# openrouteservice-py Documentation

*Release 0.4*

**Nils Nolde**

**May 11, 2020**

# Contents


**1 Description** **3**

**2 Requirements** **5**

**3 Installation** **7**

**4 Testing** **9**

**5 Usage** **11**
- 5.1 Basic example . . . 11
- 5.2 Optimize route . . . 12
- 5.3 Decode Polyline . . . 12
- 5.4 Dry run . . . 12
- 5.5 Local ORS instance . . . 12

**6 Support** **15**

**7 Acknowledgements** **17**

**8 Library reference** **19**
- 8.1 Submodules . . . 19
- 8.2 openrouteservice.client module . . . 19
- 8.3 openrouteservice.convert module . . . 20
- 8.4 openrouteservice.directions module . . . 20
- 8.5 openrouteservice.isochrones module . . . 22
- 8.6 openrouteservice.distance_matrix module . . . 23
- 8.7 openrouteservice.geocode module . . . 24
- 8.8 openrouteservice.elevation module . . . 27
- 8.9 openrouteservice.places module . . . 28
- 8.10 openrouteservice.optimization module . . . 29
- 8.11 openrouteservice.exceptions module . . . 30
- 8.12 Module contents . . . 30

**9 Indices and tables** **33**
- 9.1 Quickstart . . . 33
  - 9.1.1 Description . . . 33
  - 9.1.2 Requirements . . . 34
  - 9.1.3 Installation . . . 34

9.1.4 Testing . . . 34
9.1.5 Usage . . . 34
9.1.5.1 Basic example . . . 34
9.1.5.2 Optimize route . . . 35
9.1.5.3 Decode Polyline . . . 35
9.1.5.4 Dry run . . . 36
9.1.5.5 Local ORS instance . . . 36
9.1.6 Support . . . 36
9.1.7 Acknowledgements . . . 36
9.2 Library reference . . . 37
9.2.1 Submodules . . . 37
9.2.2 openrouteservice.client module . . . 37
9.2.3 openrouteservice.convert module . . . 37
9.2.4 openrouteservice.directions module . . . 38
9.2.5 openrouteservice.isochrones module . . . 40
9.2.6 openrouteservice.distance_matrix module . . . 41
9.2.7 openrouteservice.geocode module . . . 42
9.2.8 openrouteservice.elevation module . . . 45
9.2.9 openrouteservice.places module . . . 45
9.2.10 openrouteservice.optimization module . . . 46
9.2.11 openrouteservice.exceptions module . . . 47
9.2.12 Module contents . . . 48

**Python Module Index** **49**

**Index** **51**

# CHAPTER 1

# Description

The openrouteservice library gives you painless access to the openrouteservice (ORS) routing API's. It performs requests against our API's for

- directions
- isochrones
- matrix routing calculations
- places
- elevation
- Pelias geocoding
- Pelias reverse geocoding
- Pelias structured geocoding
- Pelias autocomplete
- Optimization

For further details, please visit:

- homepage
- ORS API documentation
- openrouteservice-py documentation

We also have a repo with a few useful examples here.

For support, please ask our forum.

By using this library, you agree to the ORS terms and conditions.

CHAPTER 2

# Requirements

openrouteservice-py is tested against CPython 2.7, 3.4, 3.5, 3.6, 3.7 and 3.8-dev, and PyPy3.5.

For setting up a testing environment, install `requirements-dev.txt`:

```
pip install -r requirements-dev.txt
```

CHAPTER 3

# Installation

To install from PyPI, simply use pip:

```
pip install openrouteservice
```

To install the latest and greatest from source:

```
pip install git+git://github.com/GIScience/openrouteservice-py@development
```

For `conda` users:

```
conda install -c nilsnolde openrouteservice
```

This command group will install the library to your global environment. Also works in virtual environments.

CHAPTER 4

# Testing

If you want to run the unit tests, see *Requirements*. `cd` to the library directory and run:

```
nosetests -v
```

`-v` flag for verbose output (recommended).

# CHAPTER 5

# Usage

For an interactive Jupyter notebook have a look on mybinder.org.

## 5.1 Basic example

```
import openrouteservice

coords = ((8.34234,48.23424),(8.34423,48.26424))

client = openrouteservice.Client(key='') # Specify your personal API key
routes = client.directions(coords)

print(routes)
```

For convenience, all request performing module methods are wrapped inside the `client` class. This has the disadvantage, that your IDE can't auto-show all positional and optional arguments for the different methods. And there are a lot!

The slightly more verbose alternative, preserving your IDE's smart functions, is

```
import openrouteservice
from openrouteservice.directions import directions

    coords = ((8.34234,48.23424),(8.34423,48.26424))

    client = openrouteservice.Client(key='') # Specify your personal API key
    routes = directions(client, coords) # Now it shows you all arguments for .
↪directions
```

## 5.2 Optimize route

If you want to optimize the order of multiple waypoints in a simple Traveling Salesman Problem, you can pass a `optimize_waypoints` parameter:

```
import openrouteservice

coords = ((8.34234,48.23424),(8.34423,48.26424), (8.34523,48.24424), (8.41423,48.
↪21424))

client = openrouteservice.Client(key='') # Specify your personal API key
routes = client.directions(coords, profile='cycling-regular', optimize_waypoints=True)

print(routes)
```

## 5.3 Decode Polyline

By default, the directions API returns encoded polylines. To decode to a `dict`, which is a GeoJSON geometry object, simply do

```
import openrouteservice
from openrouteservice import convert

coords = ((8.34234,48.23424),(8.34423,48.26424))

client = openrouteservice.Client(key='') # Specify your personal API key

# decode_polyline needs the geometry only
geometry = client.directions(coords)['routes'][0]['geometry']

decoded = convert.decode_polyline(geometry)

print(decoded)
```

## 5.4 Dry run

Although errors in query creation should be handled quite decently, you can do a dry run to print the request and its parameters:

```
import openrouteservice

coords = ((8.34234,48.23424),(8.34423,48.26424))

client = openrouteservice.Client()
client.directions(coords, dry_run='true')
```

## 5.5 Local ORS instance

If you're hosting your own ORS instance, you can alter the `base_url` parameter to fit your own:

```
import openrouteservice

coords = ((8.34234,48.23424),(8.34423,48.26424))

# key can be omitted for local host
client = openrouteservice.Client(base_url='http://localhost/ors')

# Only works if you didn't change the ORS endpoints manually
routes = client.directions(coords)

# If you did change the ORS endpoints for some reason
# you'll have to pass url and required parameters explicitly:
routes = client.request(
  url='/new_url',
  post_json={
      'coordinates': coords,
      'profile': 'driving-car',
      'format': 'geojson'
  })
```

CHAPTER 6

# Support

For general support and questions, contact our forum.

For issues/bugs/enhancement suggestions, please use https://github.com/GIScience/openrouteservice-py/issues.

CHAPTER 7

# Acknowledgements

This library is based on the very elegant codebase from googlemaps.

# CHAPTER 8

# Library reference

## 8.1 Submodules

## 8.2 openrouteservice.client module

Core client functionality, common across all API requests.

**class** openrouteservice.client.**Client**(*key=None*, *base_url='https://api.openrouteservice.org'*, *timeout=60*, *retry_timeout=60*, *requests_kwargs=None*, *retry_over_query_limit=True*)

Bases: object

Performs requests to the ORS API services.

**req**

Returns request object. Can be used in case of request failure.

**request**(*url*, *get_params=None*, *first_request_time=None*, *retry_counter=0*, *requests_kwargs=None*, *post_json=None*, *dry_run=None*)

Performs HTTP GET/POST with credentials, returning the body as JSON.

**Parameters**

- **url** (*string*) – URL path for the request. Should begin with a slash.
- **get_params** (*dict or list of key/value tuples*) – HTTP GET parameters.
- **first_request_time** (*datetime.datetime*) – The time of the first request (None if no retries have occurred).
- **retry_counter** (*int*) – The number of this retry, or zero for first attempt.
- **requests_kwargs** (*dict*) – Same extra keywords arg for requests as per __init__, but provided here to allow overriding internally on a per-request basis.
- **post_json** (*dict*) – HTTP POST parameters. Only specified by calling method.

- **dry_run** (*string*) – If 'true', only prints URL and parameters. 'true' or 'false'.

