atomx-api-python Documentation

Release 1.7

Spot Media Solutions Sdn. Bhd.

Oct 26, 2017

Contents

Contents:

CHAPTER 1

Front Matter

atomx is a python interface for the atomx rest api.

With it you can easily query, create or update resources on atomx.

Project Info

The python *atomx* package is hosted on github.

Releases and project status are available on Pypi.

The most recent published version of this documentation is at http://atomx-api-python.readthedocs.org/en/latest/index.html.

See the atomx wiki for more general information about the *atomx api*.

Installation

To install the python atomx api, simply:

```
$ pip install atomx
```

or if you want to use ipython notebook and reporting functionality:

```
$ pip install atomx[report]
```

CHAPTER 2

Usage

Creating a session

To work with the api you need to get a user from atomx and create a *atomx.Atomx* session using your email and password:

```
from atomx import Atomx

# create atomx session
atomx = Atomx('user@example.com', 'password')
```

optionally you can specify a different api endpoint for testing:

```
# create atomx session with sandbox endpoint
atomx = Atomx('user@example.com', 'password',
              api_endpoint='https://sandbox.api.atomx.com/v2')
```

Fetching resources

Use *atomx.Atomx.get()* to fetch any resource from the api. The first parameter is the resource you want to query and can be any string that the atomx api accepts. All additional keyword arguments are used to build the query string.

If the returned resource (or list of resources) is any of the *atomx.models* then that model instance will be returned so you can easily work with it.

For example to get 10 creatives and then print some information about it:

```
# get 10 creatives
creatives = atomx.get('Creatives', limit=10)
# the result is a list of `atomx.models.Creative` models
# that you can easily inspect, manipulate and update
for creative in creatives:
```

```
    print('Creative ID: {c.id}, state: {c.state}, '
          'name: {c.name}, title: {c.title}'.format(c=creative))
```

Or query all profiles of advertiser 42 ordered by last updated:

```
profiles = atomx.get('advertiser/42/profiles', order_by='updated_at.desc')
```

All non-keyword arguments that you pass *atomx.Atomx.get()* will get used to compute the `resource`. This makes it easier if you have the `id` (and/or attribute) stored in a variable.

E.g.

```
advertiser_id = 42
attribute = 'profiles'
profiles = atomx.get('advertiser', advertiser_id, attribute)
# is equivalent to atomx.get('advertiser/42/profiles')
```

You can also use a class or instance of *atomx.models* in a get requests. E.g. all of those api calls request the same resource:

```
from atomx.models import Advertiser

atomx.get('advertiser/42')  # all in 1 string
atomx.get('advertiser', 42)  # model, id split up
atomx.get(Advertiser, 42)  # using :class:`atomx.models.Advertiser`
atomx.get(Advertiser(42))  # using instance of :class:`atomx.models.Advertiser`
```

Or get all domains where the hostname contains *atom*:

```
domains = atomx.get('domains', name='*atom*')
```

Attributes that are not loaded in the model will be lazy loaded once you try to access them. E.g. if you want to access the *quickstats* for the creative we fetched earlier you don't have to to anything special, just access the *quickstats* attribute:

```
creative = creatives[0]
print(creative.quickstats)
```

Or to get the advertiser for a profile, just:

```
advertiser = profiles[0].advertiser
```

You can get a list of all changes with *atomx.models.AtomxModel.history()*.

```
advertiser.history()
```

Updating models

To change a *atomx.models* model you just change any attribute you want and call *atomx.models.AtomxModel.save()*.

E.g.

```
# update title for the first creative in list
creative = creatives[0]
```

```
creative.title = 'shiny new title'
creative.save()

# update profile click frequency
profiles[0].click_frequency_cap_per = 86400
profiles[0].save()
```

Creating models

To add a new entry in *atomx* just instantiate any *atomx.models* model with all attributes you want your newly created model to have and either call *atomx.models.AtomxModel.create()* with your *atomx.Atomx* session as parameter or use *atomx.Atomx.save()*.

