', u'KMRX', u'KMSX', u'KMTX', u'KMUX', u'KMVX', u
↪'KMXX', u'KNKX', u'KNQA', u'KOAX', u'KOHX', u'KOKX', u'KOTX', u'KPAH', u
↪'KPBZ', u'KPDT', u'KPOE', u'KPUX', u'KRAX', u'KRGX', u'KRIW', u'KRLX', u
↪'KRTX', u'KSFX', u'KSGF', u'KSHV', u'KSJT', u'KSOX', u'KSRX', u'KTBW', u
↪'KTFX', u'KTLH', u'KTLX', u'KTWX', u'KTYX', u'KUDX', u'KUEX', u'KVNX', u
↪'KVTX', u'KVWX', u'KYUX', u'PHKI', u'PHKM', u'PHMO', u'PHWA', u'TJUA']
```

Parameters

- **year** (*str or int*) – the year we are requesting available radars for (i.e 2010)
- **month** (*str or int*) – the month we are requesting available radars for (i.e. 05)
- **day** (*str or int*) – the day we are requesting available radars for (i.e. 01)

Returns A list of string representing the radar sites available in the given day, month, and year

Rtype list

get_avail_scans (*year*, *month*, *day*, *radar*)
This method allows you to get the available radar scans for a given year, month, day, and radar.

```
>>> print conn.get_avail_scans('2013','05','31','KTLX')
>>> [AwsNexradFile object - 2013/05/31/KTLX/KTLX20130531_000358_V06.gz,␣
↪AwsNexradFile object - 2013/05/31/KTLX/KTLX20130531_000834_V06.gz,␣
↪AwsNexradFile object - 2013/05/31/KTLX/KTLX20130531_001311_V06.gz,...
```

Parameters

- **year** (*str or int*) – the year we are requesting available scans for (i.e 2010)
- **month** (*str or int*) – the month we are requesting available scans for (i.e. 05)
- **day** (*str or int*) – the day we are requesting available scans for (i.e. 01)
- **radar** (*str*) – the radar id we are requesting available scans for (i.e. KTLX)

Returns A list of *AwsNexradFile* objects representing the radar scans available for a given radar, day, month, and year

Rtype list

get_avail_scans_in_range(*start*, *end*, *radar*)

Get all available scans for a radar between start and end date. If datetime's do not include a timezone they will be set to UTC.

```
>>> from datetime import datetime
>>> radarid = 'KTLX'
>>> start = datetime(2013, 5, 31, 20, 0)
>>> end = datetime(2013, 5, 31, 23, 0)
>>> print conn.get_avail_scans_in_range(start,end,radarid)
>>> [AwsNexradFile object - 2013/05/31/KTLX/KTLX20130531_200046_V06.gz,␣
↪AwsNexradFile object - 2013/05/31/KTLX/KTLX20130531_200415_V06.gz,␣
↪AwsNexradFile object - 2013/05/31/KTLX/KTLX20130531_200745_V06.gz,...
```

Parameters

- **start** (*datetime*) – start time for range
- **end** (*datetime*) – end time for range
- **radar** (*str*) – radar id

Returns A list of *AwsNexradFile* objects representing the radar scans available in the passed time range.

Rtype list

download(*awsnexradfiles*, *basepath*, *keep_aws_folders=False*, *threads=6*)

This method will download the passed AwsNexradFile object(s) to the given basepath folder. If keep_aws_folders is True then subfolders will be created under the basepath with the same structure as on AWS (year/month/day/radar/).

Parameters

- **awsnexradfiles** (*list*) – A list of *AwsNexradFile* objects to download
- **basepath** (*str*) – location to save downloaded files
- **keep_aws_folders** (*bool*) – weather or not to use the aws folder structure inside the basepath. . . (year/month/day/radar/)
- **threads** (*int*) – number of download threads to utilize (default=6)

Returns A *DownloadResults* object that contains successful downloads as *LocalNexradFile* objects as well as any *AwsNexradFile* objects that failed

:rtype *DownloadResults*:

class nexradaws.resources.awsnexradfile.**AwsNexradFile**(*scandict*)

This class contains metadata about the remote NEXRAD file on AWS

Variables

- **key** (*str*) – AWS key for this NEXRAD file
- **last_modified** (*datetime*) – when the file was last modified on AWS
- **awspath** (*str*) – filepath on AWS to NEXRAD file
- **filename** (*str*) – the NEXRAD filename
- **scan_time** (*datetime*) – volume scan time for the NEXRAD file
- **radar_id** (*str*) – the four letter radar id (i.e. KTLX)
- **filepath** – absolute path to the downloaded file on the local system

class nexradaws.resources.localnexradfile.**LocalNexradFile**(*awsnexradfile*, *localfilepath*)

This class contains metadata about the local NEXRAD file as well as methods to open the file.

Variables

- **key** (*str*) – AWS key for this NEXRAD file
- **last_modified** (*datetime*) – when the file was last modified on AWS
- **filename** (*str*) – the NEXRAD filename
- **scan_time** (*datetime*) – volume scan time for the NEXRAD file
- **radar_id** (*str*) – the four letter radar id (i.e. KTLX)
- **filepath** – absolute path to the downloaded file on the local system

open()

Provides a file object to the local nexrad radar file. Be sure to close the file object when processing is complete.

Returns file object ready for reading

Rtype file

open_pyart()

If pyart is available this method will read in the nexrad archive file and return a pyart Radar object.

Returns a pyart radar object

Rtype pyart.core.Radar

class nexradaws.resources.downloadresults.**DownloadResults**(*localfiles*, *failedfiles*)

This class contains the results of the download call as well as methods for accessing the results.

Variables

- **success** (*list*) – a list of *LocalNexradFile* objects representing the successful downloads
- **failed** – a list of any *AwsNexradFile* objects that failed to download
- **success_count** (*int*) – The number of successful downloads
- **failed_count** (*int*) – The number of downloads that failed
- **total** (*int*) – The total number of nexrad files that were attempted

iter_success()

A generator function that allows you to iterate over successful downloads

