NWSAPy

Brandon Molyneaux

Mar 12, 2022

CONTENTS

NWSAPy (APy, for short) is designed to be a pythonic approach to utilizing the National Weather Service API. The goals of the package are simple:

- **Maintain clean, simplistic, minimal, and consistent user-end code**
- **Construct URLs and request data on your behalf**
- **Minimize API knowledge overhead**

Here's a few brief examples of how easy it is to use the package:

Let's say we want to get all of the tornado warnings, then package all the information together in a dataframe:

```
from nwsapy import nwsapy

nwsapy.set_user_agent("Application Name", "youremail@domain.com or website")
active_tor_warnings = nwsapy.get_active_alerts(event = "Tornado Warning")
df = active_tor_warnings.to_dataframe()
```

Suppose we want to convert units to pint:

```
from nwsapy import nwsapy
from pint import UnitRegistry

nwsapy.set_user_agent("Application Name", "youremail@domain.com or website")
point = nwsapy.get_point(32.6099, -85.4808)  # Auburn, AL
point = point.to_pint(UnitRegistry())  # pass in your unit registry
print(point.distance) # 854.1731315087 meter
```

Implementation of the full API is in progress. The documentation will update on an as-needed basis to reflect this.

The National Weather Service API can be found here: https://www.weather.gov/documentation/services-web-api#/

CHAPTER

ONE

TABLE OF CONTENTS

1.1 Getting Started

1.1.1 Installation

You can install APy through `pip`:

```
pip install nwsapy
```

Once it's installed, go ahead and test it with this small script:

```
from nwsapy import api_connector

api_connector.set_user_agent("Application Name", "youremail@domain.com")
server_ping = api_connector.ping_server()

# Always a good idea to check to make sure an error didn't occur.
# There are times when a 400 or 500 error will occur.
if server_ping.has_any_request_errors:
    print(f"Error from server. Details: {server_ping}")
else:
print(server_ping.status)  # will print OK
```

This should give you:

```
OK
```

Note: The NWS API does require a `User Agent`, as this is a form of authentication. In accordance to their documentation:

> "A User Agent is required to identify your application. This string can be anything, and the more unique to your application the less likely it will be affected by a security event. If you include contact information (website or email), we can contact you if your string is associated to a security event. This will be replaced with an API key in the future."

APy gives functionality so that you're able to set this field (and other kinds of header information). See https://www.weather.gov/documentation/services-web-api#/ for more information regarding the User Agent and other header fields.

1.1.2 I want to...

Get watches and warnings

The NWS API provides a way for you to get various alerts. The table below provides a reference as to which methods to call using NWSAPy:

Description	NWSAPy Method
Get all of the alerts.	*get_alerts()*
Get the current active alerts.	*get_active_alerts()*
Get all the types of alerts.	*get_alert_types()*
Get active alerts by their ID.	*get_alert_by_id()*
Get the number of active alerts.	*get_alert_count()*
Get the active alerts by zone.	*get_alert_by_zone()*
Get the active alerts by area.	*get_alert_by_area()*
Get the active alerts by marine region.	*get_alert_by_marine_region()*

Get definitions of words I don't understand!

No worries, the NWS API does provide a glossary for you: *get_glossary()*.

Get metadata about a specific lat/lon point

Use *get_point()*.

1.1.3 Rate Limit

When using this package, you may encounter the rate limit. This is a limit in which the maintainers of the API have set, but is not publicly known. If you encounter this, wait a few seconds before trying again.

1.1.4 Data Validation

If there are any parameters for any `nwsapy` method, the parameters will be validated against *Data Validation Tables* listed in the documentation.

Important: If you were to use NWSAPy in an application that you plan on deploying, you want to ensure that the parameters are formatted properly before calling any `nwsapy` functions. Data Validation Errors and Parameter Type Errors should only appear during development and will stop your code from running.

For example, if we were run the following code:

```
from nwsapy import api_connector

api_connector.set_user_agent("NWSAPy", "your_email@email.com or website")
area_alert = api_connector.get_alert_by_area('FL')
```

We would retrieve all of the active alerts from Florida. However, if we were to attempt to run this code:

```
from nwsapy import api_connector

api_connector.set_user_agent("NWSAPy", "your_email@email.com or website")
area_alert = api_connector.get_alert_by_area('Flo-rida')
```

An error would be raised:

```
>>> area_alert = api_connector.get_alert_by_area('Flo-rida')
nwsapy.core.errors.DataValidationError: Invalid data input: '['Flo-rida']'. See␣
↪documentation for valid inputs.
```

This is because `Flo-rida` isn't in the associated Data Validation Table for this parameter.