E.g. create a new profile entry:

```
# create a new profile
from atomx.models import Profile
profile = Profile(advertiser_id=23, name='test profile')
# Note that you have to pass it a valid `Atomx` session for create
# or use `atomx.create(profile)`
profile.create(atomx)
```

Search

Use *atomx.Atomx.search()* to search fast for anything.

atomx.Atomx.search() returns a *dict* with all found results for: 'Advertisers', 'Campaigns', 'Creatives', 'Placements', 'Publishers', 'Sites'.

The resulting *models* have only *id* and *name* loaded since that's what's returned from the api */search* call, but attributes will be lazy loaded once you try to accessed them. Or you can just fetch everything with one api call with *AtomxModel.reload()*.

Example:

```
search_result = atomx.search('atomx')

campaign = search_result['campaigns'][0]
assert isinstance(campaign, models.Campaign)

# campaign has only `id` and `name` loaded but you
# can still access (lazy load) all attributes
print(campaign.budget)
print(campaign.profile)

# or reload all attributes with one api call
campaign.reload()
```

Reports

See *atomx.Atomx.report()* for a description of available parameters to create a report.

```
from datetime import datetime, timedelta

now = datetime.utcnow()
last_week = now - timedelta(days=7)

# get a report for a specific publisher
report = atomx.report(scope='publisher', groups=['hour'], metrics=['impressions',
↪'clicks'], where=[['publisher_id', '==', 42]], from_=last_week, to=now, timezone=
↪'America/Los_Angeles')

report.length  # get the number of rows returned
report.totals  # get the total values

# if pandas is installed you can get the pandas dataframe with `report.pandas`
# you can also get the report csv in `report.content` without pandas
df = report.pandas  # A datetime index is automatically set when group by a hour/day/
↪month.
# calculate mean, median, std per hour
means = df.resample('H').apply(['mean', 'median', 'std'])
# and plot impression and clicks per day
means['impressions'].plot()
means['clicks'].plot()
```

For more general information about atomx reporting visit the reporting atomx knowledge base entry.

CHAPTER 3

API

Session

class atomx.**Atomx**(*email*, *password*, *totp=None*, *api_endpoint='https://api.atomx.com/v3'*, *save_response=True*, *expiration=None*)

Interface for the api on api.atomx.com.

To learn more about the api visit the atomx wiki

Parameters

- **email** (*str*) – email address of your atomx user
- **password** (*str*) – password of your atomx user
- **totp** (*str*) – 6 digit auth token if the account has 2-factor authentication enabled.
- **api_endpoint** (*str*) – url for connections to the api (defaults to *https://api.atomx.com/{API_VERSION}*)
- **save_response** (*bool*) – If *True* save the last api response meta info (without the resource payload) in `Atomx.last_response`. (default: *True*)

Returns *Atomx* session to interact with the api

create(*model*)

Alias for *models.AtomxModel.create()* with *session* argument.

delete(*resource*, **args*, ***kwargs*)

Send HTTP DELETE to `resource`.

Parameters

- **resource** – Name of the resource to *DELETE*.
- **args** – All non-keyword arguments will be used to compute the final `resource`.
- **kwargs** – Optional keyword arguments will be passed as query string to the delete request.

Returns message or resource returned by the api.

get(*resource*, **args*, ***kwargs*)

Returns a list of models from *models* if you query for multiple models or a single instance of a model from *models* if you query for a specific *id*

Parameters

- **resource** (*str*) – Specify the resource to get from the atomx api.