```
>>> for localnexradfile in downloads.iter_success():
>>>       ...do something...
```

iter_failed()

A generator function that allows you to iterate over failed downloads

```
>>> for remotenexradfile in downloads.iter_failed():
>>>       ...do something...
```

In [13]:
```
%matplotlib inline
import matplotlib.pyplot as plt
import tempfile
import pytz
from datetime import datetime
import pyart

templocation = tempfile.mkdtemp()
```

CHAPTER 2

Tutorial

This notebook is designed to show examples using the nexradaws python module. The first thing we need to do is instantiate an instance of the NexradAwsInterface class. This class contains methods to query and download from the Nexrad Amazon S3 Bucket.

This notebook uses Python 3.6 for it's examples.

```
In [14]: import nexradaws
         conn = nexradaws.NexradAwsInterface()
```

2.1 Query methods

Next we will test out some of the available query methods. There are methods to get the available years, months, days, and radars.

2.1.1 Get available years

```
In [15]: years = conn.get_avail_years()
         print(years)
```

```
['1991', '1992', '1993', '1994', '1995', '1996', '1997', '1998', '1999', '2000', '2001', '2002', '20(
```

2.1.2 Get available months in a year

```
In [16]: months = conn.get_avail_months('2013')
         print(months)
```

```
['01', '02', '03', '04', '05', '06', '07', '08', '09', '10', '11', '12']
```

2.1.3 Get available days in a given year and month

```
In [17]: days = conn.get_avail_days('2013','05')
         print(days)
```

```
['01', '02', '03', '04', '05', '06', '07', '08', '09', '10', '11', '12', '13', '14', '15', '16', '17
```

2.1.4 Get available radars in a given year, month, and day

```
In [18]: radars = conn.get_avail_radars('2013','05','31')
         print(radars)
```

```
['DAN1', 'KABR', 'KABX', 'KAKQ', 'KAMA', 'KAMX', 'KAPX', 'KARX', 'KATX', 'KBBX', 'KBGM', 'KBHX', 'KBI
```

2.2 Query for available scans

There are two query methods to get available scans. * get_avail_scans() - returns all scans for a particular radar on a particular day * get_avail_scans_in_range() returns all scans for a particular radar between a start and end time

Both methods return a list of AwsNexradFile objects that contain metadata about the NEXRAD file on AWS. These objects can then be downloaded by passing them to the download method which we will discuss next.

2.2.1 Get all scans for a radar on a given day

```
In [19]: availscans = conn.get_avail_scans('2013', '05', '31', 'KTLX')
         print("There are {} NEXRAD files available for May 31st, 2013 for the KTLX radar.\n".format
         print(availscans[0:4])
```

```
There are 263 NEXRAD files available for May 31st, 2013 for the KTLX radar.

[<AwsNexradFile object - 2013/05/31/KTLX/KTLX20130531_000358_V06.gz>, <AwsNexradFile object - 2013/05
```

2.2.2 Get all scans for a radar between a start and end time

Now let's get all available scans between 5-7pm CST May 31, 2013 which is during the El Reno, OK tornado. The get_avail_scans_in_range method accepts datetime objects for the start and end time.

If the passed datetime objects are timezone aware then it will convert them to UTC before query. If they are not timezone aware it will assume the passed datetime is in UTC.