**Raises**

- ***ApiError*** – when the API returns an error.
- ***Timeout*** – if the request timed out.

**Return type** dict from JSON response.

## 8.3 openrouteservice.convert module

Converts Python types to string representations suitable for ORS API server.

openrouteservice.convert.**decode_polyline**(*polyline*, *is3d=False*)

Decodes a Polyline string into a GeoJSON geometry.

**Parameters**

- **polyline** (*string*) – An encoded polyline, only the geometry.
- **is3d** (*boolean*) – Specifies if geometry contains Z component.

**Returns** GeoJSON Linestring geometry

**Return type** dict

## 8.4 openrouteservice.directions module

Performs requests to the ORS directions API.

openrouteservice.directions.**directions**(*client*, *coordinates*, *profile='driving-car'*, *format_out=None*, *format='json'*, *preference=None*, *units=None*, *language=None*, *geometry=None*, *geometry_simplify=None*, *instructions=None*, *instructions_format=None*, *alternative_routes=None*, *roundabout_exits=None*, *attributes=None*, *maneuvers=None*, *radiuses=None*, *bearings=None*, *skip_segments=None*, *continue_straight=None*, *elevation=None*, *extra_info=None*, *suppress_warnings=None*, *optimized=None*, *optimize_waypoints=None*, *options=None*, *validate=True*, *dry_run=None*)

Get directions between an origin point and a destination point.

For more information, visit https://go.openrouteservice.org/documentation/.

**Parameters**

- **coordinates** – The coordinates tuple the route should be calculated from. In order of visit.
- **profile** (*string*) – Specifies the mode of transport to use when calculating directions. One of ["driving-car", "driving-hgv", "foot-walking", "foot-hiking", "cycling-regular", "cycling-road","cycling-mountain", "cycling-electric",]. Default "driving-car".

- **format** (*str*) – Specifies the response format. One of ['json', 'geojson', 'gpx']. Default "json". Geometry format for "json" is Google's encodedpolyline. The GPX schema the response is validated against can be found here: https://raw.githubusercontent.com/GIScience/openrouteservice-schema/master/gpx/v1/ors-gpx.xsd.
- **format_out** – DEPRECATED.
- **preference** (*string*) – Specifies the routing preference. One of ["fastest, "shortest", "recommended"]. Default "fastest".
- **units** (*string*) – Specifies the distance unit. One of ["m", "km", "mi"]. Default "m".
- **language** (*string*) – Language for routing instructions. One of ["en", "de", "cn", "es", "ru", "dk", "fr", "it", "nl", "br", "se", "tr", "gr"]. Default "en".
- **language** – The language in which to return results.
- **geometry** (*boolean*) – Specifies whether geometry should be returned. Default True.
- **geometry_simplify** (*boolean*) – Specifies whether to simplify the geometry. Default False.
- **instructions** (*boolean*) – Specifies whether to return turn-by-turn instructions. Default True.
- **instructions_format** (*string*) – Specifies the the output format for instructions. One of ["text", "html"]. Default "text".
- **alternative_routes** (*dict[int|float]*) – Specifies whether alternative routes are computed, and parameters for the algorithm determining suitable alternatives. Expects 3 keys: share_factor (float), target_count (int), weight_factor (float). More on https://openrouteservice.org/dev/#/api-docs/v2/directions/{profile}/geojson/post.
- **roundabout_exits** (*boolean*) – Provides bearings of the entrance and all passed roundabout exits. Adds the 'exit_bearings' array to the 'step' object in the response. Default False.
- **attributes** (*list or tuple of strings*) – Returns route attributes on ["detourfactor", "percentage"]. Must be a list of strings. Default None.
- **maneuvers** – Specifies whether the maneuver object is included into the step object or not. Default: false.

:type maneuvers bool

**Parameters**

- **radiuses** (*list or tuple*) – A list of maximum distances (measured in meters) that limit the search of nearby road segments to every given waypoint. The values must be greater than 0, the value of -1 specifies no limit in the search. The number of radiuses must correspond to the number of waypoints. Default None.
- **bearings** (*list or tuple or lists or tuples*) – Specifies a list of pairs (bearings and deviations) to filter the segments of the road network a waypoint can snap to. For example bearings=[[45,10],[120,20]]. Each pair is a comma-separated list that can consist of one or two float values, where the first value is the bearing and the second one is the allowed deviation from the bearing. The bearing can take values between 0 and 360 clockwise from true north. If the deviation is not set, then the default value of 100 degrees is used. The number of pairs must correspond to the number of waypoints. Setting optimized=false is mandatory for this feature to work for all profiles. The number of bearings corresponds to the length of waypoints-1 or waypoints. If the bearing information for the

last waypoint is given, then this will control the sector from which the destination waypoint may be reached.

- **skip_segments** (*list[int]*) – Specifies the segments that should be skipped in the route calculation. A segment is the connection between two given coordinates and the counting starts with 1 for the connection between the first and second coordinate.
- **continue_straight** (*boolean*) – Forces the route to keep going straight at waypoints restricting u-turns even if u-turns would be faster. This setting will work for all profiles except for driving-*. In this case you will have to set optimized=false for it to work. True or False. Default False.
- **elevation** (*boolean*) – Specifies whether to return elevation values for points. Default False.
- **extra_info** (*list or tuple of strings*) – Returns additional information on ["steepness", "suitability", "surface", "waycategory", "waytype", "tollways", "traildifficulty", "roadaccessrestrictions"]. Must be a list of strings. Default None.
- **suppress_warnings** (*bool*) – Tells the system to not return any warning messages and corresponding extra_info. For false the extra information can still be explicitly requested by adding it with the extra_info parameter.
- **optimized** (*bool*) – If set False, forces to not use Contraction Hierarchies.
- **options** (*dict*) – Refer to https://openrouteservice.org/dev/#/api-docs/v2/directions/{profile}/geojson/post for detailed documentation. Construct your own dict() following the example of the minified options object. Will be converted to json automatically.
- **optimize_waypoints** (*bool*) – If True, a Vroom instance (ORS optimization endpoint) will optimize the *via* waypoints, i.e. all coordinates between the first and the last. It assumes the first coordinate to be the start location and the last coordinate to be the end location. Only requests with a minimum of 4 coordinates, no routing options and fastest weighting. Default False.
- **validate** (*bool*) – Specifies whether parameters should be validated before sending the request. Default True.
- **dry_run** – Print URL and parameters without sending the request.
- **dry_run** – boolean

**Raises**

- **ValueError** – When parameter has wrong value.
- **TypeError** – When parameter is of wrong type.

**Returns** sanitized set of parameters

**Return type** call to Client.request()

## 8.5 openrouteservice.isochrones module

Performs requests to the ORS isochrones API.

openrouteservice.isochrones.**isochrones**(*client*, *locations*, *profile='driving-car'*, *range_type='time'*, *range=None*, *intervals=None*, *segments=None*, *interval=None*, *units=None*, *location_type=None*, *smoothing=None*, *attributes=None*, *validate=True*, *dry_run=None*)

Gets travel distance and time for a matrix of origins and destinations.

**Parameters**

- **locations** (*list or tuple of lng,lat values*) – One pair of lng/lat values.
- **profile** (*string*) – Specifies the mode of transport to use when calculating directions. One of ["driving-car", "driving-hgv", "foot-walking", "foot-hiking", "cycling-regular", "cycling-road", "cycling-mountain", "cycling-electric",]. Default "driving-car".
- **range_type** – Set 'time' for isochrones or 'distance' for equidistants. Default 'time'.
- **intervals** (*list of integer(s)*) – [SOON DEPRECATED] replaced by *range*.
- **range** (*list of integer(s)*) – Ranges to calculate distances/durations for. This can be a list of multiple ranges, e.g. [600, 1200, 1400] or a single value list. In the latter case, you can also specify the 'interval' variable to break the single value into more isochrones. In meters or seconds.
- **segments** (*integer*) – [SOON DEPRECATED] replaced by *interval*.
- **interval** (*integer*) – Segments isochrones or equidistants for one 'range' value. Only has effect if used with a single value 'range' value. In meters or seconds.
- **units** (*string*) – Specifies the unit system to use when displaying results. One of ["m", "km", "m"]. Default "m".
- **location_type** (*string*) – 'start' treats the location(s) as starting point, 'destination' as goal. Default 'start'.
- **smoothing** (*float*) – Applies a level of generalisation to the isochrone polygons generated. Value between 0 and 1, whereas a value closer to 1 will result in a more generalised shape.
- **attributes** (*list of string(s)*) – 'area' returns the area of each polygon in its feature properties. 'reachfactor' returns a reachability score between 0 and 1. 'total_pop' returns population statistics from https://ghsl.jrc.ec.europa.eu/about.php. One or more of ['area', 'reachfactor', 'total_pop']. Default 'area'.
- **validate** (*bool*) – Specifies whether parameters should be validated before sending the request. Default True.
- **dry_run** – Print URL and parameters without sending the request.
- **dry_run** – boolean

**Raises** **ValueError** – When parameter has invalid value(s).

**Return type** call to Client.request()

## 8.6 openrouteservice.distance_matrix module

Performs requests to the ORS Matrix API.

openrouteservice.distance_matrix.**distance_matrix**(*client*, *locations*, *profile='driving-car'*, *sources=None*, *destinations=None*, *metrics=None*, *resolve_locations=None*, *units=None*, *optimized=None*, *validate=True*, *dry_run=None*)

Gets travel distance and time for a matrix of origins and destinations.