1.2 Introduction to NWSAPy

1.2.1 Endpoints Dashboard

The below dashboard shows which endpoints have been implemented in the corresponding verison of NWSAPy.

Table 1: Legend

Color	Status
Green	Currently Implemented
Yellow	Will be Implemented in the Next Release
Red	Not Currently Implemented

1.2.2 API Reference

NWSAPy User API Reference

class nwsapy.**NWSAPy**

get_active_alerts(***kwargs*)

Returns an active alerts object containing all active alerts. This object is comprised of IndividualAlerts.

Endpoint: `/alerts/active`
Description: Returns all currently active alerts.

Parameters

- **area** (*str or list[str]*) – The land or marine region where the alert is in. Either a 2 letter abbreviation (i.e. "FL") or the full state name (i.e. "Florida") *Data Validation Table*
- **certainty** (*str or list[str]*) – The certainty of the alert. *Data Validation Table*
- **event** (*str or list[str]*) – The type of alert (i.e. Severe Thunderstorm Warning, etc). *Data Validation Table*
- **limit** (*int*) – The number of alerts to return at most. Will only retrieve the first n alerts.
- **message_type** (*str or list[str]*) – *Data Validation Table*

- **point** (*list[float]*) – A tuple or list containing a latitude and longitude pair.
- **region** (*str or list[str]*) – A marine region where the alert resides. *Data Validation Table*
- **region_type** (*str*) – The type of region where the alert resides. *Data Validation Table*
- **severity** (*str or list[str]*) – The severity level of the alert. *Data Validation Table*
- **status** (*str or list[str]*) – The status of the alert. *Data Validation Table*
- **urgency** (*str or list[str]*) – The urgency of the alert. *Data Validation Table*
- **zone** (*str or list[str]*) – The NWS zone of the alert. Note this has no validation checks, so a 404 error can occur.

Returns An object containing information from the `alerts/active` endpoint.

Return type nwsapy.endpoints.alerts.ActiveAlert

get_alert_by_area(*area*)

Retrieves alerts by a given area (state or marine).

Endpoint: `alerts/active/area/{area}`
Description: Retrieves alerts for the given area (state or marine area)

Parameters **area** (*str*) – 2 letter state abbreviation or full state name (i.e. Florida) *Data Validation Table*

Returns An object containing information about the specified alert.

Return type *nwsapy.endpoints.alerts.AlertByArea*

get_alert_by_id(*id*)

Retrieves an alert by ID from the `alerts/active/{id}` endpoint.

Endpoint: `alerts/active/{id}`
Description: Returns a specific alert

Parameters **id** (*string*) – The ID of an alert.

Returns An object containing information about the specified alert.

Return type *nwsapy.endpoints.alerts.AlertById*

get_alert_by_marine_region(*marine_region*)

Retrieves alerts by a specific marine region.

Endpoint: `alerts/active/region/{regionId}`
Description: Returns active alerts for the given marine region

Parameters **marine_region** (*str*) – A 2 letter marine region. *Data Validation Table*

Returns An object containing information about alerts in the specified marine region.

Return type nwsapy.endpoints.alerts.AlertByRegion

get_alert_by_zone(*zone*)
Retrieves alerts by the 6 character NWS zone or county. Note that this method does *not* have any data validation checks as of v1.0.0.

Endpoint: `/alerts/active/zone/{zoneId}`
Description: Returns active alerts for the given NWS public zone or county

Parameters **zone** (*str*) – A 6 character NWS zone or county (ex: BAC222)

Returns An object containing information about the NWS public zone or county

Return type *nwsapy.endpoints.alerts.AlertByZone*

get_alert_count()
Gets the number of land, marine, and total active alerts. Also provides a dictionary of the number of alerts by areas (states), zones, and marine regions.

Endpoint: `/alerts/active/count` Description: Returns info on the number of active alerts.

Returns An object containing the number of alerts for certain parameters.

Return type *nwsapy.endpoints.alerts.AlertCount*

get_alert_types()
Retrieves a list of the alert types that the National Weather Service puts out.

Endpoint: `/alerts/types` Description: Returns a list of alert types.

