pyflight Documentation

Release 0.1.3

Volcyy

Feb 02, 2018

Contents

1 API Reference **3**
1.1 Basic Configuration . . . 3
1.2 Making Requests . . . 3
1.3 Working with the Response . . . 7

2 Indices and tables **9**

Contents:

CHAPTER 1

API Reference

This page shows the functions and classes exposed by pyflight. A lot of attributes wrap the required parameters for the QPX API and thus result in documentation similiar to the one found on the official QPX Express API reference, licensed under the Creative Commons Attribution 3.0 License.

1.1 Basic Configuration

`pyflight.`**`set_api_key`**(*key: str*)

Set the API key to use with the API.

Parameters **`key`** (`str`) – The API key to execute requests with.

1.2 Making Requests

`class` `pyflight.`**`Request`**

Represents a Request that can be sent to the API instead of using a dictionary manually.

Please note that each Request requires at least 1 adult or senior passenger. Optional attributes default to `None`.

`raw_data`

dict – The raw JSON / dictionary data which will be sent to the API.

`adult_count`

int – The amount of passengers that are adults.

`children_count`

int – The amount of passengers that are children.

`infant_in_lap_count`

int – The amount of passengers that are infants travelling in the lap of an adult.

`infant_in_seat_count`

int – The amount of passengers that are infants assigned a seat.

`senior_count`
int – The amount of passengers that are senior citizens.

`max_price`
Optional[str] – The maximum price below which results should be returned. The currency is specified in ISO-4217, and setting this attribute is validated using the regex `[A-Z]{3}\d+(\.\d+)?`. If it does not match, a `ValueError` is raised.

`sale_country`
Optional[str] – The IATA country code representing the point of sale. Determines the currency.

`ticketing_country`
Optional[str] – The IATA country code representing the point of ticketing, for example `DE`.

`refundable`
Optional[bool] – Whether to return only results with refundable fares or not.

`solution_count`
int – The amount of solutions to return. Defaults to 1, maximum is 500. Raises a `ValueError` when trying to assign a value outside of 1 to 500.

`add_slice`(*slice_: pyflight.requester.Slice*)
Adds a slice to this Request.

Parameters **`slice`** (*`Slice`*) – The Slice to be added to the request.

Returns To ease chaining of this function, `self` is returned.

Return type self

`as_dict`() → dict
Returns the raw data associated with this request, which is sent to the API when calling send_sync or send_async.

`send_sync`(*use_containers: bool = True*) → typing.Union[pyflight.result.Result, dict]
Synchronously execute a request.

Internally, this calls *`pyflight.send_sync()`*. You can also call the function directly. For further information, please view documentation for *`pyflight.send_sync()`*.

`send_async`(*use_containers: bool = True*) → typing.Union[pyflight.result.Result, dict]
Asynchronously execute a request.

Internally, this calls *`pyflight.send_async()`*. You can also call the function directly. For further information, please view documentation for *`pyflight.send_async()`*.

`adult_count`
The amount of passengers that are adults.

`children_count`
The amount of passengers that are children.

`infant_in_lap_count`
The amount of passengers that are infants travelling in the lap of an adult.

`infant_in_seat_count`
The amount of passengers that are infants assigned a seat.

`senior_count`
The amount of passengers that are senior citizens.

`max_price`
The maximum price below which results should be returned, specified in ISO-421 format.

sale_country
The IATA country code representing the point of sale. Determines the currency.

ticketing_country
The IATA country code representing the point of ticketing, for example `DE`.

refundable
Whether to return only results with refundable fares or not.

solution_count
The amount of solutions to return. Defaults to 1.

class `pyflight.`**Slice**(*origin: str*, *destination: str*, *date: str*)
Represents a slice that makes up a single itinerary of this trip.

For example, for one-way trips, usually one slice is used. A round trip would use two slices. (e.g. SFO - FRA - SFO)

Optional attributes default to `None` or an empty list if applicable, but can be set if wanted.

raw_data
dict – The raw JSON / dictionary data which will be sent to the API.

origin
str – The airport or city IATA designator of the origin.

destination
str – The airport or city IATA designator of the destination.

date
str – The date on which this flight should take place, in the format YYYY-MM-DD.

max_stops
Optional[int] – The maximum amount of stops that the passenger(s) are willing to accept on this slice.

max_connection_duration
Optional[int] – The longest duration (in minutes) between two legs that passengers are willing to accept

preferred_cabin
Optional[str] – The preferred cabin for this slice. Allowed values are COACH, PREMIUM_COACH, BUSINESS, and FIRST. A `ValueError` is raised if a value is assigned that is not listed above.

earliest_departure_time
Optional[str] – The earliest time for departure, local to the point of departure. Formatted as HH:MM.

latest_departure_time
Optional[str] – The latest time for departure, local to the point of departure. Formatted as HH:MM.

permitted_carriers
List[str] – A list of 2-letter IATA airline designators for which results should be returned.

prohibited_carriers
List[str] – A list of 2-letter IATA airline designators, for which no results will be returned.

origin
The airport or city IATA designator of the origin.

destination
The airport or city IATA designator of the destination.

date
The date on which this flight should take place, in the format YYYY-MM-DD.

max_stops
The maximum amount of stops that the passenger(s) are willing to accept on this slice.

max_connection_duration
The longest duration (in minutes) between two legs that passengers are willing to accept

preferred_cabin
The preferred cabin for this slice. Allowed values are COACH, PREMIUM_COACH, BUSINESS, and FIRST. A `ValueError` is raised if a value is assigned that is not listed above.

earliest_departure_time
The earliest time for departure, local to the point of departure. Formatted as HH:MM.

latest_departure_time
The latest time for departure, local to the point of departure. Formatted as HH:MM.

permitted_carriers
A list of 2-letter IATA airline designators for which results should be returned.

prohibited_carriers
A list of 2-letter IATA airline designators,

for which no results will be returned.