```
In [20]: central_timezone = pytz.timezone('US/Central')
         radar_id = 'KTLX'
         start = central_timezone.localize(datetime(2013,5,31,17,0))
         end = central_timezone.localize (datetime(2013,5,31,19,0))
         scans = conn.get_avail_scans_in_range(start, end, radar_id)
         print("There are {} scans available between {} and {}\n".format(len(scans), start, end))
         print(scans[0:4])
```

```
There are 26 scans available between 2013-05-31 17:00:00-05:00 and 2013-05-31 19:00:00-05:00

[<AwsNexradFile object - 2013/05/31/KTLX/KTLX20130531_220114_V06.gz>, <AwsNexradFile object - 2013/05
```

2.3 Downloading Files

Now let's download some radar files from our previous example. Let's download the first 4 scans from our query above.

There are two optional keyword arguments to the download method...

- keep_aws_folders - Boolean (default is False)... if True then the AWS folder structure will be replicated underneath the basepath given to the method. i.e. if basepath is /tmp and keep_aws_folders is True then the full path would be 2013/05/31/KTLX/KTLX20130531_220114_V06.gz
- threads - integer (default is 6) the number of threads to run concurrent downloads

```
In [21]: results = conn.download(scans[0:4], templocation)
```

```
Downloaded KTLX20130531_221445_V06.gz
Downloaded KTLX20130531_220537_V06.gz
Downloaded KTLX20130531_221011_V06.gz
Downloaded KTLX20130531_220114_V06.gz
4 out of 4 files downloaded...0 errors
```

The download method returns a DownloadResult object. The success attribute returns a list of LocalNexradFile objects that were successfully downloaded. There is also an iter_success method that creates a generator for easily looping through the objects.

```
In [22]: print(results.success)
```

```
[<LocalNexradFile object - /tmp/tmpj0ly5nl_/KTLX20130531_220114_V06.gz>, <LocalNexradFile object - /t
```

```
In [23]: for scan in results.iter_success():
             print ("{} volume scan time {}".format(scan.radar_id,scan.scan_time))
```

```
KTLX volume scan time 2013-05-31 22:01:14+00:00
KTLX volume scan time 2013-05-31 22:05:37+00:00
KTLX volume scan time 2013-05-31 22:10:11+00:00
KTLX volume scan time 2013-05-31 22:14:45+00:00
```

You can check for any failed downloads using the failed_count attribute. You can get a list of the failed AwsNexradFile objects by calling the failed attribute. There is also a generator method called iter_failed that can be used to loop through the failed objects.