**Parameters**

- **locations** (*a single location, or a list of locations, where a location is a list or tuple of lng,lat values*) – One or more pairs of lng/lat values.
- **profile** (*string*) – Specifies the mode of transport to use when calculating directions. One of ["driving-car", "driving-hgv", "foot-walking", "foot-hiking", "cycling-regular", "cycling-road", "cycling-safe", "cycling-mountain", "cycling-tour", "cycling-electric",]. Default "driving-car".
- **sources** (*list or tuple*) – A list of indices that refer to the list of locations (starting with 0). If not passed, all indices are considered.
- **destinations** (*list or tuple*) – A list of indices that refer to the list of locations (starting with 0). If not passed, all indices are considered.
- **metrics** (*list of strings*) – Specifies a list of returned metrics. One or more of ["distance", "duration"]. Default ['duration'].
- **resolve_locations** (*boolean*) – Specifies whether given locations are resolved or not. If set 'true', every element in destinations and sources will contain the name element that identifies the name of the closest street. Default False.
- **units** (*string*) – Specifies the unit system to use when displaying results. One of ["m", "km", "m"]. Default "m".
- **optimized** (*boolean*) – Specifies whether Dijkstra algorithm ('false') or any available technique to speed up shortest-path routing ('true') is used. For normal Dijkstra the number of visited nodes is limited to 100000. Default True
- **validate** (*bool*) – Specifies whether parameters should be validated before sending the request. Default True.
- **dry_run** – Print URL and parameters without sending the request.
- **dry_run** – boolean

**Raises** **ValueError** – When profile parameter has wrong value.

**Return type** call to Client.request()

## 8.7 openrouteservice.geocode module

Performs requests to the ORS geocode API (direct Pelias clone).

openrouteservice.geocode.**pelias_autocomplete**(*client*, *text*, *focus_point=None*, *rect_min_x=None*, *rect_min_y=None*, *rect_max_x=None*, *rect_max_y=None*, *country=None*, *sources=None*, *layers=None*, *validate=True*, *dry_run=None*)

Autocomplete geocoding can be used alongside /search to enable real-time feedback. It represents a type-ahead

functionality, which helps to find the desired location, without to require a fully specified search term.

This endpoint queries directly against a Pelias instance. For fully documentation, please see https://github.com/pelias/documentation/blob/master/autocomplete.md

**Parameters**

- **text** (*string*) – Full-text query against search endpoint. Required.
- **focus_point** – Focusses the search to be around this point and gives results within a 100 km radius higher scores.
- **rect_min_x** (*float*) – Min longitude by which to constrain request geographically.
- **rect_min_y** (*float*) – Min latitude by which to constrain request geographically.
- **rect_max_x** (*float*) – Max longitude by which to constrain request geographically.
- **rect_max_y** (*float*) – Max latitude by which to constrain request geographically.
- **country** (*str*) – Constrain query by country. Accepts a alpha-2 or alpha-3 digit ISO-3166 country codes.
- **sources** (*list of strings*) – The originating source of the data. One or more of ['osm', 'oa', 'wof', 'gn']. Currently only 'osm', 'wof' and 'gn' are supported.
- **layers** (*list of strings*) – The administrative hierarchy level for the query. Refer to https://github.com/pelias/documentation/blob/master/search.md#filter-by-data-type for details.
- **dry_run** – Print URL and parameters without sending the request.
- **dry_run** – boolean

**Raises**

- **ValueError** – When parameter has invalid value(s).
- **TypeError** – When parameter is of the wrong type.

**Return type** dict from JSON response

openrouteservice.geocode.**pelias_reverse**(*client*, *point*, *circle_radius=None*, *sources=None*, *layers=None*, *country=None*, *size=None*, *validate=True*, *dry_run=None*)

Reverse geocoding is the process of converting geographic coordinates into a human-readable address.

This endpoint queries directly against a Pelias instance.

**Parameters**

- **point** (*list or tuple of [Lon, Lat]*) – Coordinate tuple. Required.
- **circle_radius** (*integer*) – Radius around point to limit query in km. Default 1.
- **sources** (*list of strings*) – The originating source of the data. One or more of ['osm', 'oa', 'wof', 'gn']. Currently only 'osm', 'wof' and 'gn' are supported.
- **layers** (*list of strings*) – The administrative hierarchy level for the query. Refer to https://github.com/pelias/documentation/blob/master/search.md#filter-by-data-type for details.
- **country** (*str*) – Constrain query by country. Accepts a alpha-2 or alpha-3 digit ISO-3166 country codes.
- **size** (*integer*) – The amount of results returned. Default 10.
- **dry_run** – Print URL and parameters without sending the request.

- **dry_run** – boolean

**Raises** **ValueError** – When parameter has invalid value(s).

**Return type** dict from JSON response

openrouteservice.geocode.**pelias_search**(*client*, *text*, *focus_point=None*, *rect_min_x=None*, *rect_min_y=None*, *rect_max_x=None*, *rect_max_y=None*, *circle_point=None*, *circle_radius=None*, *sources=None*, *layers=None*, *country=None*, *size=None*, *validate=True*, *dry_run=None*)

Geocoding is the process of converting addresses into geographic coordinates.

This endpoint queries directly against a Pelias instance.

**Parameters**

- **text** (*string*) – Full-text query against search endpoint. Required.
- **focus_point** – Focusses the search to be around this point and gives results within a 100 km radius higher scores.
- **rect_min_x** (*float*) – Min longitude by which to constrain request geographically.
- **rect_min_y** (*float*) – Min latitude by which to constrain request geographically.
- **rect_max_x** (*float*) – Max longitude by which to constrain request geographically.
- **rect_max_y** (*float*) – Max latitude by which to constrain request geographically.
- **circle_point** (*list or tuple of (Long, Lat)*) – Geographical constraint in form a circle.
- **circle_radius** (*integer*) – Radius of circle constraint in km. Default 50.
- **sources** (*list of strings*) – The originating source of the data. One or more of ['osm', 'oa', 'wof', 'gn']. Currently only 'osm', 'wof' and 'gn' are supported.
- **layers** (*list of strings*) – The administrative hierarchy level for the query. Refer to https://github.com/pelias/documentation/blob/master/search.md#filter-by-data-type for details.
- **country** (*str*) – Constrain query by country. Accepts a alpha-2 or alpha-3 digit ISO-3166 country code.
- **size** (*integer*) – The amount of results returned. Default 10.
- **dry_run** – Print URL and parameters without sending the request.
- **dry_run** – boolean

**Raises**

- **ValueError** – When parameter has invalid value(s).
- **TypeError** – When parameter is of the wrong type.

**Return type** call to Client.request()

openrouteservice.geocode.**pelias_structured**(*client*, *address=None*, *neighbourhood=None*, *borough=None*, *locality=None*, *county=None*, *region=None*, *postalcode=None*, *country=None*, *validate=True*, *dry_run=None*)

With structured geocoding, you can search for the individual parts of a location. Structured geocoding is an option on the search endpoint, which allows you to define a query that maintains the individual fields.