  Examples:

  Query all advertisers:

```
>>> atomx = Atomx('apiuser@example.com', 'password')
>>> advertisers = atomx.get('advertisers')
>>> assert isinstance(advertisers, list)
>>> assert isinstance(advertisers[0], atomx.models.Advertiser)
```

  Get publisher with id 23:

```
>>> publisher = atomx.get('publisher/23')
>>>> # or get the same publisher using the id as parameter
>>> publisher = atomx.get('publisher', 23)
>>>> # or use an atomx model
>>> publisher = atomx.get(atomx.models.Publisher(23))
>>> assert publisher.id == 23
>>> assert isinstance(publisher, atomx.models.Publisher)
```

  Get all profiles for advertiser 42:

```
>>> profiles = atomx.get('advertiser/42/profiles')
>>> assert isinstance(profiles, list)
>>> assert isinstance(profiles[0], atomx.models.Profile)
>>> assert profiles[0].advertiser.id == 42
```

- **args** – All non-keyword arguments will get used to compute the `resource`. This makes it easier if you want to work with a variable resource path.

```
advertiser_id = 42
attribute = 'profiles'
profiles = atomx.get('advertiser', advertiser_id, attribute)
# is equivalent to atomx.get('advertiser/42/profiles')
```

- **kwargs** – Any argument is passed as URL parameter to the respective api endpoint. See API URL Parameters in the wiki.

  Example: Get the first 20 domains that contain `atom`:

```
>>> atom_domains = atomx.get('domains', hostname='*atom*',␣
↪limit=20)
>>> assert len(atom_domains) == 20
>>> assert 'atom' in atom_domains[1].hostname
```

Returns a class from *models* or a list of models depending on param *resource*

login(*email*, *password*, *totp=None*, *expiration=None*)

Gets new authentication token for user `email`.

This method is automatically called in `__init__()` so you rarely have to call this method directly.

Parameters

- **email** (*str*) – Email to use for login.
- **password** (*str*) – Password to use for login.
- **totp** (*str*) – 6 digit auth token if the account has 2-factor authentication enabled.
- **expiration** (*int*) – Number of seconds that the auth token should be valid. (optional)

Returns None

Raises *exceptions.InvalidCredentials* if `email`/`password` is wrong

logout()
Removes authentication token from session.

post(*resource*, *json*, ***kwargs*)
Send HTTP POST to `resource` with `json` content.

Used by *models.AtomxModel.create()*.

Parameters

- **resource** – Name of the resource to *POST* to.
- **json** – Content of the *POST* request.
- **kwargs** – URL Parameters of the request.

Returns `dict` with the newly created resource.

put(*resource*, *id*, *json*, ***kwargs*)
Send HTTP PUT to `resource`/`id` with `json` content.

Used by *models.AtomxModel.save()*.

Parameters

- **resource** – Name of the resource to *PUT* to.
- **id** – Id of the resource you want to modify
- **json** – Content of the *PUT* request.
- **kwargs** – URL Parameters of the request.

Returns `dict` with the modified resource.

remove(*model*)
Alias for *models.AtomxModel.delete()* with *session* argument.

report(*scope=None*, *groups=None*, *metrics=None*, *where=None*, *from_=None*, *to=None*, *daterange=None*, *timezone='UTC'*, *emails=None*, *when=None*, *interval=None*, *name=None*, *sort=None*, *limit=None*, *offset=None*, *save=True*, *editable=False*)
Create a report.

See the reporting atomx wiki for details about parameters and available groups, metrics.

Parameters

- **scope** (*str*) – Specifies the report type. Should be one of: 'advertiser', 'publisher', 'inventory', 'dsp', 'network_managed', 'network_buy', 'network_sell'. If undefined it tries to determine the *scope* automatically based on the access rights of the api user.
- **groups** (*list*) – columns to group by.
- **metrics** (*list*) – columns to sum on.