```
In [24]: print("{} downloads failed.".format(results.failed_count))
```

```
0 downloads failed.
```

```
In [25]: print(results.failed)
```

```
[]
```

2.4 Working with LocalNexradFile objects

Now that we have downloaded some files let's take a look at what is available with the LocalNexradFile objects.

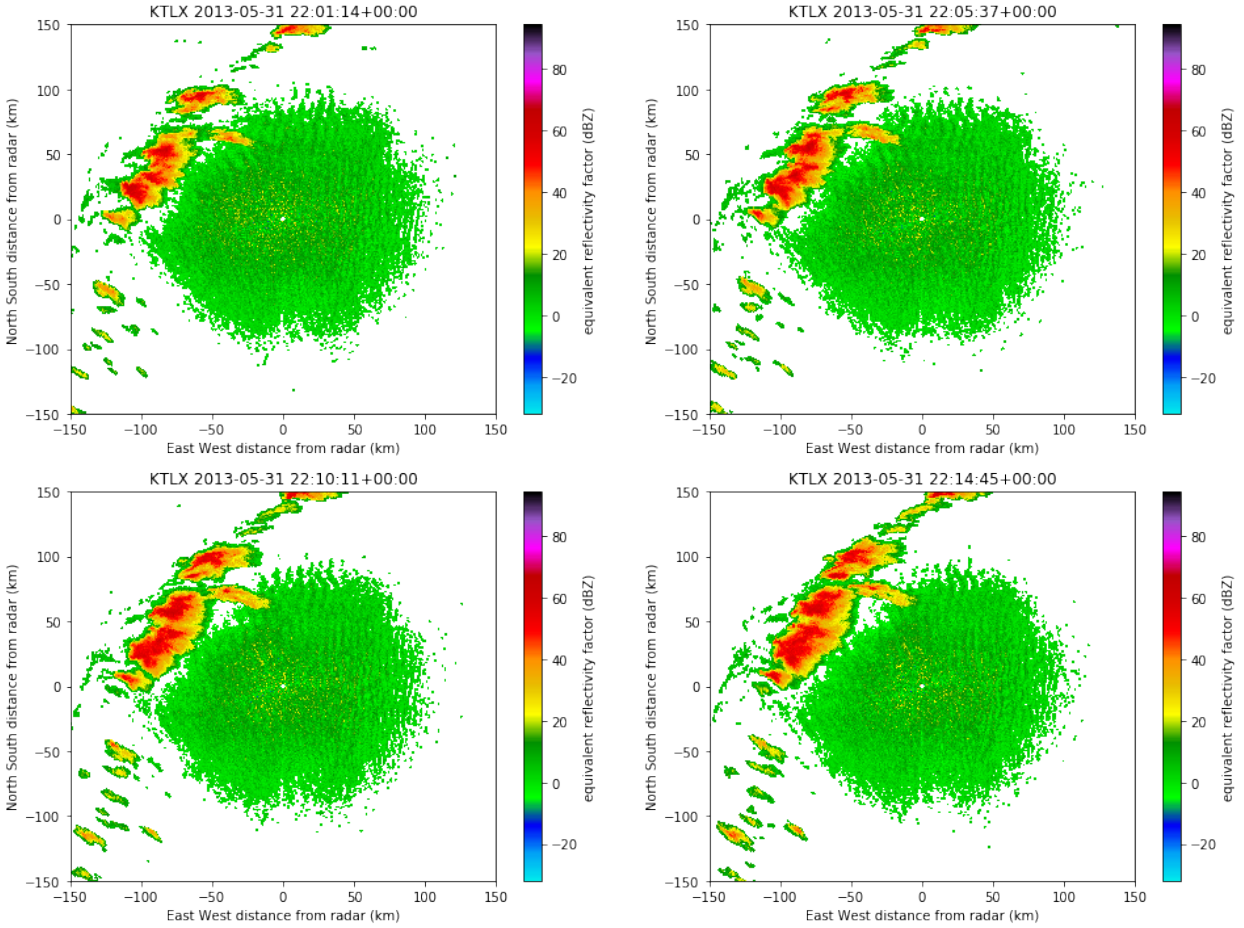
These objects have attributes containing metadata about the local file including local filepath, last_modified (on AWS), filename, volume scan time, and radar id.

There are two methods available on the LocalNexradFile to open the local file.

- open() - returns a file object. Be sure to close the file object when done.
- open_pyart() - if pyart is installed this will return a pyart Radar object.

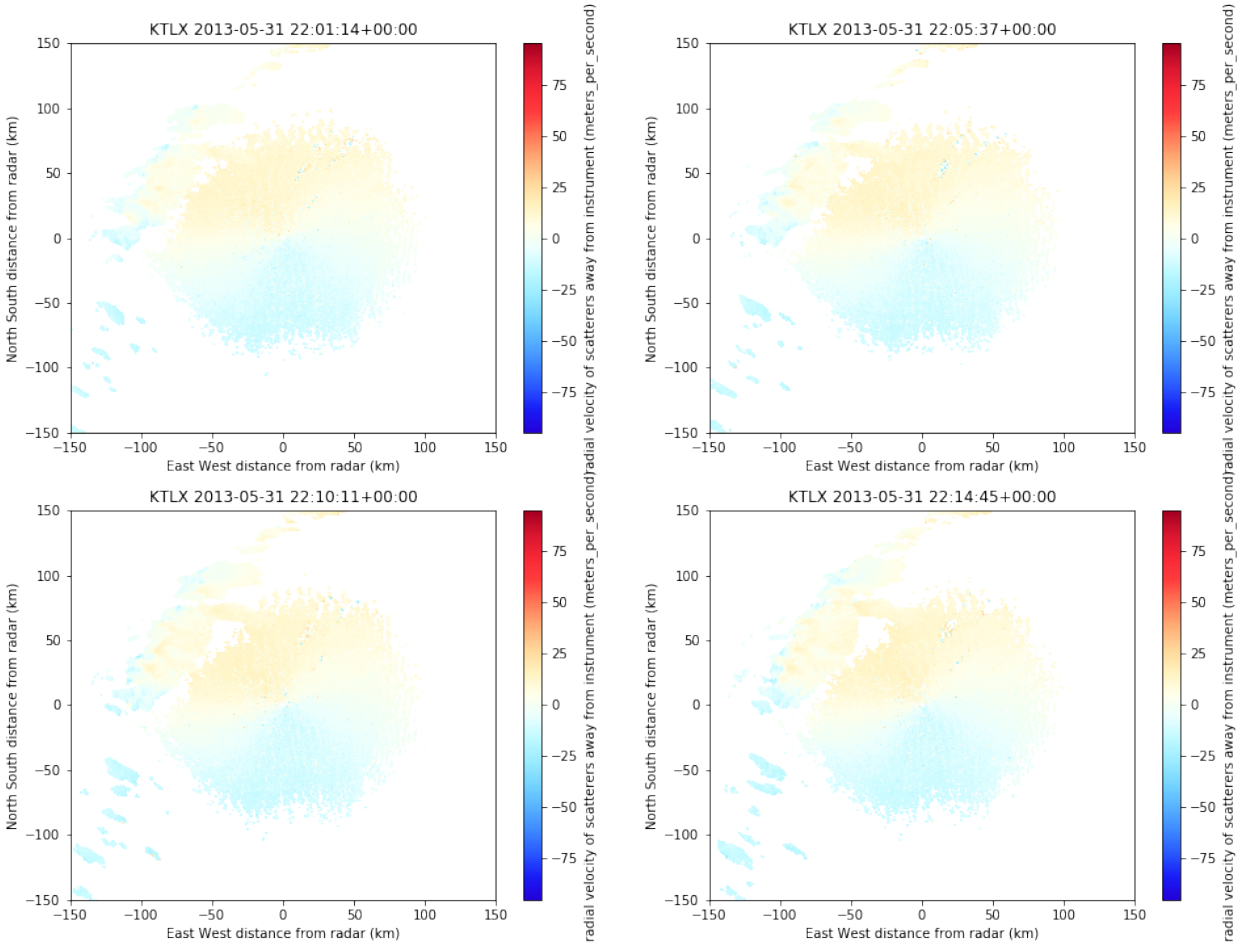
Let's look at an example of using pyart to open and plot our newly downloaded NEXRAD file from AWS. We will zoom into within 150km of the radar to see the storms a little better in these examples.