This endpoint queries directly against a Pelias instance. For full documentation, please see https://github.com/pelias/documentation/blob/master/structured-geocoding.md

**Parameters**

- **address** (*string*) – Can contain a full address with house number or only a street name.
- **neighbourhood** (*string*) – Neighbourhoods are vernacular geographic entities that may not necessarily be official administrative divisions but are important nonetheless.

# Check all passed arguments convert._is_valid_args(locals())

**Parameters**

- **borough** (*string*) – Mostly known in the context of New York City, even though they may exist in other cities.
- **locality** (*string*) – Localities are equivalent to what are commonly referred to as cities.
- **county** (*string*) – Administrative divisions between localities and regions. Not as commonly used in geocoding as localities, but useful when attempting to disambiguate between localities.
- **region** (*string*) – Normally the first-level administrative divisions within countries, analogous to states and provinces in the United States and Canada. Can be a full name or abbreviation.
- **postalcode** (*integer*) – Dictated by an administrative division, which is almost always countries. Postal codes are unique within a country.
- **country** (*string*) – Highest-level divisions supported in a search. Can be a full name or abbreviation.
- **dry_run** – Print URL and parameters without sending the request.
- **dry_run** – boolean

**Raises** **TypeError** – When parameter is of the wrong type.

**Return type** dict from JSON response

## 8.8 openrouteservice.elevation module

Performs requests to the ORS elevation API.

openrouteservice.elevation.**elevation_line**(*client*, *format_in*, *geometry*, *format_out='geojson'*, *dataset='srtm'*, *validate=True*, *dry_run=None*)

POSTs 2D point to be enriched with elevation.

**Parameters**

- **format_in** (*string*) – Format of input geometry. One of ['geojson', 'polyline', 'encodedpolyline']
- **geometry** (*depending on format_in, either list of coordinates, LineString geojson or string*) – Point geometry
- **format_out** (*string*) – Format of output geometry, one of ['geojson', 'polyline', 'encodedpolyline']
- **dataset** (*string*) – Elevation dataset to be used. Currently only SRTM v4.1 available.

- **dry_run** – Print URL and parameters without sending the request.
- **dry_run** – boolean

**Returns** correctly formatted parameters

**Return type** Client.request()

openrouteservice.elevation.**elevation_point**(*client*, *format_in*, *geometry*, *format_out='geojson'*, *dataset='srtm'*, *validate=True*, *dry_run=None*)

POSTs 2D point to be enriched with elevation.

**Parameters**

- **format_in** (*string*) – Format of input geometry. One of ['geojson', 'point']
- **geometry** (*depending on format_in, either list of coordinates or Point geojson*) – Point geometry
- **format_out** (*string*) – Format of output geometry, one of ['geojson', 'point']
- **dataset** (*string*) – Elevation dataset to be used. Currently only SRTM v4.1 available.
- **dry_run** – Print URL and parameters without sending the request.
- **dry_run** – boolean

**Returns** correctly formatted parameters

**Return type** Client.request()

## 8.9 openrouteservice.places module

Performs requests to the ORS Places API.

openrouteservice.places.**places**(*client*, *request*, *geojson=None*, *bbox=None*, *buffer=None*, *filter_category_ids=None*, *filter_category_group_ids=None*, *filters_custom=None*, *limit=None*, *sortby=None*, *validate=True*, *dry_run=None*)

Gets POI's filtered by specified parameters.

**Parameters**

- **request** (*string*) – Type of request. One of ['pois', 'list', 'stats']. 'pois': returns geojson of pois; 'stats': returns statistics of passed categories; 'list': returns mapping of category ID's to textual representation.
- **geojson** (*dict*) – GeoJSON dict used for the query.
- **buffer** – Buffers geometry of 'geojson' or 'bbox' with the specified value in meters.
- **filter_category_ids** – Filter by ORS custom category IDs. See https://github.com/GIScience/openrouteservice-docs#places-response for the mappings.
- **filter_category_group_ids** – Filter by ORS custom high-level category groups. See https://github.com/GIScience/openrouteservice-docs#places-response for the mappings.
- **filters_custom** (*dict of lists/str*) – Specify additional filters by key/value. Default ORS filters are 'name': free text 'wheelchair': ['yes', 'limited', 'no', 'designated'] 'smoking': ['dedicated','yes','separated','isolated', 'no', 'outside'] 'fee': ['yes','no', 'str']
- **limit** (integer base_url='http://localhost:5000') – limit for POI queries.

- **sortby** (*string*) – Sorts the returned features by 'distance' or 'category'. For request='pois' only.
- **dry_run** – Print URL and parameters without sending the request.
- **dry_run** – boolean

**Return type** call to Client.request()

## 8.10 openrouteservice.optimization module

Performs requests to the ORS optimization API.

**class** openrouteservice.optimization.**Job**(*id*, *location=None*, *location_index=None*, *service=None*, *amount=None*, *skills=None*, *priority=None*, *time_windows=None*)

Bases: object

Class to create a Job object for optimization endpoint.

Full documentation at https://github.com/VROOM-Project/vroom/blob/master/docs/API.md#jobs.

**class** openrouteservice.optimization.**Shipment**(*pickup=None*, *delivery=None*, *amount=None*, *skills=None*, *priority=None*)

Bases: object

Class to create a Shipment object for optimization endpoint.

Full documentation at https://github.com/VROOM-Project/vroom/blob/master/docs/API.md#shipments.

**class** openrouteservice.optimization.**ShipmentStep**(*id=None*, *location=None*, *location_index=None*, *service=None*, *time_windows=None*)

Bases: object

Class to create a Shipment object for optimization endpoint.

Full documentation at https://github.com/VROOM-Project/vroom/blob/master/docs/API.md#shipments.

**class** openrouteservice.optimization.**Vehicle**(*id*, *profile='driving-car'*, *start=None*, *start_index=None*, *end=None*, *end_index=None*, *capacity=None*, *skills=None*, *time_window=None*)

Bases: object

Class to create a Vehicle object for optimization endpoint.

Full documentation at https://github.com/VROOM-Project/vroom/blob/master/docs/API.md#vehicles.

openrouteservice.optimization.**optimization**(*client*, *jobs=None*, *vehicles=None*, *shipments=None*, *matrix=None*, *geometry=None*, *dry_run=None*)

Optimize a fleet of vehicles on a number of jobs.

For more information, visit https://github.com/VROOM-Project/vroom/blob/master/docs/API.md.

Example:

```
>>> from openrouteservice import Client, optimization
>>> coordinates = [[8.688641, 49.420577], [8.680916, 49.415776]]
>>> jobs, vehicles = list(), list()
```

(continues on next page)

(continued from previous page)

```
>>> for idx, coord in enumerate(coordinates):
        jobs.append(optimization.Job(id=idx, location=coord))
        vehicles.append(optimization.Vehicle(id=idx, location=coord))
>>> api = Client(key='somekey')
>>> result = api.optimization(jobs=jobs, vehicles=vehicles)
```

**Parameters**

- **jobs** (*list of Job*) – The Job objects to fulfill.
- **vehicles** (*list of Vehicle*) – The vehicles to fulfill the *openrouteservice.optimization.Job*'s.
- **shipments** (*list of Shipment*) – The Shipment objects to fulfill.
- **matrix** (*list of lists of int*) – Specify a custom cost matrix. If not specified, it will be calculated with the `openrouteservice.matrix.matrix()` endpoint.
- **geometry** (*bool*) – If the geometry of the resulting routes should be calculated. Default False.
- **dry_run** – Print URL and parameters without sending the request.
- **dry_run** – boolean

**Returns** Response of optimization endpoint.

**Return type** dict

## 8.11 openrouteservice.exceptions module

Defines exceptions that are thrown by the ORS client.

**exception** openrouteservice.exceptions.**ApiError**(*status*, *message=None*)
Bases: exceptions.Exception

Represents an exception returned by the remote API.

**exception** openrouteservice.exceptions.**HTTPError**(*status_code*)
Bases: exceptions.Exception

An unexpected HTTP error occurred.

**exception** openrouteservice.exceptions.**Timeout**
Bases: exceptions.Exception

The request timed out.

