G API Python Client Documentation

Release 2.26.2

G Adventures

Apr 21, 2020

Contents

Contents:

CHAPTER 1

G API Python Client

A client for the G Adventures REST API (https://developers.gadventures.com)

- GitHub Repository: https://github.com/gadventures/gapipy/
- Documentation: http://gapipy.readthedocs.org.
- Free software: MIT license

1.1 Quick Start

```
>>> from gapipy import Client
>>> api = Client(application_key='MY_SECRET_KEY')

>>> # Get a resource by id
>>> tour_dossier = api.tour_dossiers.get(24309)
>>> tour_dossier.product_line
u'AHEH'
>>> tour_dossier.departures.count()
134
>>> tour_dossier.name
u'Essential India'
>>> itinerary = tour_dossier.structured_itineraries[0]
>>> {day.day: day.summary for day in itinerary.days[:3]}
{1: u'Arrive at any time. Arrival transfer included through the G Adventures-
 ↪supported Women on Wheels project.',
2: u'Take a morning walk through the city with a young adult from the G Adventures-
 ↪supported New Delhi Streetkids Project. Later, visit Old Delhi, explore the spice␣
 ↪markets, and visit Jama Masjid and Connaught Place.',
3: u"Arrive in Jaipur and explore this gorgeous 'pink city'."}
```

(continues on next page)

(continued from previous page)

```
>>> # Create a new resource
>>> booking = api.bookings.create({'currency': 'CAD', 'external_id': 'abc'})

>>> # Modify an existing resource
>>> booking.external_id = 'def'
>>> booking.save()
```

Since **'2.25.0 (2020-01-02)'_**

```
>>> # since 2.25.0 reference stubs that fail to fetch will return a

>>> # subclass of requests.HTTPError (See: https://github.com/gadventures/gapipy/pull/
↪119)
>>> # This can also be done on Query.get by passing a Falsy value for the
>>> # httperrors_mapped_to_none kwarg.
>>>
>>> dep = api.departures.get('404_404', httperrors_mapped_to_none=None)
... # omitted stacktrace
HTTPError: 404 Client Error: {"http_status_code":404,"message":"Not found.","errors
↪":[],"time":"2020-01-02T19:46:07Z","error_id":"gapi_asdf1234"} for url: https://
↪rest.gadventures.com/departures/404_404

>>> dep = api.departures.get('404404')
>>> dep.start_address.country
<Country: BR (stub)>

>>> # lets have GAPI return a _404_ error here for the country stub `fetch`
>>> # when we attempt to retrieve the continent attribute

>>> dep.start_address.country.continent  # reference/stub forces a fetch

>>> # pre 2.25.0 behaviour
... # omitted stacktrace
AttributeError: 'Country' has no field 'continent' available

>>> # post 2.25.0 behaviour
... # omitted stacktrace
HTTPError: 404 Client Error: {"http_status_code":404,"message":"Not found.","errors
↪":[],"time":"2020-01-02T19:46:07Z","error_id":"gapi_qwer5678"} for url: https://
↪rest.gadventures.com/countries/BR
```

1.2 Resources

Resource objects are instantiated from python dictionaries created from JSON data. The fields are parsed and converted to python objects as specified in the resource class.

A nested resource will only be instantiated when its corresponding attribute is accessed in the parent resource. These resources may be returned as a `stub`, and upon access of an attribute not present, will internally call `.fetch()` on the resource to populate it.

A field pointing to the URL for a collection of a child resources will hold a `Query` object for that resource. As for nested resources, it will only be instantiated when it is first accessed.

1.3 Queries

A Query for a resource can be used to fetch resources of that type (either a single instance or an iterator over them, possibly filtered according to some conditions). Queries are roughly analogous to Django's QuerySets.

An API client instance has a query object for each available resource (accessible by an attribute named after the resource name)

1.3.1 Methods on Query objects

All queries support the `get`, `create` and `options` methods. The other methods are only supported for queries whose resources are listable.

`options()` Get the options for a single resource

`get(resource_id, [headers={}])` Get a single resource; optionally passing in a dictionary of header values.

`create(data)` Create an instance of the query resource using the given data.

`all([limit=n])` Generator over all resources in the current query. If `limit` is a positive integer `n`, then only the first `n` results will be returned.

- A `TypeError` will be raised if limit is not `None` or `int` type
- A `ValueError` will be raised if `limit <= 0`>

`filter(field1=value1, [field2=value2, ...])`

`filter(**{"nested.field": "value"})` Filter resources on the provided fields and values. Calls to `filter` can be chained. The method will return a clone of the `Query` object and must be stored in a separate variable in order to have access to **stacked** filters.

`count()` Return the number of resources in the current query (by reading the `count` field on the response returned by requesting the list of resources in the current query).