`pyflight.`**`send_async`**`(`*request_body: typing.Union[dict, pyflight.requester.Request], use_containers: bool = True*`)`

Asynchronously execute and send a JSON Request or a *`Request`*. This is a coroutine - calling this function must be awaited.

Parameters

- **`request_body`** (*`Union[dict, Request]`*) – The body of the request to be sent to the API. This must follow the structure described here: https://developers.google.com/qpx-express/v1/trips/search It is heavily recommended to use *`Request`* instead of constructing request bodies manually.
- **`use_containers`** (*`Optional[bool]`*) – Whether the containers given should be used or not. If False is given, any API call will return a dictionary of the "raw" API data without any modification. Otherwise, an API call will return a `Result` object.

Raises *`APIException`* – If the API call did not return the normal *200* status code and thus, an error occurred.

Returns

- `Result` – If `use_containers` is `True` and no Error occurred.
- *dict* – If `use_containers` is `False`, as a raw dictionary without any adjustments.

`pyflight.`**`send_sync`**`(`*request_body: typing.Union[dict, pyflight.requester.Request], use_containers: bool = True*`)`
Synchronously execute and send a JSON-Request or a :class:‘Request. Note that this function is blocking.

Parameters

- **`request_body`** (*`Union[dict, Request]`*) – The body of the request to be sent to the API. This must follow the structure described here: https://developers.google.com/qpx-express/v1/trips/search It is heavily recommended to use *`Request`* instead of constructing request bodies manually.

- **use_containers** (*Optional[bool]*) – Whether the containers given should be used or not. If False is given, any API call will return a dictionary of the "raw" API data without any modification. Otherwise, the API call will return a `Result` object.

Raises *APIException* – If the API call did not return the normal *200* status code and thus, an error occurred.

Returns

- `Result` – If `use_containers` is `True` and no Error occurred.
- *dict* – If `use_containers` is ''False', as a raw dictionary without any adjustments.

class pyflight.**APIException**(*code: int*, *message: str*, *reason: str*, **args*, ***kwargs*)

Custom Exception that is raised from the Requests when an API call goes wrong, meaning the API did not return a status code of 200.

code

int – The code of the Error that was returned

message

str – The error message as returned by the API

reason

str – The reason as specified by the API

Examples

```
try:
    flight_info = send_sync(my_request_body, use_containers=False)
except pyflight.APIException as err:
    print('Error trying to execute a request:')
    print(err)
else:
    ...
```

The Exception will be formatted as: *'<status-code>: <error-message> (reason)'*, for example `400: Bad Request (keyInvalid)`

1.3 Working with the Response

CHAPTER 2

Indices and tables

- genindex
- modindex
- search

Index

A

add_slice() (pyflight.Request method), 4
adult_count (pyflight.Request attribute), 3, 4
APIException (class in pyflight), 7
as_dict() (pyflight.Request method), 4

C

children_count (pyflight.Request attribute), 3, 4
code (pyflight.APIException attribute), 7

D

date (pyflight.Slice attribute), 5
destination (pyflight.Slice attribute), 5

E

earliest_departure_time (pyflight.Slice attribute), 5, 6

I

infant_in_lap_count (pyflight.Request attribute), 3, 4
infant_in_seat_count (pyflight.Request attribute), 3, 4

L

latest_departure_time (pyflight.Slice attribute), 5, 6

M

max_connection_duration (pyflight.Slice attribute), 5, 6
max_price (pyflight.Request attribute), 4
max_stops (pyflight.Slice attribute), 5
message (pyflight.APIException attribute), 7

O

origin (pyflight.Slice attribute), 5

P

permitted_carriers (pyflight.Slice attribute), 5, 6
preferred_cabin (pyflight.Slice attribute), 5, 6
prohibited_carriers (pyflight.Slice attribute), 5, 6

R

raw_data (pyflight.Request attribute), 3
raw_data (pyflight.Slice attribute), 5
reason (pyflight.APIException attribute), 7
refundable (pyflight.Request attribute), 4, 5
Request (class in pyflight), 3

S

sale_country (pyflight.Request attribute), 4
send_async() (in module pyflight), 6
send_async() (pyflight.Request method), 4
send_sync() (in module pyflight), 6
send_sync() (pyflight.Request method), 4
senior_count (pyflight.Request attribute), 3, 4
set_api_key() (in module pyflight), 3
Slice (class in pyflight), 5
solution_count (pyflight.Request attribute), 4, 5

T

ticketing_country (pyflight.Request attribute), 4, 5