**exception** openrouteservice.exceptions.**ValidationError**(*errors*)
Bases: exceptions.Exception

Something went wrong during cerberus validation

## 8.12 Module contents

openrouteservice.**get_ordinal**(*number*)
Produces an ordinal (1st, 2nd, 3rd, 4th) from a number

- genindex
- modindex
- search

CHAPTER 9

# Indices and tables

## 9.1 Quickstart

### 9.1.1 Description

The openrouteservice library gives you painless access to the openrouteservice (ORS) routing API's. It performs requests against our API's for

- directions
- isochrones
- matrix routing calculations
- places
- elevation
- Pelias geocoding
- Pelias reverse geocoding
- Pelias structured geocoding
- Pelias autocomplete
- Optimization

For further details, please visit:

- homepage
- ORS API documentation
- openrouteservice-py documentation

We also have a repo with a few useful examples here.

For support, please ask our forum.

By using this library, you agree to the ORS terms and conditions.

### 9.1.2 Requirements

openrouteservice-py is tested against CPython 2.7, 3.4, 3.5, 3.6, 3.7 and 3.8-dev, and PyPy3.5.

For setting up a testing environment, install `requirements-dev.txt`:

```
pip install -r requirements-dev.txt
```

### 9.1.3 Installation

To install from PyPI, simply use pip:

```
pip install openrouteservice
```

To install the latest and greatest from source:

```
pip install git+git://github.com/GIScience/openrouteservice-py@development
```

For `conda` users:

```
conda install -c nilsnolde openrouteservice
```

This command group will install the library to your global environment. Also works in virtual environments.

### 9.1.4 Testing

If you want to run the unit tests, see *Requirements*. `cd` to the library directory and run:

```
nosetests -v
```

`-v` flag for verbose output (recommended).

### 9.1.5 Usage

For an interactive Jupyter notebook have a look on mybinder.org.

#### 9.1.5.1 Basic example

```
import openrouteservice

coords = ((8.34234,48.23424),(8.34423,48.26424))

client = openrouteservice.Client(key='') # Specify your personal API key
routes = client.directions(coords)

print(routes)
```

For convenience, all request performing module methods are wrapped inside the `client` class. This has the disadvantage, that your IDE can't auto-show all positional and optional arguments for the different methods. And there are a lot!

The slightly more verbose alternative, preserving your IDE's smart functions, is

```
import openrouteservice
from openrouteservice.directions import directions

    coords = ((8.34234,48.23424),(8.34423,48.26424))

    client = openrouteservice.Client(key='') # Specify your personal API key
    routes = directions(client, coords) # Now it shows you all arguments for .
↪directions
```

#### 9.1.5.2 Optimize route

If you want to optimize the order of multiple waypoints in a simple Traveling Salesman Problem, you can pass a `optimize_waypoints` parameter:

```
import openrouteservice

coords = ((8.34234,48.23424),(8.34423,48.26424), (8.34523,48.24424), (8.41423,48.
↪21424))

client = openrouteservice.Client(key='') # Specify your personal API key
routes = client.directions(coords, profile='cycling-regular', optimize_waypoints=True)

print(routes)
```

#### 9.1.5.3 Decode Polyline

By default, the directions API returns encoded polylines. To decode to a `dict`, which is a GeoJSON geometry object, simply do

```
import openrouteservice
from openrouteservice import convert

coords = ((8.34234,48.23424),(8.34423,48.26424))

client = openrouteservice.Client(key='') # Specify your personal API key

# decode_polyline needs the geometry only
geometry = client.directions(coords)['routes'][0]['geometry']
```

(continues on next page)

(continued from previous page)

```
decoded = convert.decode_polyline(geometry)

print(decoded)
```

#### 9.1.5.4 Dry run

Although errors in query creation should be handled quite decently, you can do a dry run to print the request and its parameters:

```
import openrouteservice

coords = ((8.34234,48.23424),(8.34423,48.26424))

client = openrouteservice.Client()
client.directions(coords, dry_run='true')
```

#### 9.1.5.5 Local ORS instance

If you're hosting your own ORS instance, you can alter the `base_url` parameter to fit your own:

```
import openrouteservice

coords = ((8.34234,48.23424),(8.34423,48.26424))

# key can be omitted for local host
client = openrouteservice.Client(base_url='http://localhost/ors')

# Only works if you didn't change the ORS endpoints manually
routes = client.directions(coords)

# If you did change the ORS endpoints for some reason
# you'll have to pass url and required parameters explicitly:
routes = client.request(
  url='/new_url',
  post_json={
      'coordinates': coords,
      'profile': 'driving-car',
      'format': 'geojson'
  })
```

### 9.1.6 Support

For general support and questions, contact our forum.

For issues/bugs/enhancement suggestions, please use https://github.com/GIScience/openrouteservice-py/issues.

### 9.1.7 Acknowledgements

This library is based on the very elegant codebase from googlemaps.

## 9.2 Library reference

### 9.2.1 Submodules

### 9.2.2 openrouteservice.client module

Core client functionality, common across all API requests.

**class** openrouteservice.client.**Client**(*key=None*, *base_url='https://api.openrouteservice.org'*, *timeout=60*, *retry_timeout=60*, *requests_kwargs=None*, *retry_over_query_limit=True*)

Bases: object

Performs requests to the ORS API services.

**req**

Returns request object. Can be used in case of request failure.

**request**(*url*, *get_params=None*, *first_request_time=None*, *retry_counter=0*, *requests_kwargs=None*, *post_json=None*, *dry_run=None*)

Performs HTTP GET/POST with credentials, returning the body as JSON.

**Parameters**

- **url** (*string*) – URL path for the request. Should begin with a slash.
- **get_params** (*dict or list of key/value tuples*) – HTTP GET parameters.
- **first_request_time** (*datetime.datetime*) – The time of the first request (None if no retries have occurred).
- **retry_counter** (*int*) – The number of this retry, or zero for first attempt.
- **requests_kwargs** (*dict*) – Same extra keywords arg for requests as per __init__, but provided here to allow overriding internally on a per-request basis.
- **post_json** (*dict*) – HTTP POST parameters. Only specified by calling method.
- **dry_run** (*string*) – If 'true', only prints URL and parameters. 'true' or 'false'.

**Raises**

- ***ApiError*** – when the API returns an error.
- ***Timeout*** – if the request timed out.

**Return type** dict from JSON response.

### 9.2.3 openrouteservice.convert module

Converts Python types to string representations suitable for ORS API server.

openrouteservice.convert.**decode_polyline**(*polyline*, *is3d=False*)

Decodes a Polyline string into a GeoJSON geometry.

**Parameters**

- **polyline** (*string*) – An encoded polyline, only the geometry.
- **is3d** (*boolean*) – Specifies if geometry contains Z component.

**Returns** GeoJSON Linestring geometry

**Return type** dict

### 9.2.4 openrouteservice.directions module

Performs requests to the ORS directions API.

openrouteservice.directions.**directions**(*client*, *coordinates*, *profile='driving-car'*, *format_out=None*, *format='json'*, *preference=None*, *units=None*, *language=None*, *geometry=None*, *geometry_simplify=None*, *instructions=None*, *instructions_format=None*, *alternative_routes=None*, *roundabout_exits=None*, *attributes=None*, *maneuvers=None*, *radiuses=None*, *bearings=None*, *skip_segments=None*, *continue_straight=None*, *elevation=None*, *extra_info=None*, *suppress_warnings=None*, *optimized=None*, *optimize_waypoints=None*, *options=None*, *validate=True*, *dry_run=None*)

Get directions between an origin point and a destination point.

For more information, visit https://go.openrouteservice.org/documentation/.

**Parameters**

- **coordinates** – The coordinates tuple the route should be calculated from. In order of visit.
- **profile** (*string*) – Specifies the mode of transport to use when calculating directions. One of ["driving-car", "driving-hgv", "foot-walking", "foot-hiking", "cycling-regular", "cycling-road","cycling-mountain", "cycling-electric",]. Default "driving-car".
- **format** (*str*) – Specifies the response format. One of ['json', 'geojson', 'gpx']. Default "json". Geometry format for "json" is Google's encodedpolyline. The GPX schema the response is validated against can be found here: https://raw.githubusercontent.com/GIScience/openrouteservice-schema/master/gpx/v1/ors-gpx.xsd.
- **format_out** – DEPRECATED.
- **preference** (*string*) – Specifies the routing preference. One of ["fastest, "shortest", "recommended"]. Default "fastest".
- **units** (*string*) – Specifies the distance unit. One of ["m", "km", "mi"]. Default "m".
- **language** (*string*) – Language for routing instructions. One of ["en", "de", "cn", "es", "ru", "dk", "fr", "it", "nl", "br", "se", "tr", "gr"]. Default "en".
- **language** – The language in which to return results.
- **geometry** (*boolean*) – Specifies whether geometry should be returned. Default True.
- **geometry_simplify** (*boolean*) – Specifies whether to simplify the geometry. Default False.
- **instructions** (*boolean*) – Specifies whether to return turn-by-turn instructions. Default True.
- **instructions_format** (*string*) – Specifies the the output format for instructions. One of ["text", "html"]. Default "text".

