Tilapya Documentation

Release 1.0.0

Carson Lam

Feb 24, 2020

Contents

Tilapya is a Python wrapper around the TransLink Open API, which provides real-time transit information for the Metro Vancouver region.

Tilapya has three interfaces which correspond directly to components of the TransLink Open API:

- *RTTI*: Real-Time Transit Information
- RTDS: Regional Traffic Data System
- *GTFSRT*: GTFS-realtime feeds

Tilapya is more than a thin wrapper around the underlying REST APIs. Where possible, it smooths over some inconvenient return values, and guarantees a consistent schema in returned errors and responses.

CHAPTER 1

Installation

Install Tilapya using pip:

```
$ pip install tilapya
```

The source is also available on GitHub.

CHAPTER 2

Getting started

Use of the TransLink Open API, and thus Tilapya, requires an API key. If you don't already have an API key, you can get one by registering for an account at https://developer.translink.ca/Account/Register.

Tilapya's documentation is at http://tilapya.readthedocs.io. Tilapya's API docs contain examples for common operations.

CHAPTER 3

Documentation

3.1 Real-Time Transit Information

Tilapya wrapper around TransLink's Real-Time Transit Information (RTTI) API.

Note: This API is limited to real-time information for buses. In addition, some routes and vehicles may not be available. Buses that are not in service are not exposed by the API.

See also:

TransLink's RTTI API reference. Much of it is replicated here for convenience. However, the docs here reflect Tilapya-specific behaviour.

3.1.1 Usage examples

Listing 1: Find the name of bus stop 53095, and whether it's wheelchair-accessible.

```
>>> from tilapya import RTTI
>>> api = RTTI('my key')
>>> stop = api.stop('53095')
>>> stop.Name
'WB DOVER ST FS ROYAL OAK AVE'
>>> stop.WheelchairAccess
False
```

Listing 2: Get all the route map KML links for bus route 324.

```
>>> route = api.route('324')
>>> [pattern.RouteMap.Href for pattern in route.Patterns]
['http://nb.translink.ca/geodata/trip/324-NB1.kmz', 'http://nb.translink.ca/geodata/
↪trip/324-NB1L.kmz', 'http://nb.translink.ca/geodata/trip/324-SB1.kmz']
```

(continues on next page)

(continued from previous page)

Listing 3: Find the last reported route and position of bus 2543.

```
>>> bus = api.bus('2543')
>>> f'{bus.RouteNo} {bus.Destination} ({bus.Direction})'
'020 VICTORIA (SOUTH)'
>>> bus.Latitude, bus.Longitude
(49.2805, -123.11725)
>>> bus.RecordedTime.isoformat()
'2018-02-19T22:07:57-08:00'
```

Listing 4: Get the next two predicted (or scheduled) arrival times for the 502 bus at bus stop 55070.

```
>>> est = api.stop_estimates('55070', count=2, route_number='502')[0]
>>> [f'{sked.ExpectedLeaveTime.isoformat()} - {est.RouteNo} {sked.Destination}' for␣
↪sked in est.Schedules]
['2018-02-19T22:30:00-08:00 - 502 LANGLEY CTR', '2018-02-19T22:58:00-08:00 - 502␣
↪LANGLEY CTR']
```

3.1.2 API reference

class `tilapya.`**`RTTI`**(*api_key*, *session=None*)

The wrapper around TransLink's Real-Time Transit Information (RTTI) API.

Parameters

- **`api_key`** – TransLink API key.
- **`session`** (*`requests.Session`*) – Session to use, instead of the default.

`bus`(*bus_number*)

Get a bus by its bus vehicle number.

Parameters **`bus_number`** – A vehicle id. It is not possible to get a bus that is not currently in service.

Note: This endpoint erroneously rejects 5-digit bus numbers.

Return type *Bus*

`buses`(*stop_number=None*, *route_number=None*)

Retrieve vehicle information of all or a filtered set of buses.

Parameters

- **`stop_number`** – If present, will search for buses for stop id specified.
- **`route_number`** – If present, will search for stops specific to route.

Return type list[*Bus*]

`route`(*route_number*)

Get a route by its route number.

Parameters **`route_number`** – A bus route number.

Return type *Route*

`routes` (*stop_number=None*)
Get routes.

Note: This endpoint may intermittently and incorrectly return error code 4014 (no routes for specified stop).

Parameters **`stop_number`** – If present, will search for routes passing through this stop.

Note: Though it's implied that leaving this unspecified will return all routes, in practice, this parameter is required.

Return type list[*Route*]

`status` (*service='all'*)
Gets the bus location and real-time schedule information update status.

Parameters **`service`** – A service name.

- `location` for bus location information,
- `schedule` for real-time schedule information
- `all` for both services

Return type list[*Status*]

`stop` (*stop_number*)
Get a bus stop by bus stop number.

