Voir quelques spécifications des API de routing et géocoding.

Ces API sont pour l'instant atteignable sur un serveur OVH privés, dont l'adresse est http://51.75.246.38

Elles ont pour vocations à être installées sur un serveur PRO de sysoco.

En attendant leur mise en place vous pouvez déjà les tester en sachant que pour le Coding seul la région PACA est fonctionnelle. Pour le routing la France entière est valide.

Pour le geocoding:

URL: http://51.75.246.38:7878

Liberté • Égalité • Fraternité
RÉPUBLIQUE FRANÇAISE

adresse.data.gouv.fr

Utilisation de l'API

/search/

Point d'entrée pour le géocodage.

Utiliser le paramètre **q** pour faire une recherche plein texte:

```
curl https://api-adresse.data.gouv.fr/search/?q=8+bd+du+port
```

Avec **limit** on peut contrôler le nombre d'éléments retournés:

```
curl https://api-adresse.data.gouv.fr/search/?q=8+bd+du+port&limit=15
```

Avec **autocomplete** on peut désactiver les traitements d'auto-complétion:

```
curl https://api-adresse.data.gouv.fr/search/?q=8+bd+du+port&autocomplete=0
```

Avec **lat** et **lon** on peut donner une priorité géographique:

```
curl https://api-adresse.data.gouv.fr/search/?q=8+bd+du+port&lat=48.789&lon=2.789
```

Les filtres **type**, **postcode** (code Postal) et **citycode** (code INSEE) permettent de restreindre la recherche:

```
curl https://api-adresse.data.gouv.fr/search/?q=8+bd+du+port&postcode=44380
curl https://api-adresse.data.gouv.fr/search/?q=paris&type=street
```

Le retour est un geojson *FeatureCollection* respectant la spec GeoCodeJSON:


```
{
  "attribution": "BAN",
  "licence": "ODbL 1.0",
  "query": "8 bd du port",
  "type": "FeatureCollection",
  "version": "draft",
  "features": [
    {
      "properties": {
        "context": "80, Somme, Picardie",
        "housenumber": "8",
        "label": "8 Boulevard du Port 80000 Amiens",
        "postcode": "80000",
```

```
        "citycode": "80021",
        "id": "ADRNIVX_0000000260875032",
        "score": 0.3351181818181818,
        "name": "8 Boulevard du Port",
        "city": "Amiens",
        "type": "housenumber"
      },
      "geometry": {
        "type": "Point",
        "coordinates": [
          2.29009,
          49.897446
        ]
      },
      "type": "Feature"
    },
    {
      "properties": {
        "context": "34, Hérault, Languedoc-Roussillon",
        "housenumber": "8",
        "label": "8 Boulevard du Port 34140 Mèze",
        "postcode": "34140",
        "citycode": "34157",
        "id": "ADRNIVX_0000000284423783",
        "score": 0.3287575757575757,
        "name": "8 Boulevard du Port",
        "city": "Mèze",
        "type": "housenumber"
      },
      "geometry": {
        "type": "Point",
        "coordinates": [
          3.605875,
          43.425232
        ]
      },
      "type": "Feature"
    }
  ]
}
```


Les attributs retournés sont :

- *id* : identifiant de l’ adresse (non stable: actuellement identifiant IGN)
- *type* : type de résultat trouvé
  - *housenumber* : numéro « à la plaque »
  - *street* : position « à la voie », placé approximativement au centre de celle-ci
  - *locality* : lieu-dit
  - *municipality* : numéro « à la commune »
- *score* : valeur de 0 à 1 indiquant la pertinence du résultat
- *housenumber* : numéro avec indice de répétition éventuel (bis, ter, A, B)
- *name* : numéro éventuel et nom de voie ou lieu dit

- *postcode* : code postal
- *citycode* : code INSEE de la commune
- *city* : nom de la commune
- *context* : n° de département, nom de département et de région
- *label* : libellé complet de l'adresse

/reverse/

Point d'entrée pour le géocodage inverse.

Les paramètres **lat** et **lon** sont obligatoires:

```
curl https://api-adresse.data.gouv.fr/reverse/?lon=2.37&lat=48.357
```

Le paramètre **type** permet forcer le type de retour:

```
curl https://api-adresse.data.gouv.fr/reverse/?lon=2.37&lat=48.357&type=street
```

Même format de réponse que pour le point d'entrée **/search/**.