1.4 Caching

`gapipy` can be configured to use a cache to avoid having to send HTTP requests for resources it has already seen. Cache invalidation is not automatically handled: it is recommended to listen to G API webhooks to purge resources that are outdated.

By default, `gapipy` will use the cached data to instantiate a resource, but a fresh copy can be fetched from the API by passing `cached=False` to `Query.get`. This has the side-effect of recaching the resource with the latest data, which makes this a convenient way to refresh cached data.

Caching can be configured through the `cache_backend` and `cache_options` settings. `cached_backend` should be a string of the fully qualified path to a cache backend, i.e. a subclass of `gapipy.cache.BaseCache`. A handful of cache backends are available out of the box:

- **`gapipy.cache.SimpleCache`** A simple in-memory cache for single process environments and is not thread safe.
- **`gapipy.cache.RedisCache`** A key-value cache store using Redis as a backend.
- **`gapipy.cache.NullCache`** **(Default)** A cache that doesn't cache.

Since the cache backend is defined by a python module path, you are free to use a cache backend that is defined outside of this project.

1.5 Connection Pooling

We use the `requests` library, and you can take advantage of the provided connection pooling options by passing in a `'connection_pool_options'` dict to your client.

Values inside the `'connection_pool_options'` dict of interest are as follows:

- Set `enable` to `True` to enable pooling. Defaults to `False`.
- Use `number` to set the number of connection pools to cache. Defaults to 10.
- Use `maxsize` to set the max number of connections in each pool. Defaults to 10.
- Set `block` to `True` if the connection pool should block and wait for a connection to be released when it has reached `maxsize`. If `False` and the pool is already at `maxsize` a new connection will be created without blocking, but it will not be saved once it is used. Defaults to `False`.

See also:

- http://www.python-requests.org/en/latest/api/#requests.adapters.HTTPAdapter
- http://urllib3.readthedocs.io/en/latest/reference/index.html#module-urllib3.connectionpool

1.6 Dependencies

The only dependency needed to use the client is requests.

1.7 Testing

Running tests is pretty simple. We use *nose* as the test runner. You can install all requirements for testing with the following:

```
$ pip install -r requirements-testing.txt
```

Once installed, run unit tests with:

```
$ nosetests -A integration!=1
```

Otherwise, you'll want to include a GAPI Application Key so the integration tests can successfully hit the API:

```
$ export GAPI_APPLICATION_KEY=MY_SECRET_KEY; nosetests
```

In addition to running the test suite against your local Python interpreter, you can run tests using Tox. Tox allows the test suite to be run against multiple environments, or in this case, multiple versions of Python. Install and run the `tox` command from any place in the gapipy source tree. You'll want to export your G API application key as well:

```
$ export GAPI_APPLICATION_KEY=MY_SECRET_KEY
$ pip install tox
$ tox
```

Tox will attempt to run against all environments defined in the `tox.ini`. It is recommended to use a tool like pyenv to ensure you have multiple versions of Python available on your machine for Tox to use.

1.8 Fields

- `_model_fields` represent dictionary fields like so:

Note: `_model_fields = [('address', Address)]` and `Address` subclasses `BaseModel`

```
"address": {
  "street": "19 Charlotte St",
  "city": "Toronto",
  "state": {
    "id": "CA-ON",
    "href": "https://rest.gadventures.com/states/CA-ON",
    "name": "Ontario"
  },
  "country": {
    "id": "CA",
    "href": "https://rest.gadventures.com/countries/CA",
    "name": "Canada"
  },
  "postal_zip": "M5V 2H5"
}
```

- `_model_collection_fields` represent a list of dictionary fields like so:

Note: `_model_collection_fields = [('emails', AgencyEmail),]` and `AgencyEmail` subclasses `BaseModel`

```
"emails": [
  {
    "type": "ALLOCATIONS_RELEASE",
    "address": "g@gadventures.com"
  },
  {
    "type": "ALLOCATIONS_RELEASE",
    "address": "g2@gadventures.com"
  }
]
```

- `_resource_fields` refer to another `Resource`

CHAPTER 2

Contributing

0. Run `pip install -r requirements-dev.txt` to setup dev dependencies
1. Always make your changes in a branch and submit a PR
2. Once the PR has been accepted/merged into the `master` branch, follow these steps on your local box

```
$> cd /path/to/gapipy
$> git checkout master
$> git pull origin master
```

Then, modify the following files:

- `gapipy/__init__.py`
  - update the `__version__` variable
  - NOTES on incrementing the version:
    - `major.minor.patch`
    - update `major` when we switch to `python3` only support
    - update `minor` if there is some breaking change or adding a New resource
    - update `patch` when adding new fields, fixing bugs introduced by a minor release
    - See semver.org for more information.
- `HISTORY.rst`
  - update this file with the new `version` & `date`
  - Add some brief notes describing the changes