Note: NWSAPy has this built in as a data validation table. To retrieve:

```
from nwsapy.services.validation import valid_products
products = valid_products()
```

Using this will allow for your code to be run quicker, as it retrieves a list compared to a NWS API request.

Returns An object containing information about the alert types.

Return type nwsapy.endpoints.alert.AlertType

get_alerts(***kwargs*)
Returns an alerts object with the previous 500 alerts. Note that this is the maximum value and also the default.

Endpoint: `/alerts`
Description: Returns all alerts.

Parameters

- **area** (*str or list[str]*) – The land or marine region where the alert is in. Either a 2 letter abbreviation (i.e. "FL") or the full state name (i.e. "Florida") *Data Validation Table*
- **certainty** (*str or list[str]*) – The certainty of the alert. *Data Validation Table*
- **event** (*str or list[str]*) – The type of alert (i.e. Severe Thunderstorm Warning, etc). *Data Validation Table*

- **limit** (*int*) – The number of alerts to return at most. Will only retrieve the first n alerts.
- **message_type** (*str or list[str]*) – *Data Validation Table*
- **point** (*list[float]*) – A tuple or list containing a latitude and longitude pair.
- **region** (*str or list[str]*) – A marine region where the alert resides. *Data Validation Table*
- **region_type** (*str*) – The type of region where the alert resides. *Data Validation Table*
- **severity** (*str or list[str]*) – The severity level of the alert. *Data Validation Table*
- **status** (*str or list[str]*) – The status of the alert. *Data Validation Table*
- **urgency** (*str or list[str]*) – The urgency of the alert. *Data Validation Table*
- **zone** (*str or list[str]*) – The NWS zone of the alert. Note this has no validation checks, so a 404 error can occur.

Returns An object containing information from the `alerts/active` endpoint.

Return type nwsapy.endpoints.alerts.ActiveAlert

`get_glossary()`

Makes a request to the */glossary* endpoint in the API and returns a glossary object containing information about the terms listed in the glossary.

Endpoint: `/glossary`
Description: Glossary terms

Returns An object containing information from the */glossary* endpoint.

Return type *nwsapy.endpoints.glossary.Glossary*

`get_point(lat, lon)`

Makes a request to the */point* endpoint in the API and returns a point object containing metadata about the given lat/lon.

Endpoint: `/points/{point}`
Description: Returns metadata about a given latitude/longitude point.

Returns An object containing information from the */point* endpoint.

Return type *nwsapy.endpoints.point.Point*

`make_request(url)`

Makes a request to the NWS API with a given URL. This method allows you to have full control over the data without the functionality and organization of any NWSAPy `get_*` methods.

Note: This does not have any kind of data validation checks and does not handle any errors on behalf of you.

Parameters **url** (*string*) – The URL to make a request to the NWS API.

Returns A response object containing the request for the query.

Return type request.Response

`ping_server()`
Pings the server for integrity and/or testing.

Returns ServerPing object

Return type nwsapy.entrypoint.ServerPing

`set_user_agent(app_name, contact)`
Sets the User-Agent in header for requests. This should be unique to your application.

From the NWS API documentation: "A User Agent is required to identify your application. This string can be anything, and the more unique to your application the less likely it will be affected by a security event. If you include contact information (website or email), we can contact you if your string is associated to a security event."

Link: https://www.weather.gov/documentation/services-web-api#/

Parameters

- **app_name** (*str*) – The name of your application.
- **contact** (*str*) – The contact email/website. This is needed for API authentication.

Alert Endpoint Reference

Alerts

Every `Alert` object contains at least 3 methods:

- `to_df`
- `to_dict`
- `to_pint`

If the method is not implemented (that is, nothing happens when you call it), it will be explicitly marked as "Not implmented for this endpoint".

Most of the `Alert` objects contain *`IndividualAlert`* objects, which is documented below this section.

`class nwsapy.endpoints.alerts.Alerts`

`to_df()`
Returns the values of the alerts in a pandas dataframe structure.

Returns Dataframe of the values of the alerts.

Return type pandas.DataFrame

`to_dict()`
Returns the alerts in a dictionary format, where the keys are numbers which map to an individual alert.

Returns Dictionary containing the values of the active alerts.

Return type dict

`to_pint()`
Not implemented for this endpoint.

`class nwsapy.endpoints.alerts.ActiveAlerts`

to_df()
Returns the values of the alerts in a pandas dataframe structure.

Returns Dataframe of the values of the alerts.