/search/csv/

Point d'entrée pour le géocodage de masse à partir d'un fichier CSV.

Le fichier csv, encodé en UTF-8 et limité actuellement à 8Mo, doit être passé via le paramètre **data**:

```
curl -X POST -F data=@path/to/file.csv https://api-adresse.data.gouv.fr/search/csv/
```

Par défaut, toutes les colonnes sont concaténées pour constituer l'adresse qui sera géocodée. On peut définir les colonnes à utiliser via de multiples paramètres **columns**:

```
curl -X POST -F data=@path/to/file.csv -F columns=voie -F columns=ville
https://api-adresse.data.gouv.fr/search/csv/
```

Il est possible de préciser le nom d'une colonne contenant le **code INSEE** ou le **code Postal** pour limiter les recherches, exemple :

```
curl -X POST -F data=@path/to/file.csv -F columns=voie -F columns=ville -F
citycode=ma_colonne_code_insee https://api-adresse.data.gouv.fr/search/csv/
curl -X POST -F data=@path/to/file.csv -F columns=voie -F columns=ville -F
postcode=colonne_code_postal https://api-adresse.data.gouv.fr/search/csv/
```

/reverse/csv/

Point d'entrée pour le géocodage inverse de masse à partir d'un fichier CSV.

Le fichier csv, encodé en UTF-8 et limité actuellement à 8Mo, doit être passé via le paramètre **data**. Il doit contenir les colonnes **latitude** (ou *lat*) et **longitude** (ou *lon* ou *lng*).

```
curl -X POST -F data=@path/to/file.csv https://api-
adresse.data.gouv.fr/reverse/csv/
```

Le ROUTING

URL: http://51.75.246.38:5000

General options

All OSRM HTTP requests use a common structure.

The following syntax applies to all services, except as noted.

Requests

Parameter	Description
`service`	One of the following values: `route` , `nearest` , `table` , `match` , `trip` , `tile`
`version`	Version of the protocol implemented by the service. `v1` for all OSRM 5.x installations
`profile`	Mode of transportation, is determined statically by the Lua profile that is used to prepare the data using `osrm-extract` . Typically `car` , `bike` or `foot` if using one of the supplied profiles.
`coordinates`	String of format `{longitude},{latitude};{longitude},{latitude}[;{longitude},{latitude} ...]` or `polyline({polyline}) or polyline6({polyline6})` .
`format`	Only `json` is supported at the moment. This parameter is optional and defaults to `json` .

Passing any `option=value` is optional. `polyline` follows Google's polyline format with precision 5 by default and can be generated using this package.

To pass parameters to each location some options support an array like encoding:

Request options

Option	Values	Description
bearings	`{bearing};{bearing}[;{bearing} ...]`	Limits the search to segments with given bearing in degrees towards true north in clockwise direction.
radiuses	`{radius};{radius}[;{radius} ...]`	Limits the search to given radius in meters.
generate_hints	`true` (default), `false`	Adds a Hint to the response which can be used in subsequent requests, see `hints` parameter.
hints	`{hint};{hint}[;{hint} ...]`	Hint from previous request to derive position in street network.
approaches	`{approach};{approach}[;{approach} ...]`	Keep waypoints on curb side.
exclude	`{class}[,{class}]`	Additive list of classes to avoid, order does not matter.

Where the elements follow the following format:

Element	Values
bearing	`{value},{range}` `integer 0 .. 360,integer 0 .. 180`
radius	`double >= 0` or `unlimited` (default)
hint	Base64 `string`
approach	`curb` or `unrestricted` (default)
class	A class name determined by the profile or `none` .

```
{option}={element};{element}[;{element} ... ]
```

The number of elements must match exactly the number of locations (except for `generate_hints` and `exclude`). If you don't want to pass a value but instead use the default you can pass an empty `element` .