```
In [26]: fig = plt.figure(figsize=(16,12))
         for i,scan in enumerate(results.iter_success(),start=1):
             ax = fig.add_subplot(2,2,i)
             radar = scan.open_pyart()
             display = pyart.graph.RadarDisplay(radar)
             display.plot('reflectivity',0,ax=ax,title="{} {}".format(scan.radar_id,scan.scan_time))
             display.set_limits((-150, 150), (-150, 150), ax=ax)
```



Now lets plot velocity data for the same scans.

```
In [27]: fig = plt.figure(figsize=(16,12))
         for i,scan in enumerate(results.iter_success(),start=1):
             ax = fig.add_subplot(2,2,i)
             radar = scan.open_pyart()
             display = pyart.graph.RadarDisplay(radar)
             display.plot('velocity',1,ax=ax,title="{} {}".format(scan.radar_id,scan.scan_time))
             display.set_limits((-150, 150), (-150, 150), ax=ax)
```


KTLX 2013-05-31 22:01:14+00:00
KTLX 2013-05-31 22:05:37+00:00
KTLX 2013-05-31 22:10:11+00:00
KTLX 2013-05-31 22:14:45+00:00
North South distance from radar (km)
East West distance from radar (km)
radial velocity of scatterers away from instrument (meters_per_second)

CHAPTER 3

Indices and tables

- genindex
- modindex
- search

Index

A

AwsNexradFile (class in nexradaws.resources.awsnexradfile), 5

D

download() (nexradaws.NexradAwsInterface method), 5
DownloadResults (class in nexradaws.resources.downloadresults), 6

G

get_avail_days() (nexradaws.NexradAwsInterface method), 3
get_avail_months() (nexradaws.NexradAwsInterface method), 3
get_avail_radars() (nexradaws.NexradAwsInterface method), 4
get_avail_scans() (nexradaws.NexradAwsInterface method), 4
get_avail_scans_in_range() (nexradaws.NexradAwsInterface method), 5
get_avail_years() (nexradaws.NexradAwsInterface method), 3

I

iter_failed() (nexradaws.resources.downloadresults.DownloadResults method), 7
iter_success() (nexradaws.resources.downloadresults.DownloadResults method), 6

L

LocalNexradFile (class in nexradaws.resources.localnexradfile), 6

N

NexradAwsInterface (class in nexradaws), 3

O

open() (nexradaws.resources.localnexradfile.LocalNexradFile method), 6
open_pyart() (nexradaws.resources.localnexradfile.LocalNexradFile method), 6