3. Check the generated long_description rST file is valid

```
$> python setup.py sdist
# this created `gapipy-a.b.c.tar.gz` in the `./dist` directory
# where a.b.c is the ``__version__`` value
```

(continues on next page)

(continued from previous page)

```
$> twine check dist/gapipy-a.b.c.tar.gz
# checks the long-form rST file is valid

# if there are any errors fix, and repeat

# example success check
$> twine check dist/gapipy-a.b.c.tar.gz
Checking dist/gapipy-a.b.c.tar.gz: PASSED
```

4. Push the new commit

   - Use `Release a.b.c (YYYY-MM-DD)` format for the commit title. Optionally add a description that matches the changes to `HISTORY.rst`

5. Create a release on github with the following description (This will be tagged to the `version bump` commit and not the PR commit)

```
# Version a.b.c

PR: #123

A brief description describing the changes
* bullet points
* make for easy reading
```

6. Back to your local box

```
# build `gapipy-a.b.c.tar.gz` in the `./dist` directory
# where a.b.c is the ``__version__`` value
$> python setup.py sdist

# check the long-form rST file is valid
$> twine check dist/gapipy-a.b.c.tar.gz

$> twine upload dist/gapipy-a.b.c.tar.gz
# this will upload & create the release pypi
```

Thanks for helping!

CHAPTER 3

History

3.1 2.26.2 (2020-04-20)

- Fix for *2.26.1 (2020-04-20)* and Issue #113. * See PR 125 * Remove the `_set_resource_collection_field` method in `TourDossier` * Introducing the `_Parent` namedtuple in PR 123 broke being able to Query-chain from Tour-Dossiers to departures * Buggy behaviour from `2.26.1` below:

```
>>> from gapipy import Client
>>> api = Client(application_key='MY_SECRET_KEY')

>>> api.tour_dossiers(24309).departures.count()
# AttributeError: 'tuple' object has no attribute 'uri'
```

3.2 2.26.1 (2020-04-20)

- Fix for *2.26.0 (2020-04-14)* and Issue #113. * Calls to `APIRequestor.list_raw` will use initialised its parameters, unless the URI provides its own. * See PR 123.
- Add the ability to define the `max_retries` values on the requestor. * New `env` value `GAPI_CLIENT_MAX_RETRIES`. * The default value will be `0`, and if provided will override the `retry` value on the `requests.Session`. * This change will also always initialize a `requests.Session` value on initialisation of the `gapipy.Client`. * See PR 124.
- Add `variation_id` field to the `Image` resource. * See Commit edc8d9b.
- Update the `ActivityDossier` and `AccommodationDossier` resources. * Remove the `is_prepaid` field. * Adds the `has_costs` field. * See Commit bd35531.

3.3 2.26.0 (2020-04-14)

Breaking Changes

- The `Query.filter` method will return a clone/copy of itself. This will preserve the state of `filters` on the original Query object.
- The `Query.all` method will **not** clear the filters after returning.
- The `Query.all` method will return a `TypeError` if a type other than an `int` is passed to the `limit` argument.
- The `Query.count` method will **not** clear the filters after returning.
- See PR 121

New behaviour with the `Query.filter` method:

```
>>> from gapipy import Client
>>> api = Client(application_key='MY_SECRET_KEY')

# create a filter on the departures
>>> query = api.departures.filter(**{"tour_dossier.id": "24309"})
>>> query.count()
494

# we preserve the filter status of the current query
>>> query.filter(**{"availability.status": "AVAILABLE"}).count()
80

>>> query.count()
494
```

- The `AgencyChain.agencies` attribute returns a list of `Agency` objects * See Commit f34afd52

3.4 2.25.1 (2020-01-02)

- Improve contribution instructions to check long_description rST file in dist
- Dev Requirement updates: * Add `readme_renderer==24.0` * Add `twine==1.15.0` for `twine check` command

3.5 2.25.0 (2020-01-02)

- Failing to fetch inlined Resource (from Stubs) will raise the underlying requests.HTTPError instead of AttributeError resulting from a `None`.
- Adds `httperrors_mapped_to_none` kwarg to `gapipy.query.Query.get` with default value `gapipy.query.HTTPERRORS_MAPPED_TO_NONE`
- Modifies `gapipy.resources.base.Resource.fetch` to pass `httperrors_mapped_to_none=None` to `Query.get`
- This ensures that any underlying `requests.HTTPError` from `Query.get` is bubbled up to the caller. It is most prevalent when reference Resource stubs fail to be retrieved from the G API. Prior to this change

`Resource.fetch` would return a `None` value resulting in an `AttributeError`. Now, if the stub fails to fetch due to an HTTPError, that will be raised instead