Example: 2nd location use the default value for `option` :

```
{option}={element};;{element}
```

GET

/{service}/{version}/{profile}/{coordinates}[.{format}]?option=value&option=value

Example Requests

```
# Query on Berlin with three coordinates:
curl 'http://router.project-
osrm.org/route/v1/driving/13.388860,52.517037;13.397634,52.529407;13.428555,52.5232
19?overview=false'

# Query on Berlin excluding the usage of motorways:
curl 'http://router.project-
osrm.org/route/v1/driving/13.388860,52.517037;13.397634,52.529407?exclude=motorway'

# Using polyline:
curl 'http://router.project-
osrm.org/route/v1/driving/polyline(ofp_Ik_vpAilAyu@te@g`E)?overview=false'
```

Responses

Every response object has a `code` property containing one of the strings below or a service dependent code:

Type	Description
`Ok`	Request could be processed as expected.
`InvalidUrl`	URL string is invalid.
`InvalidService`	Service name is invalid.
`InvalidVersion`	Version is not found.
`InvalidOptions`	Options are invalid.
`InvalidQuery`	The query string is synctactically malformed.
`InvalidValue`	The successfully parsed query parameters are invalid.
`NoSegment`	One of the supplied input coordinates could not snap to street segment.
`TooBig`	The request size violates one of the service specific request size restrictions.

- `message` is a **optional** human-readable error message. All other status types are service dependent.
- In case of an error the HTTP status code will be `400` . Otherwise the HTTP status code will be `200` and `code` will be `Ok` .

Every response object has a `data_version` propetry containing timestamp from the original OpenStreetMap file. This field is optional. It can be ommited if data_version parametr was not set on osrm-extract stage or OSM file has not `osmosis_replication_timestamp` section.

Code

Data version

Example response

```
{
"code": "Ok",
"message": "Everything worked",
"data_version": "2017-11-17T21:43:02Z"
}
```

Services

Nearest service

Snaps a coordinate to the street network and returns the nearest `n` matches.

Where `coordinates` only supports a single `{longitude},{latitude}` entry.

In addition to the general options the following options are supported for this service:

Option Values		Description
number	`integer >= 1` (default `1`)	Number of nearest segments that should be returned.

Response

- `code` if the request was successful `Ok` otherwise see the service dependent and general status codes.
- `waypoints` array of `Waypoint` objects sorted by distance to the input coordinate. Each object has at least the following additional properties:
  - `nodes` : Array of OpenStreetMap node ids.

GET

http://{server}/nearest/v1/{profile}/{coordinates}.json?number={number}

Example Requests

```
# Querying nearest three snapped locations of `13.388860,52.517037` with a bearing
between `20° – 340°`.
curl 'http://router.project-osrm.org/nearest/v1/driving/13.388860,52.517037?
number=3&bearings=0,20'
```

Example Response

```
{
   "waypoints" : [
      {
```

```
          "nodes": [
             2264199819,
             0
          ],
          "hint" :
"KSoKADRYroqUBAEAEAAAABkAAAAGAAAAAAAAABhnCQCLtwAA_0vMAKlYIQM8TMwArVghAwEAAQH1a66g",
          "distance" : 4.152629,
          "name" : "Friedrichstraße",
          "location" : [
             13.388799,
             52.517033
          ]
       },
       {
          "nodes": [
             2045820592,
             0
          ],
          "hint" :
"KSoKADRYroqUBAEABgAAAAAAAAAAAAAAKQAAABhnCQCLtwAA7kvMAAxZIQM8TMwArVghAwAAAQH1a66g",
          "distance" : 11.811961,
          "name" : "Friedrichstraße",
          "location" : [
             13.388782,
             52.517132
          ]
       },
       {
          "nodes": [
             0,
             21487242
          ],
          "hint" :
"KioKgDbbDgCUBAEAAAAAABoAAAAAAAAAPAAAABlnCQCLtwAA50vMADJZIQM8TMwArVghAwAAAQH1a66g",
          "distance" : 15.872438,
          "name" : "Friedrichstraße",
          "location" : [
             13.388775,
             52.51717
          ],
       }
    ],
    "code" : "Ok"
}
```


Route service

Finds the fastest route between coordinates in the supplied order.

In addition to the general options the following options are supported for this service:

Option	Values	Description
alternatives	`true` , `false` (default), or Number	Search for alternative routes. Passing a number `alternatives=n` searches for up to `n` alternative routes. *
steps	`true` , `false` (default)	Returned route steps for each route leg
annotations	`true` , `false` (default), `nodes` , `distance` , `duration` , `datasources` , `weight` , `speed`	Returns additional metadata for each coordinate along the route geometry.
geometries	`polyline` (default), `polyline6` , `geojson`	Returned route geometry format (influences overview and per step)
overview	`simplified` (default), `full` , `false`	Add overview geometry either full, simplified according to highest zoom level it could be display on, or not at all.
continue _ straight	`default` (default), `true` , `false`	Forces the route to keep going straight at waypoints constraining uturns there even if it would be faster. Default value depends on the profile.
waypoints	`{index};{index};{index}...`	Treats input coordinates indicated by given indices as waypoints in returned Match object. Default is to treat all input coordinates as waypoints.