- **alternative_routes** (*dict[int|float]*) – Specifies whether alternative routes are computed, and parameters for the algorithm determining suitable alternatives. Expects 3 keys: share_factor (float), target_count (int), weight_factor (float). More on https://openrouteservice.org/dev/#/api-docs/v2/directions/{profile}/geojson/post.
- **roundabout_exits** (*boolean*) – Provides bearings of the entrance and all passed roundabout exits. Adds the 'exit_bearings' array to the 'step' object in the response. Default False.
- **attributes** (*list or tuple of strings*) – Returns route attributes on ["detourfactor", "percentage"]. Must be a list of strings. Default None.
- **maneuvers** – Specifies whether the maneuver object is included into the step object or not. Default: false.

:type maneuvers bool

**Parameters**

- **radiuses** (*list or tuple*) – A list of maximum distances (measured in meters) that limit the search of nearby road segments to every given waypoint. The values must be greater than 0, the value of -1 specifies no limit in the search. The number of radiuses must correspond to the number of waypoints. Default None.
- **bearings** (*list or tuple or lists or tuples*) – Specifies a list of pairs (bearings and deviations) to filter the segments of the road network a waypoint can snap to. For example bearings=[[45,10],[120,20]]. Each pair is a comma-separated list that can consist of one or two float values, where the first value is the bearing and the second one is the allowed deviation from the bearing. The bearing can take values between 0 and 360 clockwise from true north. If the deviation is not set, then the default value of 100 degrees is used. The number of pairs must correspond to the number of waypoints. Setting optimized=false is mandatory for this feature to work for all profiles. The number of bearings corresponds to the length of waypoints-1 or waypoints. If the bearing information for the last waypoint is given, then this will control the sector from which the destination waypoint may be reached.
- **skip_segments** (*list[int]*) – Specifies the segments that should be skipped in the route calculation. A segment is the connection between two given coordinates and the counting starts with 1 for the connection between the first and second coordinate.
- **continue_straight** (*boolean*) – Forces the route to keep going straight at waypoints restricting u-turns even if u-turns would be faster. This setting will work for all profiles except for driving-*. In this case you will have to set optimized=false for it to work. True or False. Default False.
- **elevation** (*boolean*) – Specifies whether to return elevation values for points. Default False.
- **extra_info** (*list or tuple of strings*) – Returns additional information on ["steepness", "suitability", "surface", "waycategory", "waytype", "tollways", "traildifficulty", "roadaccessrestrictions"]. Must be a list of strings. Default None.
- **suppress_warnings** (*bool*) – Tells the system to not return any warning messages and corresponding extra_info. For false the extra information can still be explicitly requested by adding it with the extra_info parameter.
- **optimized** (*bool*) – If set False, forces to not use Contraction Hierarchies.
- **options** (*dict*) – Refer to https://openrouteservice.org/dev/#/api-docs/v2/directions/{profile}/geojson/post for detailed documentation. Construct your own

dict() following the example of the minified options object. Will be converted to json automatically.

- **optimize_waypoints** (*bool*) – If True, a Vroom instance (ORS optimization endpoint) will optimize the *via* waypoints, i.e. all coordinates between the first and the last. It assumes the first coordinate to be the start location and the last coordinate to be the end location. Only requests with a minimum of 4 coordinates, no routing options and fastest weighting. Default False.
- **validate** (*bool*) – Specifies whether parameters should be validated before sending the request. Default True.
- **dry_run** – Print URL and parameters without sending the request.
- **dry_run** – boolean

**Raises**

- **ValueError** – When parameter has wrong value.
- **TypeError** – When parameter is of wrong type.

**Returns** sanitized set of parameters

**Return type** call to Client.request()

### 9.2.5 openrouteservice.isochrones module

Performs requests to the ORS isochrones API.

openrouteservice.isochrones.**isochrones**(*client*, *locations*, *profile='driving-car'*, *range_type='time'*, *range=None*, *intervals=None*, *segments=None*, *interval=None*, *units=None*, *location_type=None*, *smoothing=None*, *attributes=None*, *validate=True*, *dry_run=None*)

Gets travel distance and time for a matrix of origins and destinations.

**Parameters**

- **locations** (*list or tuple of lng,lat values*) – One pair of lng/lat values.
- **profile** (*string*) – Specifies the mode of transport to use when calculating directions. One of ["driving-car", "driving-hgv", "foot-walking", "foot-hiking", "cycling-regular", "cycling-road", "cycling-mountain", "cycling-electric",]. Default "driving-car".
- **range_type** – Set 'time' for isochrones or 'distance' for equidistants. Default 'time'.
- **intervals** (*list of integer(s)*) – [SOON DEPRECATED] replaced by *range*.
- **range** (*list of integer(s)*) – Ranges to calculate distances/durations for. This can be a list of multiple ranges, e.g. [600, 1200, 1400] or a single value list. In the latter case, you can also specify the 'interval' variable to break the single value into more isochrones. In meters or seconds.
- **segments** (*integer*) – [SOON DEPRECATED] replaced by *interval*.
- **interval** (*integer*) – Segments isochrones or equidistants for one 'range' value. Only has effect if used with a single value 'range' value. In meters or seconds.
- **units** (*string*) – Specifies the unit system to use when displaying results. One of ["m", "km", "m"]. Default "m".

- **location_type** (*string*) – 'start' treats the location(s) as starting point, 'destination' as goal. Default 'start'.
- **smoothing** (*float*) – Applies a level of generalisation to the isochrone polygons generated. Value between 0 and 1, whereas a value closer to 1 will result in a more generalised shape.
- **attributes** (*list of string(s)*) – 'area' returns the area of each polygon in its feature properties. 'reachfactor' returns a reachability score between 0 and 1. 'total_pop' returns population statistics from https://ghsl.jrc.ec.europa.eu/about.php. One or more of ['area', 'reachfactor', 'total_pop']. Default 'area'.
- **validate** (*bool*) – Specifies whether parameters should be validated before sending the request. Default True.
- **dry_run** – Print URL and parameters without sending the request.
- **dry_run** – boolean

**Raises** **ValueError** – When parameter has invalid value(s).

**Return type** call to Client.request()

### 9.2.6 openrouteservice.distance_matrix module

Performs requests to the ORS Matrix API.

openrouteservice.distance_matrix.**distance_matrix**(*client*, *locations*, *profile='driving-car'*, *sources=None*, *destinations=None*, *metrics=None*, *resolve_locations=None*, *units=None*, *optimized=None*, *validate=True*, *dry_run=None*)

Gets travel distance and time for a matrix of origins and destinations.

**Parameters**

- **locations** (*a single location, or a list of locations, where a location is a list or tuple of lng,lat values*) – One or more pairs of lng/lat values.
- **profile** (*string*) – Specifies the mode of transport to use when calculating directions. One of ["driving-car", "driving-hgv", "foot-walking", "foot-hiking", "cycling-regular", "cycling-road", "cycling-safe", "cycling-mountain", "cycling-tour", "cycling-electric",]. Default "driving-car".
- **sources** (*list or tuple*) – A list of indices that refer to the list of locations (starting with 0). If not passed, all indices are considered.
- **destinations** (*list or tuple*) – A list of indices that refer to the list of locations (starting with 0). If not passed, all indices are considered.
- **metrics** (*list of strings*) – Specifies a list of returned metrics. One or more of ["distance", "duration"]. Default ['duration'].
- **resolve_locations** (*boolean*) – Specifies whether given locations are resolved or not. If set 'true', every element in destinations and sources will contain the name element that identifies the name of the closest street. Default False.
- **units** (*string*) – Specifies the unit system to use when displaying results. One of ["m", "km", "m"]. Default "m".