3.6 2.24.3 (2019-12-12)

- Exclude the `tests` package from the package distribution

3.7 2.24.2 (2019-12-12)

- Adds the `compute_request_signature` and `compute_webhook_validation_key` utility methods * See https://github.com/gadventures/gapipy/pull/122

3.8 2.24.1 (2019-12-12)

- Add `slug` field to `TourDossier` resource * See https://github.com/gadventures/gapipy/pull/120

3.9 2.24.0 (2019-11-05)

- Add missing/new fields to resources * Accommodation Dossier: categories, suggested_dossiers, visited_countries, and visited_cities * Activity Dossier: suggested_dossiers, visited_countries, and visited_cities * Departure: local_payments * Itinerary: publish_state * See https://github.com/gadventures/gapipy/pull/117
- Add `continent` and `place` references to the `Countries` resource * See https://github.com/gadventures/gapipy/pull/115
- Accept `additional_headers` optional kwarg on `create` * See https://github.com/gadventures/gapipy/pull/114

3.10 2.23.0 (2019-11-04)

- Remove deprecated `tour_dossiers.itineraries` field and related code

3.11 2.22.0 (2019-10-10)

- Add `booking_company` field to `Booking` resource

3.12 2.21.0 (2019-04-09)

- Add `ripple_score` to `Itinerary` resource

3.13 2.20.1 (2019-02-20)

- HISTORY.rst doc fixes

3.14 2.20.0 (2019-02-20)

- Add `Requirement` and `RequirementSet` resources
- Move `Checkin` resource to the `resources.booking` module
- The `Query` object will resolve to use the `href` value when returning the iterator to fetch `all` of some resource. This is needed because `bookings/123456/requirements` actually returns a list of `RequirementSet` resources
- see https://github.com/gadventures/gapipy/releases/tag/2.20.0 for more details

3.15 2.19.4 (2019-02-14)

- Add `get_category_name` helper method to `TourDossier` resource

3.16 2.19.3 (2019-02-12)

- Attempt to fix rST formatting of `README` and `HISTORY` on pypi

3.17 2.19.2 (2019-02-12)

- Become agnostic between redis 2.x.x && 3.x.x versions
  - the `setex` method argument order changes between the major versions

3.18 2.19.1 (2019-02-12)

- HotFix for `2.19.0` – adds `requirements.txt` file to the distribution `MANIFEST`

3.19 2.19.0 (2019-02-12)

- Add `booking_companies` field to `Itinerary` resource
- Pin our requirement/dependency versions
  - pin `future == 0.16.0`
  - pin `requests >= 2.18.4, < 3.0.0`
  - read `setup.py` requirements from `requirements.txt`

3.20 2.18.1 (2019-02-07)

- Add `customers` nested resource to `bookings`

3.21 2.18.0 (2018-12-14)

- Add `merchandise` resource
- Add `merchandise_services` resources

3.22 2.17.0 (2018-11-12)

- Add `membership_programs` field to the `Customer` resource

3.23 2.16.0 (2018-11-07)

- Completely remove the deprecated `add_ons` field from the Departure resource
- Add missing fields to various Dossier resources
  - Accommodation Dossier: `flags`, `is_prepaid`, `service_time`, `show_on_reservation_sheet`
  - Activity Dossier: `is_prepaid`, `service_time`, `show_on_reservation_sheet`
  - Country Dossier: `flags`
  - Place Dossier: `flags`
  - Transport Dossier: `flags`
- Add `valid_during_ranges` list field to the Itinerary resource. This field is a list field of the newly added `ValidDuringRange` model (described below)
- Add `ValidDuringRange` model. It consists of two date fields, `start_date`, and `end_date`. It also provides a number of convenience methods to determine if the date range provided is valid, or relative to some date.
  - `is_expired`: Is it expired relative to `datetime.date.today` (occurs in the past)
  - `is_valid_today`: Is it valid relative to `datetime.date.today`
  - `is_valid_during_range`: Is it valid for some give start/end date range
  - `is_valid_on_or_after_date`: Is it valid on or after some date
  - `is_valid_on_or_before_date`: Is it valid on or before some date
  - `is_valid_on_date`: Is it valid on some date
  - `is_valid_sometime`: Is it valid at all

3.24 2.15.0 (2018-10-10)

- Add `country` reference to `Nationality` resource
- Moved `resources/bookings/nationality.py` to `resources/geo/*`

3.25 2.14.6 (2018-08-01)

- Check for presence of `id` field directly in the Resource `__dict__` in order to prevent a chicken/egg situation when attempting to `save`. This is needed due to the change introduced in 2.14.4, where we explicitly raise an AttributeError when trying to access the `id` attribute.
- Added `service_code` field for Activty & Accommodation Dossier resources