* Please note that even if alternative routes are requested, a result cannot be guaranteed.

Response

- `code` if the request was successful `Ok` otherwise see the service dependent and general status codes.
- `waypoints` : Array of `Waypoint` objects representing all waypoints in order:
- `routes` : An array of `Route` objects, ordered by descending recommendation rank.

In case of error the following `code` s are supported in addition to the general ones:

Type	Description
`NoRoute`	No route found.

All other properties might be undefined.

GET

/route/v1/{profile}/{coordinates}?alternatives={true|false|number}&steps={true|false}&geometries={polyline|polyline6|geojson}&overview={full|simplified|false}&annotations={true|false}

Example Request

```
# Query on Berlin with three coordinates and no overview geometry returned:
curl 'http://router.project-osrm.org/route/v1/driving/13.388860,52.517037;13.397634,52.529407;13.428555,52.523219?overview=false'
```

Table service

Computes the duration of the fastest route between all pairs of supplied coordinates. Returns the durations or distances or both between the coordinate pairs. Note that the distances are not the shortest distance between two coordinates, but rather the distances of the fastest routes. Duration is in seconds and distances is in meters.

Options

In addition to the general options the following options are supported for this service:

Option	Values	Description
sources	`{index};{index}[;{index} ...]` or `all` (default)	Use location with given index as source.
destinations	`{index};{index}[;{index} ...]` or `all` (default)	Use location with given index as destination.
annotations	`duration` (default), `distance` , or `duration,distance`	Return the requested table or tables in response.
fallback_speed	`double > 0`	If no route found between a source/destination pair, calculate the as-the-crow-flies distance, then use this speed to estimate duration.
fallback_coordinate	`input` (default), or `snapped`	When using a `fallback_speed` , use the user-supplied coordinate (`input`), or the snapped location (`snapped`) for calculating distances.
scale_factor	`double > 0`	Use in conjunction with `annotations=durations` . Scales the table `duration` values by this number.

Unlike other array encoded options, the length of `sources` and `destinations` can be **smaller or equal** to number of input locations;

Example:

```
sources=0;5;7&destinations=5;1;4;2;3;6
```

Element	Values
index	`0 <= integer < #locations`

Response

- `code` if the request was successful `Ok` otherwise see the service dependent and general status codes.
- `durations` array of arrays that stores the matrix in row-major order. `durations[i][j]` gives the travel time from the i-th waypoint to the j-th waypoint. Values are given in seconds. Can be `null` if no route between `i` and `j` can be found.
- `distances` array of arrays that stores the matrix in row-major order. `distances[i][j]` gives the travel distance from the i-th waypoint to the j-th waypoint. Values are given in meters. Can be `null` if no route between `i` and `j` can be found. Note that computing the `distances` table is currently only implemented for CH. If `annotations=distance` or `annotations=duration,distance` is requested when running a MLD router, a `NotImplemented` error will be returned.
- `sources` array of `Waypoint` objects describing all sources in order
- `destinations` array of `Waypoint` objects describing all destinations in order
- `fallback_speed_cells` (optional) array of arrays containing `i,j` pairs indicating which cells contain estimated values based on `fallback_speed` . Will be absent if `fallback_speed` is not used.

In case of error the following `code` s are supported in addition to the general ones:

Type	Description
`NoTable`	No route found.
`NotImplemented`	This request is not supported

All other properties might be undefined.