Return type pandas.DataFrame

to_dict()
Returns the alerts in a dictionary format, where the keys are numbers which map to an individual alert.

Returns Dictionary containing the values of the active alerts.

Return type dict

to_pint()
Not implemented for this endpoint.

class nwsapy.endpoints.alerts.AlertById

to_df()
Returns the values of the alerts in a pandas dataframe structure.

Returns Dataframe of the values of the alerts.

Return type pandas.DataFrame

to_dict()
Returns the alerts in a dictionary format, where the keys are numbers which map to an individual alert.

Returns Dictionary containing the values of the active alerts.

Return type dict

to_pint()
Not implemented for this endpoint.

class nwsapy.endpoints.alerts.AlertByArea

to_df()
Returns the values of the alerts in a pandas dataframe structure.

Returns Dataframe of the values of the alerts.

Return type pandas.DataFrame

to_dict()
Returns the alerts in a dictionary format, where the keys are numbers which map to an individual alert.

Returns Dictionary containing the values of the active alerts.

Return type dict

to_pint()
Not implemented for this endpoint.

class nwsapy.endpoints.alerts.AlertByZone

to_df()
Returns the values of the alerts in a pandas dataframe structure.

Returns Dataframe of the values of the alerts.

Return type pandas.DataFrame

to_dict()
Returns the alerts in a dictionary format, where the keys are numbers which map to an individual alert.

Returns Dictionary containing the values of the active alerts.

Return type dict

to_pint()
Not implemented for this endpoint.

class nwsapy.endpoints.alerts.AlertByMarineRegion

to_df()
Returns the values of the alerts in a pandas dataframe structure.

Returns Dataframe of the values of the alerts.

Return type pandas.DataFrame

to_dict()
Returns the alerts in a dictionary format, where the keys are numbers which map to an individual alert.

Returns Dictionary containing the values of the active alerts.

Return type dict

to_pint()
Not implemented for this endpoint.

class nwsapy.endpoints.alerts.AlertByType

to_df()
Not implemented for this endpoint.

to_dict()
Not implemented for this endpoint.

to_pint()
Not implemented for this endpoint.

class nwsapy.endpoints.alerts.AlertCount

to_df()
Not implemented for this endpoint.

to_dict()
Not implemented for this endpoint.

to_pint()
Not implemented for this endpoint.

Individual Alerts

Individual Alert objects are the backbone to most `Alert` objects. Each indivieual Alert object contains attributes.

class nwsapy.endpoints.alerts.**IndividualAlert**(*alert_list*)

effective_after(*other*)
Method to compare effective times. All times are compared in UTC.

Parameters **other** (alerts.IndividualAlert) – Another individual alert object.

Returns True if the alert was effective after `other`.

Return type bool

effective_before(*other*)
Method to compare effective times. All times are compared in UTC.

Parameters **other** (alerts.IndividualAlert) – Another individual alert object.

Returns True if the alert was effective before `other`.

Return type bool

ends_after(*other*)
Method to compare end times. All times are compared in UTC.

Parameters **other** (alerts.IndividualAlert) – Another individual alert object.

Returns True if the alert ends after `other`.

Return type bool

ends_before(*other*)
Method to compare end times. All times are compared in UTC.

Parameters **other** (alerts.IndividualAlert) – Another individual alert object.

Returns True if the alert ends before `other`.

Return type bool

expires_after(*other*)
Method to compare expire times. All times are compared in UTC.

Parameters **other** (alerts.IndividualAlert) – Another individual alert object.

Returns True if the alert expires before `other`.

Return type bool

expires_before(*other*)
Method to compare expire times. All times are compared in UTC.

Parameters **other** (alerts.IndividualAlert) – Another individual alert object.

Returns True if the alert expires before `other`.

Return type bool

onset_after(*other*)
Method to compare onset times. All times are compared in UTC.

Parameters **other** (alerts.IndividualAlert) – Another individual alert object.

Returns True if the alert was onset after `other`.

Return type bool

`onset_before`(*other*)
Method to compare onset times. All times are compared in UTC.

Parameters **other** (alerts.IndividualAlert) – Another individual alert object.

Returns True if the alert was onset before other.

Return type bool

`sent_after`(*other*)
Method to compare sent times. All times are compared in UTC.

Parameters **other** (alerts.IndividualAlert) – Another individual alert object.

Returns True if the alert was after before other.