- **optimized** (*boolean*) – Specifies whether Dijkstra algorithm ('false') or any available technique to speed up shortest-path routing ('true') is used. For normal Dijkstra the number of visited nodes is limited to 100000. Default True
- **validate** (*bool*) – Specifies whether parameters should be validated before sending the request. Default True.
- **dry_run** – Print URL and parameters without sending the request.
- **dry_run** – boolean

**Raises** **ValueError** – When profile parameter has wrong value.

**Return type** call to Client.request()

### 9.2.7 openrouteservice.geocode module

Performs requests to the ORS geocode API (direct Pelias clone).

openrouteservice.geocode.**pelias_autocomplete**(*client*, *text*, *focus_point=None*, *rect_min_x=None*, *rect_min_y=None*, *rect_max_x=None*, *rect_max_y=None*, *country=None*, *sources=None*, *layers=None*, *validate=True*, *dry_run=None*)

Autocomplete geocoding can be used alongside /search to enable real-time feedback. It represents a type-ahead functionality, which helps to find the desired location, without to require a fully specified search term.

This endpoint queries directly against a Pelias instance. For fully documentation, please see https://github.com/pelias/documentation/blob/master/autocomplete.md

**Parameters**

- **text** (*string*) – Full-text query against search endpoint. Required.
- **focus_point** – Focusses the search to be around this point and gives results within a 100 km radius higher scores.
- **rect_min_x** (*float*) – Min longitude by which to constrain request geographically.
- **rect_min_y** (*float*) – Min latitude by which to constrain request geographically.
- **rect_max_x** (*float*) – Max longitude by which to constrain request geographically.
- **rect_max_y** (*float*) – Max latitude by which to constrain request geographically.
- **country** (*str*) – Constrain query by country. Accepts a alpha-2 or alpha-3 digit ISO-3166 country codes.
- **sources** (*list of strings*) – The originating source of the data. One or more of ['osm', 'oa', 'wof', 'gn']. Currently only 'osm', 'wof' and 'gn' are supported.
- **layers** (*list of strings*) – The administrative hierarchy level for the query. Refer to https://github.com/pelias/documentation/blob/master/search.md#filter-by-data-type for details.
- **dry_run** – Print URL and parameters without sending the request.
- **dry_run** – boolean

**Raises**

- **ValueError** – When parameter has invalid value(s).
- **TypeError** – When parameter is of the wrong type.

**Return type** dict from JSON response

openrouteservice.geocode.**pelias_reverse**(*client*, *point*, *circle_radius=None*, *sources=None*, *layers=None*, *country=None*, *size=None*, *validate=True*, *dry_run=None*)

Reverse geocoding is the process of converting geographic coordinates into a human-readable address.

This endpoint queries directly against a Pelias instance.

**Parameters**

- **point** (*list or tuple of [Lon, Lat]*) – Coordinate tuple. Required.
- **circle_radius** (*integer*) – Radius around point to limit query in km. Default 1.
- **sources** (*list of strings*) – The originating source of the data. One or more of ['osm', 'oa', 'wof', 'gn']. Currently only 'osm', 'wof' and 'gn' are supported.
- **layers** (*list of strings*) – The administrative hierarchy level for the query. Refer to https://github.com/pelias/documentation/blob/master/search.md#filter-by-data-type for details.
- **country** (*str*) – Constrain query by country. Accepts a alpha-2 or alpha-3 digit ISO-3166 country codes.
- **size** (*integer*) – The amount of results returned. Default 10.
- **dry_run** – Print URL and parameters without sending the request.
- **dry_run** – boolean

**Raises** **ValueError** – When parameter has invalid value(s).

**Return type** dict from JSON response

openrouteservice.geocode.**pelias_search**(*client*, *text*, *focus_point=None*, *rect_min_x=None*, *rect_min_y=None*, *rect_max_x=None*, *rect_max_y=None*, *circle_point=None*, *circle_radius=None*, *sources=None*, *layers=None*, *country=None*, *size=None*, *validate=True*, *dry_run=None*)

Geocoding is the process of converting addresses into geographic coordinates.

This endpoint queries directly against a Pelias instance.

**Parameters**

- **text** (*string*) – Full-text query against search endpoint. Required.
- **focus_point** – Focusses the search to be around this point and gives results within a 100 km radius higher scores.
- **rect_min_x** (*float*) – Min longitude by which to constrain request geographically.
- **rect_min_y** (*float*) – Min latitude by which to constrain request geographically.
- **rect_max_x** (*float*) – Max longitude by which to constrain request geographically.
- **rect_max_y** (*float*) – Max latitude by which to constrain request geographically.
- **circle_point** (*list or tuple of (Long, Lat)*) – Geographical constraint in form a circle.
- **circle_radius** (*integer*) – Radius of circle constraint in km. Default 50.
- **sources** (*list of strings*) – The originating source of the data. One or more of ['osm', 'oa', 'wof', 'gn']. Currently only 'osm', 'wof' and 'gn' are supported.

- **layers** (*list of strings*) – The administrative hierarchy level for the query. Refer to https://github.com/pelias/documentation/blob/master/search.md#filter-by-data-type for details.
- **country** (*str*) – Constrain query by country. Accepts a alpha-2 or alpha-3 digit ISO-3166 country code.
- **size** (*integer*) – The amount of results returned. Default 10.
- **dry_run** – Print URL and parameters without sending the request.
- **dry_run** – boolean

**Raises**

- **ValueError** – When parameter has invalid value(s).
- **TypeError** – When parameter is of the wrong type.

**Return type** call to Client.request()

openrouteservice.geocode.**pelias_structured**(*client*, *address=None*, *neighbourhood=None*, *borough=None*, *locality=None*, *county=None*, *region=None*, *postalcode=None*, *country=None*, *validate=True*, *dry_run=None*)

With structured geocoding, you can search for the individual parts of a location. Structured geocoding is an option on the search endpoint, which allows you to define a query that maintains the individual fields.

This endpoint queries directly against a Pelias instance. For full documentation, please see https://github.com/pelias/documentation/blob/master/structured-geocoding.md

**Parameters**

- **address** (*string*) – Can contain a full address with house number or only a street name.
- **neighbourhood** (*string*) – Neighbourhoods are vernacular geographic entities that may not necessarily be official administrative divisions but are important nonetheless.

# Check all passed arguments convert._is_valid_args(locals())

**Parameters**

- **borough** (*string*) – Mostly known in the context of New York City, even though they may exist in other cities.
- **locality** (*string*) – Localities are equivalent to what are commonly referred to as cities.
- **county** (*string*) – Administrative divisions between localities and regions. Not as commonly used in geocoding as localities, but useful when attempting to disambiguate between localities.
- **region** (*string*) – Normally the first-level administrative divisions within countries, analogous to states and provinces in the United States and Canada. Can be a full name or abbreviation.
- **postalcode** (*integer*) – Dictated by an administrative division, which is almost always countries. Postal codes are unique within a country.
- **country** (*string*) – Highest-level divisions supported in a search. Can be a full name or abbreviation.
- **dry_run** – Print URL and parameters without sending the request.
- **dry_run** – boolean

**Raises** **TypeError** – When parameter is of the wrong type.

**Return type** dict from JSON response

### 9.2.8 openrouteservice.elevation module

Performs requests to the ORS elevation API.

openrouteservice.elevation.**elevation_line**(*client*, *format_in*, *geometry*, *format_out='geojson'*, *dataset='srtm'*, *validate=True*, *dry_run=None*)

POSTs 2D point to be enriched with elevation.

**Parameters**

- **format_in** (*string*) – Format of input geometry. One of ['geojson', 'polyline', 'encodedpolyline']
- **geometry** (*depending on format_in, either list of coordinates, LineString geojson or string*) – Point geometry
- **format_out** (*string*) – Format of output geometry, one of ['geojson', 'polyline', 'encodedpolyline']
- **dataset** (*string*) – Elevation dataset to be used. Currently only SRTM v4.1 available.
- **dry_run** – Print URL and parameters without sending the request.
- **dry_run** – boolean

**Returns** correctly formatted parameters

**Return type** Client.request()

openrouteservice.elevation.**elevation_point**(*client*, *format_in*, *geometry*, *format_out='geojson'*, *dataset='srtm'*, *validate=True*, *dry_run=None*)

POSTs 2D point to be enriched with elevation.

