Note that most of these functions are get requests which means these functions do not return anything.

All of the functions that update nodes of a placement take in the placement_id parameter. This parameter is the id of the placement you want to update and must be passed as a string.

Function: FW_Auth

Params: username, password, filetype

This function is how you generate your auth token once you have been granted access by the freewheel support team. Enter your username and password as the parameters of the function. For filetype, you can choose from xml or json, if you are using a v3 API pass 'xml' and if you are using a v4 API pass 'json'. Almost every single function in this module has headers as the parameter so create a variable called headers and set it equal to FW_Auth.

Example:

import FreeWheel4py as fw

headers = fw.FW_Auth(username, password, 'xml')

Function: create_placement

Params = headers, io, name

This Function creates a blank placement with a name. all you need to do is specify the headers, then pass in the insertion order id as a string and then the name you want for the placement. The function returns the placement id of the placement it created.

Example:

import FreeWheel4py as fw

headers = fw.FW_Auth(username, password, 'xml')

io = '12345'

name = 'test placement'

fw.create_placement(headers,io,name)

Function: update_schedule

Params = start_date, end_date, headers, placement_id

This Function updates the flighting of a placement and its default time zone is New York. Start and end date must be a string and look like M-d-Y.

Example:

```
import FreeWheel4py as fw

headers = fw.FW_Auth(username, password, 'xml')

start_date = '9-1-2022'

end_date = '10-31-2022'

placement_id = '12345'

fw.update_placement(start_date, end_date,headers,placement_id)
```

Function: update_pricing

Params: price_model, flat_fee_amount, headers, placement_id

This function updates the pricing of a placement. The price_model parameter is to determine which pricing structure the placement will use. For flat fee, only enter that amount if the price model is 'FLAT_FEE_SPONSORSHIP'. If its not, just pass in 0.

Example:

```
import FreeWheel4py as fw

headers = fw.FW_Auth(username, password, 'xml')

placement_id = '12345'

update_pricing('ACTUAL_ECPM',0,headers,placement_id)
```

Function: update_audience_targeting

Params: headers, list_of_ids, relation_in_sets, relation_between_sets, exclude, placement_id

This function updates the audience targeting of a placement. This function is very useful, but the parameters are slippery. If you only have one set, list_of_ids must be a list of audience segments ids for example, list_of_ids = ['123','456','789']. If you have multiple sets, list_of_ids must be a list of lists and inside each list are audience ids. For example, list_of_ids = [['123],['456']]. The relation_in_sets parameter tells FreeWheel if the audiences should be AND'ed or OR'ed together and must be passed in a list of Boolean values. For example, if you have two sets and you want every single audience in set one to be AND'ed together and want every single audience in set two to be OR'ed together you would set relation_in_sets equal to ['AND','OR]. Also, and/or must be capitalized. The relation_between_sets parameter tells FreeWheel if you want your sets to be AND'ed or OR'ed together. This parameter is only present when you have more than one set. If you have one set, you can just pass in 0, if you have two sets pass in the Boolean logic in a list. So if you want set one and two to be OR'ed together you would set relation_between_sets equal to ['OR']. If you have three sets that all needed to be AND'ed together, you set relation_between_sets equal to ['AND','AND']. Note, the number of relation_between_sets has to be equal to 1 – the number of sets in a placement. The exclude parameter tells FreeWheel the

audience that you want to exclude from the placement. Pass in a list of audience ids to exclude them from a placement.

Example:

```
import FreeWheel4py as fw

headers = fw.FW_Auth(username, password, 'xml')

relation_in_set = ['OR','OR','AND']

relation_between_sets = ['AND','OR']

list_of_ids = [['566467'],['567008'],['568520']]

exclude = ['567328']

placement_id = '12345'

fw.update_audience_targeting(headers,list_of_ids,relation_in_set,relation_between_sets,exlcude,placement_id)
```

Function: update_content_targeting

Params: headers, list_of_sets, relation_in_set, relation_between_sets, exclude, placement_id

This function updates the content targeting of a placement and is very similar to the update_audience_targeting function. List of sets must be set equal to a list containing multiple dictionaries, the number of dictionaries depends on how many sets you want (max 3). The dictionary inside the list is dependent on the items that you want to update. For example, imagine you want the placement to have two sets. Set one has two video series and one site group and set two has one video group, list_of_sets would look like [{'series' : ['123','456'], 'site_group' : '123'}, {'video_group' : '123'}]. Note, if you have one id, the key of the dictionary inside the list must be string which is the items id, if you have more than one, it must be a list of id's as strings. the relation_in_set and relation_between_sets acts the same way as update_audience_targeting, so please refer to that functions documentation. The exclude parameter must be a list containing a dictionary, similar to the list_of_sets parameter.