Parameters **`stop_number`** – 5-digit bus stop number.

Return type *Stop*

`stop_estimates` (*stop_number*, *count=None*, *timeframe=None*, *route_number=None*)
Gets the next bus estimates for a particular stop. Returns schedule data if estimates are not available.

Parameters

- **`stop_number`** – A five-digit stop number.
- **`count`** (*`int`*) – The number of buses to return. Default 6.
- **`timeframe`** (*`int`*) – The search time frame in minutes. Default 120.
- **`route_number`** – If present, will search for stops specific to route.

Returns A list of `StopEstimate`. Appears to be grouped by route, destination, and direction (not documented).

Return type list[*StopEstimate*]

`stops` (*lat*, *long*, *radius_m=None*, *route_number=None*)
Search for stops around a certain point.

Parameters

- **`lat`** (*`float`*) – Latitude.
- **`long`** (*`float`*) – Longitude.

- **radius_m** (*int*) – Search this radius for stops. Default 500. Maximum 2000.
- **route_number** – Search for stops served by this route.

Return type list[*Stop*]

3.1.3 Response objects

These returned objects are read-only. Do not attempt to modify their fields.

class tilapya.rtti.**Stop**

Stops are locations where buses provide scheduled service.

Variables

- **StopNo** (*int*) – The 5-digit stop number.
- **Name** – The stop name.
- **BayNo** – The bay number, if applicable.
- **City** – The city in which the stop is located.
- **OnStreet** – The street name the stop is located on.
- **AtStreet** – The intersecting street of the stop.
- **Latitude** (*float*) – The latitude of the stop.
- **Longitude** (*float*) – The longitude of the stop.
- **WheelchairAccess** (*bool*) – Specifies wheelchair accessible stop.
- **Distance** – Distance away from the search location.
- **Routes** (*list[Route]*) – The list of routes that the stop services.

Create new instance of Stop(StopNo, Name, BayNo, City, OnStreet, AtStreet, Latitude, Longitude, WheelchairAccess, Distance, Routes)

class tilapya.rtti.**StopEstimate**

Bus arrival estimates for a route at a stop.

Variables

- **RouteNo** – The bus route number.
- **RouteName** – The bus route name.
- **Direction** – The direction of the route at the specific stop.
- ***RouteMap*** (RouteMap) – The element containing the route map information.
- **Schedules** (*list[Schedule]*) – The element containing the list of schedules.

Create new instance of StopEstimate(RouteNo, RouteName, Direction, RouteMap, Schedules)

class tilapya.rtti.**RouteMap**

Bus route map.

Variables **href** – The location of the route map file in KMZ format.

Create new instance of RouteMap(Href,)

class tilapya.rtti.**Schedule**

A piece of real-time or scheduled arrival time information for a single bus.

Variables

- ***Pattern*** – The pattern of the specific trip.
- **Destination** – The destination of the trip.
- **ExpectedLeaveTime** (*datetime*) – The expected departure time of the trip at the specific stop. The original value is something like "05:20pm 2018-02-18". This is converted to an absolute datetime with time zone. Seconds are always 0.
- **ExpectedCountDown** (*int*) – The expected departure time in minutes.
- **ScheduleStatus** – The status of the trip.
  - * indicates scheduled time
  - − indicates delay
  - + indicates bus is running ahead of schedule
- **AddedTrip** (*bool*) – Indicates if trip is added.
- **CancelledTrip** (*bool*) – Indicates if trip is cancelled.
- **CancelledStop** (*bool*) – Indicates if stop is cancelled.
- **AddedTrip** – Indicates if trip is added.
- **AddedStop** (*bool*) – Indicates if stop is added.
- **LastUpdate** (*datetime*) – The last updated time of the trip. The original value is something like "05:20:30 pm". This is converted to an absolute datetime with time zone.

Create new instance of Schedule(Pattern, Destination, ExpectedLeaveTime, ExpectedCountdown, ScheduleStatus, CancelledTrip, CancelledStop, AddedTrip, AddedStop, LastUpdate)

class tilapya.rtti.**Bus**

Information about a bus.

Variables

- **VehicleNo** – The vehicle number of the bus.
- **TripId** (*int*) – The id of the trip the bus currently running.
- **RouteNo** – The route number of the vehicle.
- **Direction** – The direction of the trip.
- **Destination** – The destination headsign of the trip. *This field is not in the RTTI API documentation.*
- ***Pattern*** – The pattern of the trip.
- **Latitude** (*float*) – The latitude of the vehicle location.
- **Longitude** (*float*) – The longitude of the vehicle location.
- **RecordedTime** (*datetime*) – The recorded time of the last location of the vehicle. The original value is something like "05:20:30 pm". This is converted to an absolute datetime with time zone.
- ***RouteMap*** (RouteMap) – The element containing the route map information.