**Parameters**

- **format_in** (*string*) – Format of input geometry. One of ['geojson', 'point']
- **geometry** (*depending on format_in, either list of coordinates or Point geojson*) – Point geometry
- **format_out** (*string*) – Format of output geometry, one of ['geojson', 'point']
- **dataset** (*string*) – Elevation dataset to be used. Currently only SRTM v4.1 available.
- **dry_run** – Print URL and parameters without sending the request.
- **dry_run** – boolean

**Returns** correctly formatted parameters

**Return type** Client.request()

### 9.2.9 openrouteservice.places module

Performs requests to the ORS Places API.

openrouteservice.places.**places**(*client*, *request*, *geojson=None*, *bbox=None*, *buffer=None*, *filter_category_ids=None*, *filter_category_group_ids=None*, *filters_custom=None*, *limit=None*, *sortby=None*, *validate=True*, *dry_run=None*)

Gets POI's filtered by specified parameters.

**Parameters**

- **request** (*string*) – Type of request. One of ['pois', 'list', 'stats']. 'pois': returns geojson of pois; 'stats': returns statistics of passed categories; 'list': returns mapping of category ID's to textual representation.
- **geojson** (*dict*) – GeoJSON dict used for the query.
- **buffer** – Buffers geometry of 'geojson' or 'bbox' with the specified value in meters.
- **filter_category_ids** – Filter by ORS custom category IDs. See https://github.com/GIScience/openrouteservice-docs#places-response for the mappings.
- **filter_category_group_ids** – Filter by ORS custom high-level category groups. See https://github.com/GIScience/openrouteservice-docs#places-response for the mappings.
- **filters_custom** (*dict of lists/str*) – Specify additional filters by key/value. Default ORS filters are 'name': free text 'wheelchair': ['yes', 'limited', 'no', 'designated'] 'smoking': ['dedicated','yes','separated','isolated', 'no', 'outside'] 'fee': ['yes','no', 'str']
- **limit** (integer base_url='http://localhost:5000') – limit for POI queries.
- **sortby** (*string*) – Sorts the returned features by 'distance' or 'category'. For request='pois' only.
- **dry_run** – Print URL and parameters without sending the request.
- **dry_run** – boolean

**Return type** call to Client.request()

### 9.2.10 openrouteservice.optimization module

Performs requests to the ORS optimization API.

**class** openrouteservice.optimization.**Job**(*id*, *location=None*, *location_index=None*, *service=None*, *amount=None*, *skills=None*, *priority=None*, *time_windows=None*)

Bases: object

Class to create a Job object for optimization endpoint.

Full documentation at https://github.com/VROOM-Project/vroom/blob/master/docs/API.md#jobs.

**class** openrouteservice.optimization.**Shipment**(*pickup=None*, *delivery=None*, *amount=None*, *skills=None*, *priority=None*)

Bases: object

Class to create a Shipment object for optimization endpoint.

Full documentation at https://github.com/VROOM-Project/vroom/blob/master/docs/API.md#shipments.

**class** openrouteservice.optimization.**ShipmentStep**(*id=None*, *location=None*, *location_index=None*, *service=None*, *time_windows=None*)

Bases: object

Class to create a Shipment object for optimization endpoint.

Full documentation at https://github.com/VROOM-Project/vroom/blob/master/docs/API.md#shipments.

**class** openrouteservice.optimization.**Vehicle**(*id*, *profile='driving-car'*, *start=None*, *start_index=None*, *end=None*, *end_index=None*, *capacity=None*, *skills=None*, *time_window=None*)

Bases: object

Class to create a Vehicle object for optimization endpoint.

Full documentation at https://github.com/VROOM-Project/vroom/blob/master/docs/API.md#vehicles.

openrouteservice.optimization.**optimization**(*client*, *jobs=None*, *vehicles=None*, *shipments=None*, *matrix=None*, *geometry=None*, *dry_run=None*)

Optimize a fleet of vehicles on a number of jobs.

For more information, visit https://github.com/VROOM-Project/vroom/blob/master/docs/API.md.

Example:

```
>>> from openrouteservice import Client, optimization
>>> coordinates = [[8.688641, 49.420577], [8.680916, 49.415776]]
>>> jobs, vehicles = list(), list()
>>> for idx, coord in enumerate(coordinates):
        jobs.append(optimization.Job(id=idx, location=coord))
        vehicles.append(optimization.Vehicle(id=idx, location=coord))
>>> api = Client(key='somekey')
>>> result = api.optimization(jobs=jobs, vehicles=vehicles)
```

**Parameters**

- **jobs** (*list of Job*) – The Job objects to fulfill.
- **vehicles** (*list of Vehicle*) – The vehicles to fulfill the *openrouteservice.optimization.Job*'s.
- **shipments** (*list of Shipment*) – The Shipment objects to fulfill.
- **matrix** (*list of lists of int*) – Specify a custom cost matrix. If not specified, it will be calculated with the openrouteservice.matrix.matrix() endpoint.
- **geometry** (*bool*) – If the geometry of the resulting routes should be calculated. Default False.
- **dry_run** – Print URL and parameters without sending the request.
- **dry_run** – boolean

**Returns** Response of optimization endpoint.

**Return type** dict

### 9.2.11 openrouteservice.exceptions module

Defines exceptions that are thrown by the ORS client.

**exception** openrouteservice.exceptions.**ApiError**(*status*, *message=None*)
Bases: exceptions.Exception

Represents an exception returned by the remote API.

**exception** openrouteservice.exceptions.**HTTPError**(*status_code*)
Bases: exceptions.Exception

An unexpected HTTP error occurred.

**exception** openrouteservice.exceptions.**Timeout**
Bases: exceptions.Exception

The request timed out.

**exception** openrouteservice.exceptions.**ValidationError**(*errors*)
Bases: exceptions.Exception

Something went wrong during cerberus validation

### 9.2.12 Module contents

openrouteservice.**get_ordinal**(*number*)
Produces an ordinal (1st, 2nd, 3rd, 4th) from a number

# Python Module Index

## o

openrouteservice, 48
openrouteservice.client, 37
openrouteservice.convert, 37
openrouteservice.directions, 38
openrouteservice.distance_matrix, 41
openrouteservice.elevation, 45
openrouteservice.exceptions, 47
openrouteservice.geocode, 42
openrouteservice.isochrones, 40
openrouteservice.optimization, 46
openrouteservice.places, 45

# Index


## A

ApiError, 30, 47

## C

Client (*class in openrouteservice.client*), 19, 37

## D

decode_polyline() (*in module openrouteservice.convert*), 20, 37
directions() (*in module openrouteservice.directions*), 20, 38
distance_matrix() (*in module openrouteservice.distance_matrix*), 23, 41

## E

elevation_line() (*in module openrouteservice.elevation*), 27, 45
elevation_point() (*in module openrouteservice.elevation*), 28, 45

## G

get_ordinal() (*in module openrouteservice*), 30, 48

## H

HTTPError, 30, 48

## I

isochrones() (*in module openrouteservice.isochrones*), 22, 40

## J

Job (*class in openrouteservice.optimization*), 29, 46

## O

openrouteservice (*module*), 30, 48
openrouteservice.client (*module*), 19, 37
openrouteservice.convert (*module*), 20, 37
openrouteservice.directions (*module*), 20, 38
openrouteservice.distance_matrix (*module*), 23, 41
openrouteservice.elevation (*module*), 27, 45
openrouteservice.exceptions (*module*), 30, 47
openrouteservice.geocode (*module*), 24, 42
openrouteservice.isochrones (*module*), 22, 40
openrouteservice.optimization (*module*), 29, 46
openrouteservice.places (*module*), 28, 45
optimization() (*in module openrouteservice.optimization*), 29, 47

## P

pelias_autocomplete() (*in module openrouteservice.geocode*), 24, 42
pelias_reverse() (*in module openrouteservice.geocode*), 25, 43
pelias_search() (*in module openrouteservice.geocode*), 26, 43
pelias_structured() (*in module openrouteservice.geocode*), 26, 44
places() (*in module openrouteservice.places*), 28, 45

## R

req (*openrouteservice.client.Client attribute*), 19, 37
request() (*openrouteservice.client.Client method*), 19, 37

## S

Shipment (*class in openrouteservice.optimization*), 29, 46
ShipmentStep (*class in openrouteservice.optimization*), 29, 46

## T

Timeout, 30, 48

## V

ValidationError, 30, 48

Vehicle (*class in openrouteservice.optimization*), 29, 47