Example:

```
import FreeWheel4py as fw

headers = fw.FW_Auth(username, password, 'xml')

relation_in_set = ['OR','AND','AND']

relation_between_sets = ['AND','OR']
```

list_of_sets = [{'video_group' : '94545496656' , 'site_group': ['78823232201', '8723238769']},{'video_group' : '20717543625'},{'series' : '302934252328'}]

exclude = [{'video_group': '1872244232023'}]

placement_id = '6224242185583'

fw.update_content_targeting(headers,list_of_sets,relation_in_set,relation_between_sets,exclude,placement_id)

Function: attach_ad_units

Params: headers, placement_id, ad_units, prices

This function updates the placements ad units and their prices. The ad_units parameter must be a list of ad_unit_ids , even if you only have one ad unit. And prices parameter is a list of prices for each ad unit. The number of ad_units and prices must be equal to each other.

Example:

import FreeWheel4py as fw

headers = fw.FW_Auth(username, password, ‘xml’)

ad_units = ['59106','59104']

prices = ['40','40']

placement_id = ‘12345’

fw. attach_ad_units(headers,placement_id,ad_units,prices)

Function: attach creatives

Params: headers, placement_id, creatives

This function attaches creatives to a placement’s ad unit node so there must be ad units attached to a placement before you can use this function. This function also assumes that every single ad unit will have the same creatives attached to its node. The creatives parameter must be a list of creative ids, even if you only have one creative.

Example:

import FreeWheel4py as fw

headers = fw.FW_Auth(username, password, ‘xml’)

placement_id = ‘12345’

creatives = [‘44444’]

fw. attach_creatives(headers,placement_id,creatives)

Function: update_platform_targeting

Parameters: headers, list_of_devices, placement_id

This function is used to update the platform targeting of a placement. Passing a list of all device ids and the placement id of the placement you want to update.

Example:

import FreeWheel4py as fw

headers = fw.FW_Auth(username, password, ‘xml’)

placement_id = ‘12345’

list_of_devices = ['27800', '118222', '118214', '1233207']

update_platform_targeting(headers,list_of_devices,placement_id)

Function: update_geo_targeting

Parameters: headers, inc_exc, geo_type, list_of_codes, placement_id

This function updates the geography targeting of a placement. Pass in what type of geography targeting you to target by setting geo_type equal to dma, state, or zip_code. Set inc_exc equal to ‘include’ if you want to include the geography or ‘exclude’ if you want to exclude it. List_of_codes must be a list of codes of what ever geo you want to target.

Example:

import FreeWheel4py as fw

headers = fw.FW_Auth(username, password, ‘xml’)

placement_id = ‘12345’

inc_exc = ‘exclude’

geo_type = ‘dma’

list_of_codes = [‘1,’66’,’34’,’97]

fw.update_geo_targeting(headers,inc_exc,geo_type,list_of_codes,placement_id)

Function: update_daypart_targeting

Parameters: headers, list_of_days, start_time, end_time, timezone, placement_id

This function is used to update the daypart targeting of a placement. Pass in the days you want the placement to run as a list of days as string. Pass in the start time and end time of the placement as hour, minute, and AM or PM. Next pass in the time zone , refer to the official FreeWheel documentation for the accept values for time zone.

Example:

import FreeWheel4py as fw

headers = fw.FW_Auth(username, password, 'xml')

placement_id = '12345'

list_of_days = ['monday', 'tuesday','friday']

start_time = '04:00AM'

end_time = '05:00PM'

timezone = '(GMT-05:00) America - New York'

fw.update_daypart_targeting(headers,list_of_days,start_time,end_time,timezone,placement_id)

---

function: update_fcap

parameters: headers, number_of_fcaps, imps, time, placement_id

this function updates the frequency cap of a placement and is under the delivery node of a placement. When using the function, this sets the delivery node of a placement equal to the below.

dict_obj['delivery'] = {'priority': 'GUARANTEED',

'pacing': 'SMOOTH_AS',

'override_repeat_mode': 'NONE',

'frequency_cap': ***parameters you define***

'ignore_brand_frequency_cap': 'false',

'excess_inventory': 'NETWORK_DEFAULT',

'dynamic_ad_insertion': 'DYNAMIC_ENABLED',

'excess_inventory_precondition': '0',

'level_to_optimize_for_profit': 'NONE'}

If you want me to expand on this function and make it update_delivery, please reach out and I will make it!