Return type bool

`sent_before`(*other*)
Method to compare sent times. All times are compared in UTC.

Parameters **other** (alerts.IndividualAlert) – Another individual alert object.

Returns True if the alert was sent before other.

Return type bool

`to_dict`()
Converts all of the attributes to a dictionary.

Returns A dictionary containing all of the attributes of the object.

Return type dict

Glossary

`class nwsapy.endpoints.glossary.Glossary`

`to_df`()
Returns the values of the glossary in a pandas dataframe structure.

Returns Dataframe of the values of the glossary.

Return type pandas.DataFrame

`to_dict`()
Returns the glossary in a dictionary format.

Returns Dictionary containing the values of the glossary.

Return type dict

`to_pint`()
Not implemented for this endpoint.

Point API

class nwsapy.endpoints.point.**Point**

to_df()
Not implemented for this endpoint.

to_dict() → dict
Returns a dictionary with all of the attributes to the class.

Returns Dictionary of the attributes of the class.

Return type dictionary

to_pint(*unit_registry: pint.registry.UnitRegistry*) → object
Returns a new self object with units using Pint. It does NOT update in-place.

Parameters **unit_registry** (*pint.UnitRegistry*) – Your unit registry used in your application.

Returns Dictionary with values converted to pint units.

Return type dictionary

1.2.3 Some key things to know

- NWSAPy does *not* handle fetching data in the NWS servers. That is, it only interfaces with the National Weather Service API, and their API handles `GET` requests for their server.
- All data from the NWS API is done through `get_*(kwargs)` functions. For example, `get_active_alerts(event = "Tornado Warning")` will get all active Tornado warnings.
- Not all endpoints are currently implemented. See the *Endpoints Dashboard* to see what is currently implemented.
- When passing in parameters to `nwsapy` methods (see: second point in this section for an example), ensure that it is spelled *exactly* the same way, including upper case and lower case methods. These values can be found in their respective *Data Validation Tables*.

1.2.4 An Example, Explained

NWSAPy is designed to be a simple, yet pythonic way to interface with the National Weather Service API. This section outlines how to use NWSAPy, generally speaking, for your application by stepping through a simple example. More details for each individual `get_*(kwargs)` method can be found in their respective documentation areas.

Starting with a simple example:

```
from nwsapy import api_connector

api_connector.set_user_agent('Application Name', 'Contact information')
server_ping = api_connector.ping_server()
```

This will print `OK` if the request was successful. Breaking it down:

```
from nwsapy import api_connector
```

`api_connector` is the object that's being used to interface with the package. That is, all methods that are called are encapsulated in `api_connector`.

Note: For those who have used NWSAPy before v1.0.0, this was originally `from nwsapy import nwsapy`. You are still able to do this, but this will be removed in a future version.

The next line is `set_user_agent`:

```
api_connector.set_user_agent('Application Name', 'Contact Information')
```

The user agent gets put into the header information when making a request. This is a field that the maintainers of the National Weather Service API would like to have in case your request is associated with a security event. `Application Name` should be unique to your application, and `Contact Information` can be a website or an email. Without this line, NWSAPy will warn you and let you know that you should use this when making any kind of request to the National Weather Service API.

Important: NWSAPy does *not* store the information to be used outside of making a request to the National Weather Sevice API. This information is passed into a method in the `requests` module (specifically, `requests.get()`).

Following this:

```
server_ping = api_connector.ping_server()
```

This pings the server and returns an object that you are able to utilize. In this case, it is a `nwsapy.endpoints.server_ping.ServerPing` object. See the documentation on `ServerPing` to see the attributes of the `ServerPing` object obtained from `get_server_ping()`.

this is where NWSAPy is able to be leveraged for your project: it organizes all of the data coming from the Natoinal Weather Service API in a pythonic manner through `get_*(kwargs)` methods. It also keeps code that you create using this package clean and straight-forward.

For instance, if we were to iterate through the `ServerPing` object:

```
for key, value in server_ping:
    print(f'Key: {key}, Value: {value}')
```

We would see:

```
Key: status, Value: OK
```

Under the hood (in this instance), it's iterating through a dictionary. There are some objects that have a list under the hood that it is iterating through. Details of this can be found in the respective `get_*(kwargs)` method.

1.3 Data Validation Tables

Data Validation Tables are designed to provide a snapshot of the parameters that are valid and will not kick back an error from the NWS API.