- **where** (*list*) – is a list of expression lists. An expression list is in the form of `[column, op, value]`:
  - `column` can be any of the `groups` or `metrics` parameter columns.
  - `op` can be any of ==, !=, <, >, `in` or `not in` as a string.
  - `value` is either a number or in case of `in` and `not in` a list of numbers.
- **from** (*datetime.datetime*) – `datetime.datetime` where the report should start (inclusive). (Defaults to last week)
- **to** (*datetime.datetime*) – `datetime.datetime` where the report should end (exclusive). (Defaults to *datetime.now()* if undefined)
- **daterange** (*str*) – Use **:param:'daterange'** to automatically set the reports

from and *to* parameters relativ to the current date. Both **:param:'from_'** and **:param:'to'** have to be `None` for it. Dateranges are: today, yesterday, last7days, last14days, last30days, monthtodate,

lastmonth, yeartodate, lifetime. (Defaults to `None`)

Parameters

- **timezone** (*str*) – Timezone used for all times. (Defaults to *UTC*) For a supported list see https://wiki.atomx.com/timezones
- **emails** (*str or list*) – One or multiple email addresses that should get notified once the report is finished and ready to download.
- **when** (*str*) – When should the scheduled report run. (daily, monthly, monday-sunday)
- **interval** (*str*) – Time period included in the scheduled report ('N days' or 'N month')
- **name** (*str*) – Optional name for the report.
- **or list sort** (*str*) – List of columns to sort by.
- **limit** (*int*) – Number of rows to return
- **offset** (*int*) – Number of rows to skip.
- **save** (*bool*) – Should the report appear in the users report history (defaults to *True*).
- **editable** (*bool*) – Should other users be able to change the date range of this report.

Returns A *atomx.models.Report* model

save(*model*)

Alias for *models.AtomxModel.save()* with *session* argument.

search(*query*, *index=None*)

Search for `query`.

Returns a *dict* with all found results for: 'Advertisers', 'Campaigns', 'Creatives', 'Placements', 'Publishers', 'Sites'.

The resulting *models* have only *id* and *name* loaded since that's what's returned from the api */search* call, but attributes will be lazy loaded once you try to accessed them. Or you can just fetch everything with one api call with *AtomxModel.reload()*.

Example:

```
>>> atomx = Atomx('apiuser@example.com', 'password')
>>> search_result = atomx.search('atomx')
>>> assert 'campaigns' in search_result
>>> campaign = search_result['campaigns'][0]
>>> assert isinstance(campaign, models.Campaign)
>>> # campaign has only `id` and `name` loaded but you
>>> # can still access (lazy load) all attributes
>>> assert isinstance(campaign.budget, float)
>>> # or reload all attributes with one api call
>>> campaign.reload()
```

Parameters

- **query** (*str*) – keyword to search for.
- **index** (*list*) – `str` or `list` of the indexes you want to get returned. E.g. `index=['campaigns', 'domains']`.

Returns dict with list of *models* as values

Models

class `atomx.models.`**AtomxModel**(*id=None*, *session=None*, ***attributes*)

A generic atomx model that the other models from *atomx.models* inherit from.

Parameters

- **id** (*int*) – Optional model ID. Can also be passed in via *attributes* as *id*.
- **session** (`atomx.Atomx`) – The *atomx.Atomx* session to use for the api requests.
- **attributes** – model attributes

create(*session=None*)

POST the model to the api and populates attributes with api response.

Parameters **session** – The *atomx.Atomx* session to use for the api call. (Optional if you specified a *session* at initialization)

Returns `self`

Return type *AtomxModel*

delete(*session=None*)

DELETE the model in the api. A *deleted* attribute is to `True` on `self` so you can check if a *AtomxModel* is deleted or not.

> **Warning:** Calling this method will permanently remove this model from the API.

Parameters **session** – The *atomx.Atomx* session to use for the api call. (Optional if you specified a *session* at initialization)

Returns A `dict` of all models that the API removed. Keys are the model names and values are a list of IDs.

Return type `dict`

history(*session=None*, *offset=0*, *limit=100*, *sort='date.asc'*)
Show the changelog of the model.