Also note that the frequency cap type can only be set to impressions and not package, let me know if you want that option as well.

Pass in number of fcaps as a list of how many frequency caps you want the placement to have. Next set imps equal to number of impressions for each imp cap as a list. Next set time equal to the number of minutes as a string the fcap should be set for.

Example:

If you wanted a placement to have an fcap of 4 impressions per minute and 20 impressions every 6 hours. This is what you would make the function look like.

```
import FreeWheel4py as fw

headers = fw.FW_Auth(username, password, 'xml')

placement_id = '12345'

number_of_fcaps = 2

imps = ['4','20']

time = ['60','360']

fw.update_fcap(headers,number_of_fcaps,imps,time,placement_id)
```

function: get_placement

parameters: headers, fwid, node

this gets a specific node of a placement. Set headers equal to FW_Auth and fwid as the placement you want to view. Refer to the FreeWheel official documentation if you wish to view a specific node but if you wish to view everything set node equal to 'all'. This function is a get requests and return a dictionary object of a placement.

Example:

```
import FreeWheel4py as fw

headers = fw.FW_Auth(username, password, 'xml')

fwid = '12345'

node = 'all'

placement = get_placement(headers,fwid,node)
```

the placement variable is everything you need to know about a placement. If you define node as all, and you want to view 'audience_targeting' you would say:

```
print(placement['placement']['audience_targeting'])
```

---

function: run_forecast

parameters: headers, placement_id

this function runs a forecast on a specific placement. All you need to do is define headers and the placement id. This function uses a post a request, and return a dictionary with the keys being placement_id and job_id and the value of the keys are the placement_id you defined and the job_id of the forecast. I have not created a get a forecast function, but I will be doing in the future.

Example:

import FreeWheel4py as fw

headers = fw.FW_Auth(username, password, 'json')

placement_id = '12345'

params = fw.get_placement(headers,placement_id)

---

function: get_nightly_forecast

parameters: headers, placement_id

this function returns a nightly forecast of a placement and returns a dictionary object where are the results of the placements nighly forecast.

Example:

import FreeWheel4py as fw

headers = fw.FW_Auth(username, password, 'json')

placement_id = '12345'

fw.get_nightly_forecast(headers,placement_id)

---

function: get_all_ids_and_names

parameters: headers, item, condition, per_page

this function gets all items (audience items, site group, video group, series, video) and its respective id and stores it as a tuple object where the 0^{th} index is the name and 1^{st} index is the Id. Set item equal to audience_items, site_group, video_group, series or video. Use the condition parameter if you want the item you need to have a specific string in it, note you can only have one condition. For example if you want all audiences that have 'demographic' in their name you would set condition equal to 'demographic'. If there is no condition, set condition equal to int(0). The per_page parameter shows you how many items you want per request, use a string for this parameter. For the purpose of efficient API usage, use 100 for audience_items and site_group and use 500 for everything else.

Example:

import FreeWheel4py as fw

headers = fw.FW_Auth(username, password, ‘json’)

item = ‘audience_items’

condition = 0

per_page = ‘100’

fw.get_all_ids_and_names(headers,item,condition,per_page)

---

function: get_last_page_for_FWitems

parameters: headers, item, per_page

this function returns the last page of a get request for the specific items. This function grabs the most recent updates for the specific item. This returns the same tuple as the get_all_items_and_ids, so please refer to that function’s documentation.

example:

import FreeWheel4py as fw

headers = fw.FW_Auth(username, password, ‘json’)

item = ‘audience_items’

per_page = ‘100’

fw.get_last_page_for_FWitems(headers,item,per_page)

---

function: get_all_ioids_in_a_campaign

parameters: headers, campaign_id

this function returns all the io ids in a campaign and stores the values in a tuple where the 0^{th} index is the name of the campaign and the 1^{st} index is the id of the io.

Example:

import FreeWheel4py as fw

headers = fw.FW_Auth(username, password, ‘xml’)

campaign_id = ‘12345’

fw.get_all_ioids_in_a_campaign(headers,campaign_id)

---

function: get_placement_ids_from_io

parameters: headers, io_id

this function returns all placements in an insertion order and stores the value in a tuple where the 0th index is the name of the placement and 1st index is the id of the placement

Example:

```
import FreeWheel4py as fw

headers = fw.FW_Auth(username, password, 'xml')

io_id = '12345'

fw.get_placement_ids_from_io(headers,io_id)
```