3.26 2.14.5 (2018-08-01)

- deleted

3.27 2.14.4 (2018-07-13)

- Raise an AttributeError when trying to access *id* on Resource.__getattr__
- Don't send duplicate params when paginating through list results
- Implement first() method for Query

3.28 2.14.3 (2018-05-29)

- Expose Linked Bookings via the API

3.29 2.14.1 (2018-05-15)

- Add `booking_companies` field to Agency resource
- Remove `bookings` field from Agency resource
- Add `requirements` as_is field to Departure Service resource
- Add `policy_emergency_phone_number` field to Insurance Service resource

3.30 2.14.0 (2018-05-15)

- Remove deprecated `add_ons` field from `Departure` resource
- Add `costs` field to `Accommodation & Activity Dossier` resources

3.31 2.13.0 (2018-03-31)

- Add `meal_budgets` list field to `Country Dossier` resource
- Add `publish_state` field to `Dossier Features` resource

3.32 2.12.0 (2018-02-14)

- Add optional `headers` parameter to Query.get to allow HTTP-Headers to be passed. e.g. `client.<resource>.get(1234, headers={'A':'a'})` (PR/91)
- Add `preferred_display_name` field to Agency resource (#92)
- Add `booking_companies` array field to all Product-type Resources. (PR/93)
  - Accommodation
  - Activity
  - AgencyChain
  - Departure
  - SingleSupplement
  - TourDossier
  - Transport

3.33 2.11.4 (2018-01-29)

- Add `agency_chain` field to `Booking` resource
- Add `id` field as part of the `DossierDetail` model (PR/89)
- Add `agency_chains` field to the `Agency` resource (PR/90)
- see https://github.com/gadventures/gapipy/releases/tag/2.11.3 for more details

3.34 2.11.0 (2017-12-18)

- The Customer Address uses `Address` model, and is no longer a dict.
- Passing in `uuid=True` to `Client` kwargs enables `uuid` generation for every request.

3.35 2.10.0 (2017-12-01)

- Add the `amount_pending` field to the `Booking` resource
- The `PricePromotion` model extends from the `Promotion` resource (PR/85)
- Update the `Agent` class to use BaseModel classes for the `role` and `phone_numbers` fields.
- see https://github.com/gadventures/gapipy/releases/tag/2.10.0 for more details

3.36 2.9.3 (2017-11-23)

- Expose `requirement_set` for `departure_services` and `activity_services`.
- *NOTE*: We have skipped `2.9.2` due to pypi upload issues.

3.37 2.9.1 (2017-11-22)

- Adds the `options` method on the Resource Query object. A more detailed description of the issue can be found at: https://github.com/gadventures/gapipy/releases/tag/2.9.1
- *NOTE*: We have skipped `2.9.0` due to pypi upload issues

3.38 2.8.2 (2017-11-14)

- Adds fields `sale_start_datetime` and `sale_finish_datetime` to the Promotion resource. The fields mark the start/finish date-time values for when a Promotion is applicable. The values represented are in UTC.

3.39 2.8.1 (2017-10-25)

- Add new fields to the `Agency` and `AgencyChain` resources

3.40 2.8.0 (2017-10-23)

- This release adds a behaviour change to the `.all()` method on resource Query objects. Prior to this release, the base Resource Query object would retain any previously added `filter` values, and be used in subsequent calls. Now the underlying filters are reset after a `<resource>.all()` call is made.

  A more detailed description of the issue and fix can be found at:

  - https://github.com/gadventures/gapipy/issues/76
  - https://github.com/gadventures/gapipy/pull/77
- Adds missing fields to the Agency and Flight Service resources (PR/78)

3.41 2.7.6 (2017-10-04)

- Add `agency` field to `Booking` resource.

3.42 2.7.5 (2017-09-25)

- Add test fix for Accommodation. It is listable resource as of `2.7.4`
- Add regression test for departures.addon.product model * Ensure Addon's are instantiated to the correct underlying model. * Prior to this release, all Addon.product resources were instantiated as `Accommodation`.

3.43 2.7.4 (2017-09-20)

- Add `videos`, `images`, and `categories` to Activity, Transport, Place, and, Accommodation Dossier resources.
- Add `flags` to Itinerary resource
- Add list view of `Accommodations` resource

3.44 2.7.3 (2017-09-06)

- Add `type` field to `AgencyDocument` model
- Add `structured_itinerary` model collection field to `Departure` resource

3.45 2.7.2 (2017-08-18)

- Fix flight_status Reference value in FlightService resource

3.46 2.7.1 (2017-08-18)

- Fix: remove FlightStatus import reference for FlightService resource
- Add fields (fixes two broken Resource tests)
  - Add `href` field for `checkins` resource
  - Add `date_cancelled` field for `departures` resource
- Fix broken UpdateCreateResource tests

3.47 2.7.0 (2017-08-18)

- Remove `flight_statuses` and `flight_segments` resources.