GET

/table/v1/{profile}/{coordinates}?{sources}=[{elem}...];&{destinations}=[{elem}...]&annotations={duration|distance|duration,distance}

Example Request

```
# Returns a 3x3 duration matrix:
curl 'http://router.project-
osrm.org/table/v1/driving/13.388860,52.517037;13.397634,52.529407;13.428555,52.5232
19'

# Returns a 1x3 duration matrix
```

```
curl 'http://router.project-
osrm.org/table/v1/driving/13.388860,52.517037;13.397634,52.529407;13.428555,52.5232
19?sources=0'

# Returns a asymmetric 3x2 duration matrix with from the polyline encoded locations
`qikdcB}~dpXkkHz`:
curl 'http://router.project-
osrm.org/table/v1/driving/polyline(egs_Iq_aqAppHzbHulFzeMe`EuvKpnCglA)?
sources=0;1;3&destinations=2;4'

# Returns a 3x3 duration matrix:
curl 'http://router.project-
osrm.org/table/v1/driving/13.388860,52.517037;13.397634,52.529407;13.428555,52.5232
19?annotations=duration'

# Returns a 3x3 distance matrix for CH:
curl 'http://router.project-
osrm.org/table/v1/driving/13.388860,52.517037;13.397634,52.529407;13.428555,52.5232
19?annotations=distance'

# Returns a 3x3 duration matrix and a 3x3 distance matrix for CH:
curl 'http://router.project-
osrm.org/table/v1/driving/13.388860,52.517037;13.397634,52.529407;13.428555,52.5232
19?annotations=distance,duration'
```

Example Response


```
{
  "sources": [
    {
      "location": [
        13.3888,
        52.517033
      ],
      "hint":
"PAMAgEVJAoAUAAAAIAAAAAcAAAAAAAAArss0Qa7LNEHiVIRA4lSEQAoAAAAQAAAABAAAAAAAAADMAAAAAE
zMAKlYIQM8TMwArVghAwEA3wps52D3",
      "name": "Friedrichstraße"
    },
    {
      "location": [
        13.397631,
        52.529432
      ],
      "hint":
"WIQBgL6mAoAEAAAABgAAAAAAAAA7AAAAhU6PQHvHj0IAAAAAQbyYQgQAAAAGAAAAAAAAADsAAADMAAAAf2
7MABiJIQOCbswA_4ghAwAAXwVs52D3",
      "name": "Torstraße"
    },
    {
      "location": [
        13.428554,
        52.523239
      ],
```

```
      "hint":
"7UcAgP___38fAAAAUQAAACYAAABTAAAAhSQKQrXq5kKRbiZCWJo_Qx8AAABRAAAAJgAAAFMAAADMAAAASu
fMAOdwIQNL58wA03AhAwMAvxBs52D3",
      "name": "Platz der Vereinten Nationen"
    }
  ],
  "durations": [
    [
      0,
      192.6,
      382.8
    ],
    [
      199,
      0,
      283.9
    ],
    [
      344.7,
      222.3,
      0
    ]
  ],
  "destinations": [
    {
      "location": [
        13.3888,
        52.517033
      ],
      "hint":
"PAMAgEVJAoAUAAAAIAAAAAcAAAAAAAAArss0Qa7LNEHiVIRA4lSEQAoAAAAQAAAABAAAAAAAAADMAAAAAE
zMAKlYIQM8TMwArVghAwEA3wps52D3",
      "name": "Friedrichstraße"
    },
    {
      "location": [
        13.397631,
        52.529432
      ],
      "hint":
"WIQBgL6mAoAEAAAABgAAAAAAAAA7AAAAhU6PQHvHj0IAAAAAQbyYQgQAAAAGAAAAAAAAADsAAADMAAAAf2
7MABiJIQOCbswA_4ghAwAAXwVs52D3",
      "name": "Torstraße"
    },
    {
      "location": [
        13.428554,
        52.523239
      ],
      "hint":
"7UcAgP___38fAAAAUQAAACYAAABTAAAAhSQKQrXq5kKRbiZCWJo_Qx8AAABRAAAAJgAAAFMAAADMAAAASu
fMAOdwIQNL58wA03AhAwMAvxBs52D3",
      "name": "Platz der Vereinten Nationen"
    }
  ],
```

```
  "code": "Ok",
  "distances": [
    [
      0,
      1886.89,
      3791.3
    ],
    [
      1824,
      0,
      2838.09
    ],
    [
      3275.36,
      2361.73,
      0
    ]
  ],
  "fallback_speed_cells": [
    [ 0, 1 ],
    [ 1, 0 ]
  ]
}
```


Match service

Map matching matches/snaps given GPS points to the road network in the most plausible way. Please note the request might result multiple sub-traces. Large jumps in the timestamps (> 60s) or improbable transitions lead to trace splits if a complete matching could not be found. The algorithm might not be able to match all points. Outliers are removed if they can not be matched successfully.