For instance, suppose you are getting all active tornado alerts:

```
tors = api_connector.get_active_alerts(event = 'Tornado')
```

This will not work, as `Tornado` is not in the *Events* Data Validation Table. NWSAPy will raise a `DataValidationError` and tell you that it is not found in the respective Data Validation Table. However, `Tornado Warning` and `Tornado Watch` are in the Data Validation Table associated with this parameter, so instead you'll read in your parameter as such:

```
tors = api_connector.get_active_alerts(event = ['Tornado Warning', 'Tornado Watch'])
```

This section provides all the Data Validation Tables, as well as what methods they are used in to validate your input. In addition, each parameter listing in the NWSAPy API reference has a link to the respective table it uses to check to ensure that the data is valid.

Warning: You *must* pay attention to capitalization and spellig when building parameters for your `get_*` functions, as the API may say it's invalid. NWSAPy will, in the very near future, will handle any capitalization errors and allow for a more robust parameter system (for a lack of better words).

1.3.1 Area

This data validation table is used in the following methods:

- *get_active_alerts()*
- *get_alerts()*
- *get_alert_by_area()*

Table 2: Valid Areas

AL	Alabama	MA	Massachussetts	TX	Texas
AK	Alaska	MI	Michigan	UT	Utah
AS	American Samoa	MN	Minnesota	VT	Vermont
AR	Arkansas	MS	Mississippi	VI	Virgin Islands
AZ	Arizona	MO	Missouri	VA	Virginia
CA	California	MT	Montana	WA	Washington
CO	Colorado	NE	Nebraska	WV	West Virginia
CT	Connecticut	NV	Nevada	WI	Wisconsin
DE	Deleware	NH	New Hampshire	WY	Wyoming
DC	District of Columbia	NJ	New Jersey	PZ	Eastern North Pacific Ocean
FL	Florida	NM	New Mexico	PK	North Pacific Ocean Near Alaska
GA	Georiga	NY	New York	PH	Central Pacific Ocean
GU	Guam	NC	North Carolina	PS	South Central Pacific Ocean
HI	Hawaii	ND	North Dakota	PM	Western Pacific Ocean
ID	Idaho	OH	Ohio	AN	Northwest North Atlantic Ocean
IL	Illinois	OK	Oklahoma	AM	West North Atlantic Ocean
IN	Indiana	OR	Oregon	GM	Gulf of Mexico
IA	Iowa	PA	Pennsylvania	LS	Lake Superior
KS	Kansas	PR	Puerto Rico	LM	Lake Michigan
KY	Kentucky	RI	Rhode Island	LH	Lake Huron
LA	Louisiana	SC	South Carolina	LC	Lake St. Clair
ME	Maine	SD	South Dakota	LE	Lake Erie
MD	Maryland	TN	Tennessee	LO	Lake Ontario

1.3.2 Certainty

This data validation table is used in the following methods:

- *get_active_alerts()*
- *get_alerts()*

Table 3: Valid Certainty Levels

likely	observed	possible	unlikely	unknown

1.3.3 Marine Regions

This data validation table is used in the following methods:

- *get_active_alerts()*
- *get_alerts()*
- *get_alert_by_marine_region()*

Table 4: Valid Marine Regions

AL	Alaska Region
AT	Atlantic Region
GL	Great Lakes
GM	Gulf of Mexico
PA	Pacific Region
PI	Pacific Islands

1.3.4 Message Types

This data validation table is used in the following methods:

- *get_active_alerts()*
- *get_alerts()*

Table 5: Valid Message Types

Alert	Cancel	Update

1.3.5 Products (Event)

This data validation table is used in the following methods:

- *get_active_alerts()*
- *get_alerts()*

Table 6: Valid Alert Products (Event)

911 Telephone Outage Emergency	Extreme Cold Watch	High Wind Watch	Small Cr
Administrative Message	Extreme Fire Danger	Hurricane Force Wind Warning	Small Cr
Air Quality Alert	Extreme Wind Warning	Hurricane Force Wind Watch	Small St