3.48 2.6.2 (2017-08-11)

- Version bump

3.49 2.6.1 (2017-08-11)

- Adds a Deprecation warning when using the `tours` resource.

3.50 2.6.0 (2017-08-11)

- Fixed issue 65: only write data into the local cache after a fetch from the API, do not write data into the local cache when fetching from the local cache.

3.51 2.5.2 (2017-04-26)

- Added `future` dependency to setup.py

3.52 2.5.1 (2017-02-08)

- Fixed an issue in which modifying a nested dictionary caused gapipy to not identify a change in the data.
- Added `tox.ini` for testing across Python platforms.
- Capture `403` Status Codes as a `None` object.

3.53 2.5.0 (2017-01-20)

- Provided Python 3 functionality (still Python 2 compatible)
- Removed Python 2 only tests
- Installed `future` module for smooth Python 2 to Python 3 migration
- Remove `DictToModel` class and the associated tests
- Add `Dossier` Resource(s)
- Minor field updates to: `Customer`, `InsuranceService`, `DepartureService`, `Booking`, `FlightStatus`, `State`

3.54 2.4.9 (2016-11-22)

- Fixed a bug with internal `_get_uri` function.

3.55 2.4.8 (2016-11-11)

- Adjusted `Checkin` resource to meet updated spec.

3.56 2.4.7 (2016-10-25)

- Added `Checkin` resource.

3.57 2.4.6 (2016-10-19)

- Fix broken `Duration` init in `ActivityDossier` (likely broke due to changes that happened in 2.0.0)

3.58 2.4.5 (2016-10-13)

- Added `Image` resource definition and put it to use in `Itinerary` and, `PlaceDossier`

3.59 2.4.4 (2016-09-09)

- Added `date_last_modified` and `date_created` to `Promotion`.

3.60 2.4.3 (2016-09-06)

- Added `gender` to `Customer`.
- Added `places_of_interest` to `Place`.

3.61 2.4.2 (2016-07-08)

- Added `departure` reference to `DepartureComponent`

3.62 2.4.1 (2016-07-06)

- Removed use of `.iteritems` wherever present in favour of `.items`
- Added `features` representation to `ActivityDossier` and, `TransportDossier`

3.63 2.4.0 (2016-06-29)

- Added `CountryDossier` resource.

3.64 2.3.0 (2016-06-28)

- Added `DossierSegment` resource.
- Added `ServiceLevel` resource.

3.65 2.2.2 (2016-06-08)

- Added day `label` field to the `Itinerary` resource.

3.66 2.2.1 (2016-06-06)

- Added `audience` field to the `Document` resource.

3.67 2.2.0 (2016-05-17)

- Added `transactional_email`, and `emails` to `Agency` resource.

3.68 2.1.2 (2016-05-17)

- Added `audience` to `Invoice` resource.

3.69 2.1.1 (2016-04-29)

- Removed invalid field, `email` from `AgencyChain`

3.70 2.1.0 (2016-04-25)

- Added new resource, `AgencyChain`

3.71 2.0.0 (2016-03-11)

The global reference to the last instantiated Client has been removed. It is now mandatory to pass in a Client instance when instantiating a Model or Resource.

In practice, this should not introduce too much changes in codebases that are using `gapipy`, since resources are mostly interacted with through a Client instance (for example, `api.tours.get(123)`, or `api.customers.create({...})`), instead of being instantiated independently. The one possible exception is unit testing: in that case, `Client.build` can be useful.

The global variable was causing issues with connection pooling when multiple client with different configurations were used at the same time.

3.72 1.1.0 (2016-03-11)

- Added new resource, `DossierFeature`

3.73 1.0.0 (2016-02-29)

- Adopted Semantic Versioning for this project.

- Refactored how the cache key is set. This is a breaking change for any modules that implemented their own cache interface. The cache modules are no longer responsible for defining the cache value, but simply storing whatever it is given into cache. The `Query` object now introduces a `query_key` function which generates the cache key sent to the cache modules.

3.74 0.6.3 (2016-01-21)

- Added better error handling to *Client.build*. An AttributeError raised when instantiating a resource won't be shadowed by the except block anymore.

3.75 0.6.2 (2016-01-20)

- Fixed a regression bug when initializing DepartureServiceRoom model.

3.76 0.6.1 (2016-01-20)

- Fixed a regression bug when initializing services.

3.77 0.6.0 (2016-01-20)

- Fixed a bug when initializing list of resources.

3.78 0.5.5 (2016-01-08)

- Added a component of type `ACCOMMODATION` to `Itineraries`.