Parameters

- **session** – The *atomx.Atomx* session to use for the api call. (Optional if you specified a *session* at initialization)
- **offset** (*int*) – Skip first `offset` history entries. (default: 0)
- **limit** (*int*) – Only return `limit` history entries. (default: 100)
- **sort** (*str*) – Sort by *date.asc* or *date.desc*. (default: 'date.asc')

Returns *list* of *dict's with 'date*, *user* and the attributes that changed (*history*).

Return type list

json
Returns the model attributes as `dict`.

reload(*session=None*, ***kwargs*)
Reload the model from the api and update attributes with the response.

This is useful if you have not all attributes loaded like when you made an api request with the *attributes* parameter or you used *atomx.Atomx.search()*.

Parameters **session** – The *atomx.Atomx* session to use for the api call. (Optional if you specified a *session* at initialization)

Returns `self`

Return type *AtomxModel*

save(*session=None*)
PUT the model to the api and update attributes with api response.

Parameters **session** – The *atomx.Atomx* session to use for the api call. (Optional if you specified a *session* at initialization)

Returns `self`

Return type *AtomxModel*

update(*session=None*)
Alias for *AtomxModel.save()*.

class atomx.models.**AccountManager**(*id=None*, *session=None*, ***attributes*)
AtomxModel for AccountManager

class atomx.models.**Advertiser**(*id=None*, *session=None*, ***attributes*)
AtomxModel for Advertiser

class atomx.models.**App**(*id=None*, *session=None*, ***attributes*)
AtomxModel for App

class atomx.models.**Appstore**(*id=None*, *session=None*, ***attributes*)
AtomxModel for Appstore

class atomx.models.**Bidder**(*id=None*, *session=None*, ***attributes*)
AtomxModel for Bidder

class atomx.models.**Browser**(*id=None*, *session=None*, ***attributes*)
AtomxModel for Browser

class atomx.models.**CampaignDebugReason**(*id=None*, *session=None*, ***attributes*)
AtomxModel for CampaignDebugReason

class atomx.models.**Campaign**(*id=None*, *session=None*, ***attributes*)
AtomxModel for Campaign

class atomx.models.**Category**(*id=None*, *session=None*, ***attributes*)
AtomxModel for Category

class atomx.models.**ConnectionType**(*id=None*, *session=None*, ***attributes*)
AtomxModel for ConnectionType

class atomx.models.**City**(*id=None*, *session=None*, ***attributes*)
AtomxModel for City

class atomx.models.**ConversionPixel**(*id=None*, *session=None*, ***attributes*)
AtomxModel for ConversionPixel

class atomx.models.**Country**(*id=None*, *session=None*, ***attributes*)
AtomxModel for Country

class atomx.models.**Creative**(*id=None*, *session=None*, ***attributes*)
AtomxModel for Creative

class atomx.models.**CreativeAttribute**(*id=None*, *session=None*, ***attributes*)
AtomxModel for CreativeAttribute

class atomx.models.**Datacenter**(*id=None*, *session=None*, ***attributes*)
AtomxModel for Datacenter

class atomx.models.**DeviceType**(*id=None*, *session=None*, ***attributes*)
AtomxModel for DeviceType

class atomx.models.**Domain**(*id=None*, *session=None*, ***attributes*)
AtomxModel for Domain

class atomx.models.**Dma**(*id=None*, *session=None*, ***attributes*)
AtomxModel for Dma

class atomx.models.**Dsp**(*id=None*, *session=None*, ***attributes*)
AtomxModel for Dsp

class atomx.models.**Fallback**(*id=None*, *session=None*, ***attributes*)
AtomxModel for Fallback

class atomx.models.**Isp**(*id=None*, *session=None*, ***attributes*)
AtomxModel for Isp

class atomx.models.**Languages**(*id=None*, *session=None*, ***attributes*)
AtomxModel for Languages

class atomx.models.**Network**(*id=None*, *session=None*, ***attributes*)
AtomxModel for Network