Create new instance of Bus(VehicleNo, TripId, RouteNo, Direction, Destination, Pattern, Latitude, Longitude, RecordedTime, RouteMap)

class tilapya.rtti.**Route**

Routes are a sequenced pattern of service.

Variables

- `RouteNo` – The bus route number.
- `Name` – The name of the route.
- `OperatingCompany` – The operating company of the route.
- `patterns` (*list [Pattern]*) – The list of patterns for the route.

Create new instance of Route(RouteNo, Name, OperatingCompany, Patterns)

class tilapya.rtti.**Pattern**

A route trip pattern.

Variables

- `PatternNo` – The pattern number.
- `Destination` – The destination of the pattern.
- *`RouteMap`* (RouteMap) – The element containing the route map information.
- `Direction` – The direction of the pattern.

Create new instance of Pattern(PatternNo, Destination, RouteMap, Direction)

class tilapya.rtti.**Status**

Status info for a service within the RTTI API.

Variables

- `name` – The name of the service (“Location” or “Schedule”)
- `value` – The status of the service (“Online” or “Offline”)

Create new instance of Status(Name, Value)

3.2 GTFS-realtime

GTFS-realtime is a specification for sharing real-time transit information. It’s primarily ingested by Google in order to power transit information within Google Maps.

See also:

TransLink’s GTFS-realtime API reference. Much of it is replicated here for convenience. However, the docs here reflect Tilapya-specific behaviour.

3.2.1 Usage example

Listing 5: Request the real-time positions feed.

```
>>> from tilapya import GTFSRT
>>> api = GTFSRT('my key')
>>> api.position()
<Response [200]>
```

The protobuf data in `response.content` can then be deserialized.

3.2.2 API reference

class `tilapya.`**`GTFSRT`**(*api_key*, *session=None*)

The wrapper around TransLink's endpoints for GTFS-realtime datasets.

Parameters

- **`api_key`** – TransLink API key.
- **`session`** (*`requests.Session`*) – Session to use, instead of the default.

`position`()

Request the position feed.

Returns The response. The raw protobuf data is in `content`.

Return type requests.Response

`service_alerts`()

Request the service alerts feed.

Returns The response. The raw protobuf data is in `content`.

Return type requests.Response

`trip_updates`()

Request the trip updates feed.

Returns The response. The raw protobuf data is in `content`.

Return type requests.Response

3.3 Error handling

If an error response is received from the TransLink Open API, *`tilapya.TransLinkAPIError`* is raised. This class parses the error body and exposes its values as members.

class `tilapya.`**`TransLinkAPIError`**(*response*)

An error response from the TransLink API.

Variables

- **`code`** – API error code. Only applies to the RTTI API.
- **`message`** – Message from the error response. Empty if the response body is invalid or empty.
- **`request`** – The request that led to the error response.
- **`response`** – The original response.

`description`

The documented description of the error code, if any.

If there was a problem deserializing the response according to the expected schema, `marshmallow.exceptions.ValidationError` is raised from the marshmallow library. This should not occur unless the underlying API changes significantly.

3.4 Testing

Tilapya is thoroughly tested, with the goal of verifying both Tilapya and the Translink Open API.

Tests are written using pytest, and are in the `tests` directory.

For performance and reproducibility, requests and responses for tests are cached using vcrpy. This info is stored in `tests/cassettes`.

3.4.1 Running the tests

To run the tests, an API key for the TransLink Open API must be provided in an environment variable named `TRANSLINK_API_KEY`.

Ensure you have Tilapya's test dependencies:

```
> pipenv install --dev
```

Then, to run the tests:

```
> pipenv run py.test tests
```

The use of prerecorded responses can be configured using `--vcr-record-mode`. See pytest-vcr docs for details.

3.4.2 Testing strategy

Generally, Tilapya's tests are written with two goals:

- Coverage of Tiliapya's own Python code
- Coverage of the TransLink Open API's documented and undocumented error states

HTTP status codes are usually ignored, as they have no consistent semantics.

3.5 History

3.5.1 1.0.0 (2019-06-23)

- Drop support for Python 2.7 and 3.4.
- Remove RTDS API. It was removed upstream.
- Switch GTFS-RT to v2 endpoints. v1 was removed upstream.
- Add service alerts endpoint to GTFS-RT. It was added to v2 upstream.

3.5.2 0.2.0 (2019-02-03)

- Require Marshmallow >= 3.0.0rc1, to fix dependency resolution problem.
- Require Requests >= 2.20.0.
- Replace API for GTFS-RT.

3.5.3 0.1.0 (2018-02-19)

- Initial version.

CHAPTER 4

License

CHAPTER 5

Indices and tables

- genindex
- modindex
- search

Python Module Index

Index