3.79 0.5.4 (2016-01-04)

- Added `associated_services` to `SingleSupplementService`

3.80 0.5.3 (2015-12-31)

- Added `name` to `Departure`.
- Happy New Year!

3.81 0.5.2 (2015-12-15)

- Added `variation_id` to `BaseCache` to fix a `TypeError` when using the `NullCache`

3.82 0.5.1 (2015-12-14)

- Add `associated_agency` to `bookings` resource

3.83 0.5.0 (2015-12-10)

- Minor adjusted in Query internals to ensure the `variation_id` of an Itinerary is handled properly.
- Added `ItineraryHighlights` and `ItineraryMedia` resources. These are sub resources of the `Itinerary`

3.84 0.4.6 (2015-12-09)

- Added connection pool caching to `RedisCache`. Instances of `gapipy` with the same cache settings (in the same Python process) will share a connection pool.

3.85 0.4.5 (2015-11-05)

- Added `code` field to the `type` of an `Itinerary`'s listed `details`.

3.86 0.4.4 (2015-11-04)

- Added the `details` field to the `Itinerary` resource – a list of textual details about an itinerary.

3.87 0.4.3 (2015-11-03)

- Added the `tour_dossier` field to the `Itinerary` resource.

3.88 0.4.2 (2015-10-28)

- Fixed a bug that would cause `amount` when looking at `Promotion` objects in the `Departure` to be removed from the data dict.

3.89 0.4.1 (2015-10-16)

- Moved an import of `requests` down from the module level. Fixes issues in CI environments.

3.90 0.4.0 (2015-10-13)

- Added connection pooling options, see docs for details on `connection_pool_options`.

3.91 0.3.0 (2015-09-24)

- Modified how the `Promotion` object is loaded within `price_bands` on a `Departure`. It now correctly captures the `amount` field.

3.92 0.2.0 (2015-09-15)

- Modified objects within `cache` module to handle `variation_id`, which is exposed within the `Itinerary` object. Previously, the `Itinerary` would not be correctly stored in cache with its variant reference.

3.93 0.1.51 (2015-08-31)

- Added the `components` field to the `Departure` resource.

3.94 0.1.50 (2015-07-28)

- Fixed an issue with the default `gapipy.cache.NullCache` when `is_cached` was used.

3.95 0.1.49 (2015-07-23)

- Added new fields to `Itinerary` revolving around variations.
- Added `declined_reason` to all service resources.

3.96 0.1.48 (2015-07-15)

- Add DeclinedReason resource

3.97 0.1.47 (2015-07-08)

- Fixed a bug in `APIRequestor.get`. Requesting a resource with with an id of `0` won't raise an Exception anymore.

3.98 0.1.46 (2015-06-10)

- Added `associated_services` and `original_departure_service` to various service resources and `departure_services` model respectively.

3.99 0.1.45 (2015-05-27)

- Fixed `products` within the `Promotion` resource to properly retain `type` and `sub_type` fields after being parsed into a dictionary.

3.100 0.1.44 (2015-05-22)

- Changed default *cache_backend* to use *gapipy.cache.NullCache*. Previously, *SimpleCache* was the default and led to confusion in production environments, specifically as to why resources were not matching the API output. Now, by default, to get any caching from gapipy you must explicitly set it.

3.101 0.1.43 (2015-04-29)

- Fixed *Place* init with empty admin_divisions

3.102 0.1.42 (2015-04-29)

- Added *description* to *TourCategory* resource.

3.103 0.1.41 (2015-04-14)

- Added *DepartureComponent* resource. See the [official G API documentation for details](https://developers.gadventures.com/docs/departure_component.html)

3.104 0.1.40 (2015-04-06)

- Added *deposit* to *DepartureService* model

3.105 0.1.39 (2015-03-31)

- Refactor `APIRequestor._request`. While this should not change existing functionality, it is now possible to override specific methods on `APIRequestor` if needed.

3.106 0.1.38 (2015-03-23)

- Fixed: Due to inconsistencies in the G API with regards to nested resources, the *fetch* function was modified to use the raw data from the API, rather than a specific set of allowed fields.

3.107 0.1.37 (2015-03-23)

- Fixed: Iterating over `products` within the `promotions` object now works as expected. Previously, accessing the `products` attribute would result in a Query object with incorrect parameters.