In addition to the general options the following options are supported for this service:

Option	Values	Description
steps	`true` , `false` (default)	Returned route steps for each route
geometries	`polyline` (default), `polyline6` , `geojson`	Returned route geometry format (influences overview and per step)
annotations	`true` , `false` (default), `nodes` , `distance` , `duration` , `datasources` , `weight` , `speed`	Returns additional metadata for each coordinate along the route geometry.
overview	`simplified` (default), `full` , `false`	Add overview geometry either full, simplified according to highest zoom level it could be display on, or not at all.
timestamps	`{timestamp};{timestamp}[;{timestamp} ...]`	Timestamps for the input locations in seconds since UNIX epoch. Timestamps need to be monotonically increasing.
radiuses	`{radius};{radius}[;{radius} ...]`	Standard deviation of GPS precision used for map matching. If applicable use GPS accuracy.
gaps	`split` (default), `ignore`	Allows the input track splitting based on huge timestamp gaps between points.
tidy	`true` , `false` (default)	Allows the input track modification to obtain better matching quality for noisy tracks.
waypoints	`{index};{index};{index}...`	Treats input coordinates indicated by given indices as waypoints in returned Match object. Default is to treat all input coordinates as waypoints.

Parameter	Values
timestamp	`integer` seconds since UNIX epoch
radius	`double >= 0` (default 5m)

The radius for each point should be the standard error of the location measured in meters from the true location. Use `Location.getAccuracy()` on Android or `CLLocation.horizontalAccuracy` on iOS. This value is used to determine which points should be considered as candidates (larger radius means more candidates) and how likely each candidate is (larger radius means far-away candidates are penalized less). The area to search is chosen such that the correct candidate should be considered 99.9% of the time (for more details see this ticket).

Response

- `code` if the request was successful `Ok` otherwise see the service dependent and general status codes.
- `tracepoints` : Array of `Waypoint` objects representing all points of the trace in order. If the trace point was ommited by map matching because it is an outlier, the entry will be `null` . Each `Waypoint` object has the following additional properties:
  - `matchings_index` : Index to the `Route` object in `matchings` the sub-trace was matched to.
  - `waypoint_index` : Index of the waypoint inside the matched route.
  - `alternatives_count` : Number of probable alternative matchings for this trace point. A value of zero indicate that this point was matched unambiguously. Split the trace at these points for incremental map matching.
- `matchings` : An array of `Route` objects that assemble the trace. Each `Route` object has the following additional properties:
  - `confidence` : Confidence of the matching. `float` value between 0 and 1. 1 is very confident that the matching is correct.

In case of error the following `code` s are supported in addition to the general ones:

Type	Description
`NoMatch`	No matchings found.

All other properties might be undefined.

GET

/match/v1/{profile}/{coordinates}?steps={true|false}&geometries={polyline|polyline6|geojson}&overview={simplified|full|false}&annotations={true|false}

Trip service

The trip plugin solves the Traveling Salesman Problem using a greedy heuristic (farthest-insertion algorithm) for 10 or more waypoints and uses brute force for less than 10 waypoints. The returned path does not have to be the fastest path. As TSP is NP-hard it only returns an approximation. Note that all input coordinates have to be connected for the trip service to work.

In addition to the general options the following options are supported for this service:

Option	Values	Description
roundtrip	`true` (default), `false`	Returned route is a roundtrip (route returns to first location)
source	`any` (default), `first`	Returned route starts at `any` or `first` coordinate
destination	`any` (default), `last`	Returned route ends at `any` or `last` coordinate
steps	`true` , `false` (default)	Returned route instructions for each trip
annotations	`true` , `false` (default), `nodes` , `distance` , `duration` , `datasources` , `weight` , `speed`	Returns additional metadata for each coordinate along the route geometry.
geometries	`polyline` (default), `polyline6` , `geojson`	Returned route geometry format (influences overview and per step)
overview	`simplified` (default), `full` , `false`	Add overview geometry either full, simplified according to highest zoom level it could be display on, or not at all.

Fixing Start and End Points

It is possible to explicitely set the start or end coordinate of the trip. When source is set to `first` , the first coordinate is used as start coordinate of the trip in the output. When destination is set to `last` , the last coordinate will be used as destination of the trip in the returned output. If you specify `any` , any of the coordinates can be used as the first or last coordinate in the output.