class atomx.models.**OperatingSystem**(*id=None*, *session=None*, ***attributes*)
AtomxModel for OperatingSystem

class atomx.models.**Placement**(*id=None*, *session=None*, ***attributes*)
AtomxModel for Placement

class atomx.models.**PlacementType**(*id=None*, *session=None*, ***attributes*)
AtomxModel for PlacementType

class atomx.models.**PriceModel**(*id=None*, *session=None*, ***attributes*)
AtomxModel for PriceModel

class atomx.models.**Profile**(*id=None*, *session=None*, ***attributes*)
AtomxModel for Profile

class atomx.models.**Publisher**(*id=None*, *session=None*, ***attributes*)
AtomxModel for Publisher

class atomx.models.**Reason**(*id=None*, *session=None*, ***attributes*)
AtomxModel for Reason

class atomx.models.**Report**(*id*, *query=None*, *name=None*, *emails=None*, *length=None*, *totals=None*, *columns=None*, *created_at=None*, *data=None*, *user_id=None*, *session=None*, *is_scheduled_report=False*, *to=None*, *from_=None*, ***kwargs*)
Represents a *report* you get back from atomx.Atomx.report().

delete(*session=None*)
Delete report

pandas
Returns the content of the *report* as a pandas data frame.

save(*session=None*)
Update report *name* and *emails*

class atomx.models.**Segment**(*id=None*, *session=None*, ***attributes*)
AtomxModel for Segment

class atomx.models.**SellerProfile**(*id=None*, *session=None*, ***attributes*)
AtomxModel for SellerProfile

class atomx.models.**Site**(*id=None*, *session=None*, ***attributes*)
AtomxModel for Site

class atomx.models.**Size**(*id=None*, *session=None*, ***attributes*)
AtomxModel for Size

class atomx.models.**Ssp**(*id=None*, *session=None*, ***attributes*)
AtomxModel for Ssp

class atomx.models.**SspResultType**(*id=None*, *session=None*, ***attributes*)
AtomxModel for SspResultType

class atomx.models.**SspSuspicious**(*id=None*, *session=None*, ***attributes*)
AtomxModel for SspSuspicious

class atomx.models.**Timezone**(*id=None*, *session=None*, ***attributes*)
AtomxModel for Timezone

class atomx.models.**User**(*id=None*, *session=None*, ***attributes*)
AtomxModel for User

class atomx.models.**Visibility**(*id=None*, *session=None*, ***attributes*)
AtomxModel for Visibility

class atomx.models.**Zipcode**(*id=None*, *session=None*, ***attributes*)
AtomxModel for Zipcode

class atomx.models.**Report**(*id*, *query=None*, *name=None*, *emails=None*, *length=None*, *totals=None*, *columns=None*, *created_at=None*, *data=None*, *user_id=None*, *session=None*, *is_scheduled_report=False*, *to=None*, *from_=None*, ***kwargs*)

Represents a *report* you get back from *atomx.Atomx.report()*.

delete(*session=None*)

Delete report

pandas

Returns the content of the *report* as a pandas data frame.

save(*session=None*)

Update report *name* and *emails*

Exceptions

exception atomx.exceptions.**APIError**

Raised when the atomx api returns an error that is not caught otherwise.

exception atomx.exceptions.**InvalidCredentials**

Raised when trying to login with the wrong e-mail or password.

exception atomx.exceptions.**MissingArgumentError**

Raised when argument is missing.

exception atomx.exceptions.**ModelNotFoundError**

Raised when trying to (re-)load a model that is not in the api.

exception atomx.exceptions.**NoPandasInstalledError**

Raised when trying to access `report.pandas` without `pandas` installed.

exception atomx.exceptions.**NoSessionError**

Raised when a model from *models* wants to `create`/`update` but it doesn't have an *atomx.Atomx* session yet.