3.108 0.1.36 (2015-03-17)

- Support free to amount price range formatting (e.g. Free-10CAD)

3.109 0.1.35 (2015-03-12)

- Added *duration_min* & *duration_max* to *ActivityDossier* model

3.110 0.1.34 (2015-03-11)

- Added *OptionalActivity* model
- All Dossiers with *details*: * Now represented as list of *DossierDetail* models * Added convenience methods for retrieving specific details
- *ItineraryComponent* and *ActivityDossier* use new *Duration* model for their *duration* field/property
- Added *duration_label* and *location_label* to *ItineraryComponent*
- Added *duration_label*, *price_per_person_label*, and *price_per_group_label* to *ActivityDossier*

3.111 0.1.33 (2015-03-02)

- Added *name* field to the Itinerary resource.

3.112 0.1.32 (2015-02-18)

- Changed cache key creation to account for *GAPI_LANGUAGE* when the environment variable is set.

3.113 0.1.31 (2015-02-18)

- Fixed a bug when setting _resource_fields in `DepartureService` resource

3.114 0.1.30 (2015-02-11)

- `TourDossier.structured_itineraries` now refers to a list of Itinerary resources

3.115 0.1.29 (2015-02-10)

- Added `TransportDossier` and `Itinerary` resources.
- The reference to the itinerary in a `DepartureService` is now a full-fledged `Itinerary` resource.

3.116 0.1.28 (2015-01-22)

- Bug fix to correctly send `Content-Type: application/json` in POST, PUT, or PATCH.

3.117 0.1.27 (2015-01-19)

- Update `DepartureService` object to contain a reference to its `Itinerary`

3.118 0.1.26 (2015-01-14)

- Normalize API request headers, to promote caching.

3.119 0.1.25 (2015-01-09)

- Added `ActivityDossier` and `AccommodationDossier` resources, as well as references to it from `Activity` and `Accommodation`.

3.120 0.1.24 (2015-01-07)

- Added `PlaceDossier` resource, as well as reference to it from `Place`

3.121 0.1.22 (2014-12-12)

- Added `advertised_departures` to `TourDossier`

3.122 0.1.21 (2014-11-26)

- Fixed a bug with promotions on a Price object. When promotions were accessed, gapipy would query for all promotions, rather than returning the inline list.

3.123 0.1.20 (2014-11-20)

- Departure resource is now listable via filters.

3.124 0.1.19 (2014-11-17)

- Fixed a bug with *RedisCache.is_cached* where it would not use the set *key_prefix* when checking for existence in cache. Effectively, it would always return False

3.125 0.1.18 (2014-11-12)

- When setting a date_field, initiate it as a *datetime.date* type.

3.126 0.1.17 (2014-11-07)

- Deprecated *RedisHashCache* from cache backends available by default. Was not well tested or reliable.

3.127 0.1.16 (2014-10-28)

- Fixed a bug where if a model field received *null* as a value, it would fail. Now, if the result is *null*, the model field will have an appropriate *None* value.

3.128 0.1.15 (2014-10-23)

- Fix a bug in the DepartureRoom model. The *price_bands* attribute is now properly set.

3.129 0.1.14 (2014-10-22)

- Fixed a bug where AgencyDocument was not included in the code base.

3.130 0.1.13 (2014-10-21)

- Add `latitude`, `longitude`, and `documents` to the `Agency` resource.

3.131 0.1.12 (2014-10-20)

- `date_created` on the `Agency` resource is correctly parsed as a local time.

3.132 0.1.11 (2014-10-15)

- Improve the performance of `Resource.fetch` by handling cache get/set.

3.133 0.1.10 (2014-10-09)

- Fix a bug in AccommodationRoom price bands. The *season_dates* and *blackout_dates* attributes are now properly set.

3.134 0.1.9 (2014-09-23)

- Add *iso_639_3* and *iso_639_1* to *Language*

3.135 0.1.8 (2014-09-17)

- Remove the *add_ons* field in *Departure*, and add *addons*.

3.136 0.1.7 (2014-08-22)

- Fix a bug when initializing AccommodationRoom from cached data.

3.137 0.1.6 (2014-08-19)

- Add Query.purge_cached

3.138 0.1.5 (2014-07-29)

- Add *details* field to the list of *incomplete_requirements* in a *DepartureService*.

3.139 0.1.4 (2014-07-21)

- Removed sending of header *X-HTTP-Method-Override: PATCH* when the update command is called. Now, when *.save(partial=True)* is called, the correct PATCH HTTP method will be sent with the request.

3.140 0.1.3 (2014-07-18)

- Return `None` instead of raising a HTTPError 404 exception when fetching a non-existing resource by id.
- Added ability to create resources from the Query objects on the client instance. e.g.: `api.customers.create({'name': {'legal_first_name': 'Pat', ...}, ...})`

3.141 0.1.2 (2014-07-14)

- Added Query.is_cached
- Added cache options

3.142 0.1.1 (2014-06-27)

- Use setuptools find_packages

3.143 0.1.0 (2014-06-20)

- First release on PyPI.

CHAPTER 4

Indices and tables

- genindex
- modindex
- search