However, if `source=any&destination=any` the returned round-trip will still start at the first input coordinate by default.

Currently, not all combinations of `roundtrip` , `source` and `destination` are supported. Right now, the following combinations are possible:

roundtrip	source	destination	supported
true	first	last	**yes**
true	first	any	**yes**
true	any	last	**yes**
true	any	any	**yes**
false	first	last	**yes**
false	first	any	no
false	any	last	no
false	any	any	no

- `code` : if the request was successful `Ok` otherwise see the service dependent and general status codes.
- `waypoints` : Array of `Waypoint` objects representing all waypoints in input order. Each `Waypoint` object has the following additional properties:
  - `trips_index` : Index to `trips` of the sub-trip the point was matched to.
  - `waypoint_index` : Index of the point in the trip.
- `trips` : An array of `Route` objects that assemble the trace.

In case of error the following `code` s are supported in addition to the general ones:

Type	Description
`NoTrips`	No trips found because input coordinates are not connected.
`NotImplemented`	This request is not supported

All other properties might be undefined.

GET

/trip/v1/{profile}/{coordinates}?roundtrip=
{true|false}&source{any|first}&destination{any|last}&steps={true|false}&geometries=
{polyline|polyline6|geojson}&overview={simplified|full|false}&annotations={true|false}'

Example Requests

```
# Round trip in Berlin with three stops:
curl 'http://router.project-
osrm.org/trip/v1/driving/13.388860,52.517037;13.397634,52.529407;13.428555,52.52321
9'
# Round trip in Berlin with four stops, starting at the first stop, ending at the
last:
curl 'http://router.project-
osrm.org/trip/v1/driving/13.388860,52.517037;13.397634,52.529407;13.428555,52.52321
9;13.418555,52.523215?source=first&destination=last'
```

Response

Tile service

This service generates Mapbox Vector Tiles that can be viewed with a vector-tile capable slippy-map viewer. The tiles contain road geometries and metadata that can be used to examine the routing graph. The tiles are generated directly from the data in-memory, so are in sync with actual routing results, and let you examine which roads are actually routable, and what weights they have applied.

The `x`, `y`, and `zoom` values are the same as described at https://wiki.openstreetmap.org/wiki/Slippy_map_tilenames, and are supported by vector tile viewers like Mapbox GL JS.

The response object is either a binary encoded blob with a `Content-Type` of `application/x-protobuf`, or a `404` error. Note that OSRM is hard-coded to only return tiles from zoom level 12 and higher (to avoid accidentally returning extremely large vector tiles).

Vector tiles contain two layers:

`speeds` layer:

Property	Type	Description
`speed`	`integer`	the speed on that road segment, in km/h
`is_small`	`boolean`	whether this segment belongs to a small (< 1000 node) strongly connected component
`datasource`	`string`	the source for the speed value (normally `lua profile` unless you're using the traffic update feature , in which case it contains the stem of the filename that supplied the speed value for this segment
`duration`	`float`	how long this segment takes to traverse, in seconds. This value is to calculate the total route ETA.
`weight`	`integer`	how long this segment takes to traverse, in units (may differ from `duration` when artificial biasing is applied in the Lua profiles). ACTUAL ROUTING USES THIS VALUE.
`name`	`string`	the name of the road this segment belongs to
`rate`	`float`	the value of `length/weight` - analagous to `speed` , but using the `weight` value rather than `duration` , rounded to the nearest integer
`is_startpoint`	`boolean`	whether this segment can be used as a start/endpoint for routes

`turns` layer:

Property	Type	Description
`bearing_in`	`integer`	the absolute bearing that approaches the intersection. -180 to +180, 0 = North, 90 = East
`turn_angle`	`integer`	the angle of the turn, relative to the `bearing_in` . -180 to +180, 0 = straight ahead, 90 = 90-degrees to the right
`cost`	`float`	the time we think it takes to make that turn, in seconds. May be negative, depending on how the data model is constructed (some turns get a "bonus").
`weight`	`float`	the weight we think it takes to make that turn. May be negative, depending on how the data model is constructed (some turns get a "bonus"). ACTUAL ROUTING USES THIS VALUE
`type`	`string`	the type of this turn - values like `turn` , `continue` , etc. See the `StepManeuver` for a partial list, this field also exposes internal turn types that are never returned with an API response
`modifier`	`string`	the direction modifier of the turn (`left` , `sharp left` , etc)