Table 6 – continued from previous page

Air Stagnation Advisory	Fire Warning	Hurricane Local Statement	Snow Sq
Arroyo And Small Stream Flood Advisory	Fire Weather Watch	Hurricane Warning	Special M
Ashfall Advisory	Flash Flood Statement	Hurricane Watch	Special W
Ashfall Warning	Flash Flood Warning	Hydrologic Advisory	Storm Su
Avalanche Advisory	Flash Flood Watch	Hydrologic Outlook	Storm Su
Avalanche Warning	Flood Advisory	Ice Storm Warning	Storm W
Avalanche Watch	Flood Statement	Lake Effect Snow Advisory	Storm W
Beach Hazards Statement	Flood Warning	Lake Effect Snow Warning	Test
Blizzard Warning	Flood Watch	Lake Effect Snow Watch	Tornado
Blizzard Watch	Freeze Warning	Lake Wind Advisory	Tornado
Blowing Dust Advisory	Freeze Watch	Lakeshore Flood Advisory	Tropical
Blowing Dust Warning	Freezing Fog Advisory	Lakeshore Flood Statement	Tropical
Brisk Wind Advisory	Freezing Rain Advisory	Lakeshore Flood Warning	Tropical
Child Abduction Emergency	Freezing Spray Advisory	Lakeshore Flood Watch	Tropical
Civil Danger Warning	Frost Advisory	Law Enforcement Warning	Tsunami
Civil Emergency Message	Gale Warning	Local Area Emergency	Tsunami
Coastal Flood Advisory	Gale Watch	Low Water Advisory	Tsunami
Coastal Flood Statement	Hard Freeze Warning	Marine Weather Statement	Typhoon
Coastal Flood Warning	Hard Freeze Watch	Nuclear Power Plant Warning	Typhoon
Coastal Flood Watch	Hazardous Materials Warning	Radiological Hazard Warning	Typhoon
Dense Fog Advisory	Hazardous Seas Warning	Red Flag Warning	Urban A
Dense Smoke Advisory	Hazardous Seas Watch	Rip Current Statement	Volcano
Dust Advisory	Hazardous Weather Outlook	Severe Thunderstorm Warning	Wind Ad
Dust Storm Warning	Heat Advisory	Severe Thunderstorm Watch	Wind Ch
Earthquake Warning	Heavy Freezing Spray Warning	Severe Weather Statement	Wind Ch
Evacuation - Immediate	Heavy Freezing Spray Watch	Shelter In Place Warning	Wind Ch
Excessive Heat Warning	High Surf Advisory	Short Term Forecast	Winter S
Excessive Heat Watch	High Surf Warning	Small Craft Advisory	Winter S
Extreme Cold Warning	High Wind Warning	Small Craft Advisory For Hazardous Seas	Winter W

1.3.6 Region Type

This data validation table is used in the following methods:

- *get_active_alerts()*
- *get_alerts()*

Table 7: Valid Region Types

Land	Marine

1.3.7 Severity

This data validation table is used in the following methods:

- *get_active_alerts()*
- *get_alerts()*

Table 8: Valid Severity

Extreme	Minor	Moderate	Severe	Unknown

1.3.8 Status

This data validation table is used in the following methods:

- *get_active_alerts()*
- *get_alerts()*

Table 9: Valid Status

Actual	Exercise	Draft	System	Test

1.3.9 Urgency

This data validation table is used in the following methods:

- *get_active_alerts()*
- *get_alerts()*

Table 10: Valid Urgency

Expected	Future	Immediate	Past	Unknown

CHAPTER

TWO

ADVANTAGES OF USING NWSAPY

- **Clean and Simplistic Code** - The syntax is very english-like.
- **No worries about JSON**. NWSAPy takes care of anything JSON-related, including formats (GeoJSON, JSON-LD, etc).
- **No worries about URLs**. Similar to the Django ORM, you're able to make a request without ever writing code to make a request.
- **404 Error Minimization.** This is handled through data validation checks, as well as handling URL construction.
- **Response errors are handled.** Response errors are handled appropriately.

CHAPTER

THREE

DEPENDENCIES

NWSAPy has minimal dependencies, with core functionality being pure python:

- shapely
- pandas
- numpy
- requests
- pint

CHAPTER

FOUR

CONTACT

It motivates me to continue support and development for this package if I hear from the community. If you have questions or comments relating to the package, you can tweet at me: @WxBDM. You can also send me an email at brand.molyn@gmail.com. If a bug is identified, open an issue on GitHub. Please provide steps on how to recreate the issue with an example. If you'd like to contribute, please issue a pull request.

CHAPTER

FIVE

IMPORTANT LINKS

- GitHub
- National Weather Service API Documentation
- National Weather Service API
- National Weather Service API Discussion

CHAPTER

SIX

INDICES AND TABLES

- genindex
- modindex
- search

INDEX