CHAPTER 4

atomx-api-python changelog

1.7

- Add `editable` parameter to *atomx.Atomx.report()*.
- Add `dsp` reporting scope.
- Add `daterange` parameter to *atomx.Atomx.report()*.
- When accessing a model attribute, attributes that are atomx models will now automatically returned as *atomx.models.AtomxModel* instead of leaving them as *dict*.
- *atomx.models.AtomxModel.reload()* takes kwargs parameter
- *atomx.Atomx.delete()* can take a *atomx.models* instance as argument
- Add new models:
  - *atomx.models.Dsp*
  - *atomx.models.Ssp*
  - *atomx.models.SspSuspicious*
  - *atomx.models.SspResultType*

1.6

- *atomx.Atomx.login()* takes `totp` parameter for users that have 2-factor auth enabled.
- *atomx.Atomx.search()* takes `index` parameter to only search specific models.
- Add support to remove models from the API. See *atomx.models.AtomxModel.delete()*.
- Add *atomx.Atomx.remove()*.
- Add `save` parameter to *atomx.Atomx.report()*.

- Add new models:
  - *atomx.models.App*
  - *atomx.models.Appstore*
  - *atomx.models.City*
  - *atomx.models.Dma*
  - *atomx.models.PriceModel*
  - *atomx.models.Timezone*
  - *atomx.models.Zipcode*

1.5

- *atomx.Atomx.report()* takes `name` parameter to name reports
- remove `network` scope in *atomx.Atomx.report()* and add `network_managed`, `network_buy`, `network_sell`
- Add new models:
  - *atomx.models.AccountManager* (alias for *User*)
  - *atomx.models.CampaignDebugReason*
  - *atomx.models.CreativeAttribute*
  - *atomx.models.PlacementType*
  - *atomx.models.Visibility*
- Add *atomx.models.Report.save()* to edit `name` and `emails` of a *atomx.models.Report*
- *atomx.Atomx.get()* also accepts a model class or instance as resource argument. E.g.: `atomx_api.get(atomx.models.Advertiser)` or `atomx_api.get(atomx.models.Advertiser(42))`
- Add `pickle` support for *atomx.models*.
- Save HTTP headers in `atomx.Atomx.last_response`.
- Add history support. *atomx.models.AtomxModel.history()*.
- Use api version 3:

```
* Fast reporting. No more polling if reports are ready.
* Use authentication token in HTTP header instead of cookies.
* Model and attribute changes
```

1.4

- Change default API version to *v2*
- If *atomx.Atomx.post()* returns a list, auto-convert list of objects to a list of *atomx.models* models. (Useful for *POST* to the */domains* endpoint)
- Add `atomx.models.ScheduledReport`

- *atomx.Atomx.report()* accepts `when` and `interval` to create a `atomx.models.ScheduledReport`

1.3

- Add *atomx.Atomx.delete()* to send a `HTTP DELETE` request to the api
- *atomx.Atomx.get()* and *atomx.Atomx.delete()* accept non-keyword arguments that are used to compute the final resource path
- Add *emails* parameter to *atomx.Atomx.report()*
- Model attributes that are dates get automatically converted to a python `datetime`
- When saving a model, dates, sets and decimals get automatically converted to there json counterpart
- Add *save_response* parameter to *atomx.Atomx* to save the response meta data of the last api call

1.2

- You can now remove model attributes with *del*
- Add `atomx.models.Report.csv()` property that returns the report content as a list
- Save logged in user as *user* property to *atomx.Atomx*
- Add network reports
- Try to determine report scope from user access rights if no scope was specified

1.1

- Fix: setup.py not working under some environments (*open* used wrong codec)
- Add SellerProfile model
- Add *offset* parameter to `atomx.models.Report.get()`

1.0

- First release

CHAPTER 5

Indices and tables

- genindex
- modindex
- search

Python Module Index

a

Index

R

S

T

U

V

Z