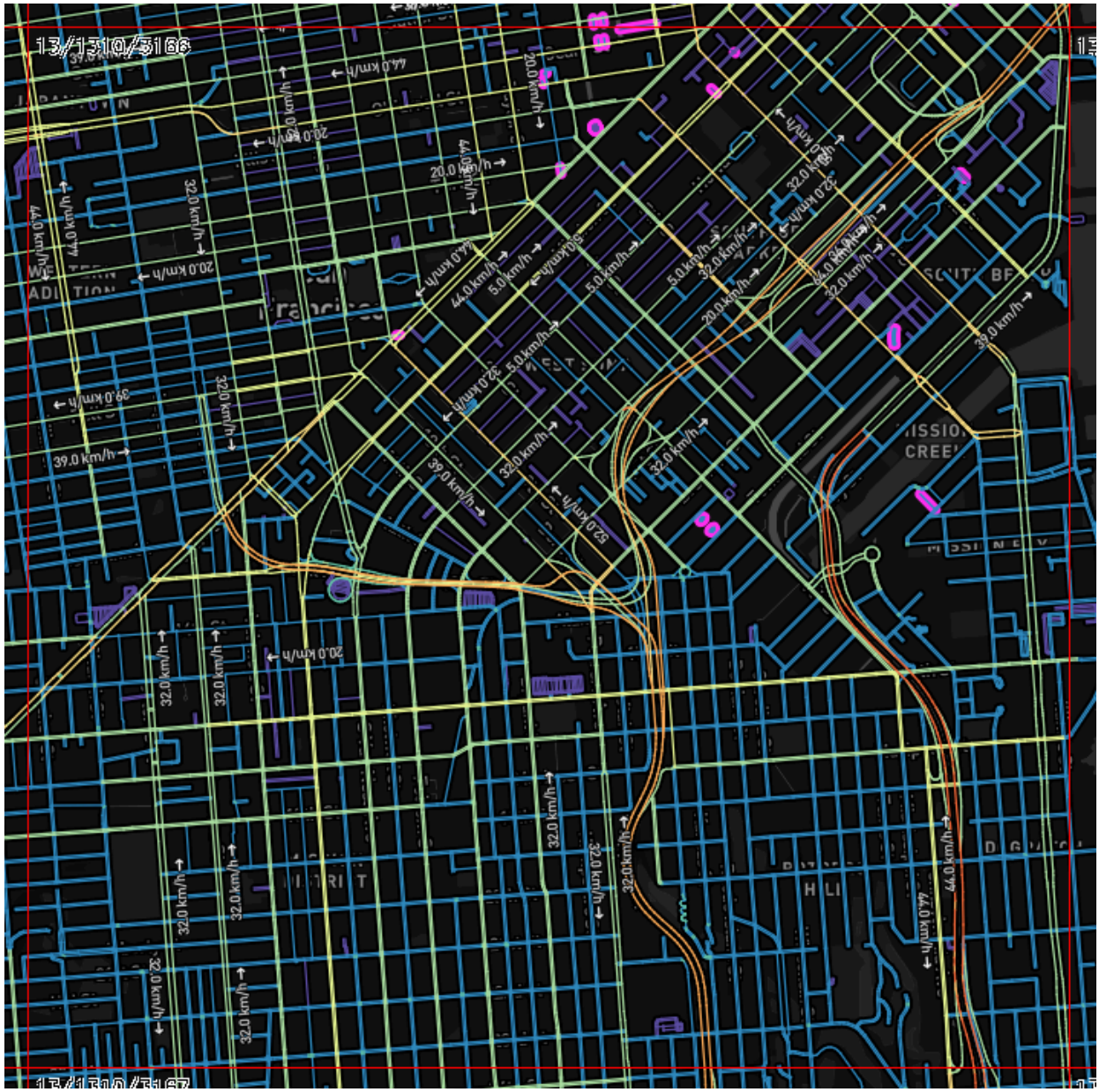
GET

/tile/v1/{profile}/tile({x},{y},{zoom}).mvt

Example request

```
# This fetches a Z=13 tile for downtown San Francisco:
curl 'http://router.project-osrm.org/tile/v1/car/tile(1310,3166,13).mvt'
```

Example response



http://map.project-osrm.org/debug/#14.33/52.5212/13.3919