

# EFA JSON API

1. Introduction
2. Common Functionality
3. SystemInfo-Request
4. StopFinder-Request
5. Trip-Request
6. DepartureMonitor-Request
7. ServingLines-Request
8. LineStop-Request
9. STT-Request
10. TTB-Request
11. ROP-Request
12. Check-MultiStopTimetable-Request
13. Coord-Request
14. GeoObject-Request
15. TripStopTimes-Request
16. StopSeqCoord-Request
17. MapRoute-Request
18. AddInfo-Request
19. Journey-Request
20. JourneyCheck-Request

21. StopList-Request
22. LineList-Request
23. StopZone-Request
24. TicketDetails-Request
25. TariffScope-Request
26. Explanatory Real Time
27. VMI

# Introduction

1. EFA Interface
2. HTTP Requests
3. Use Cases
4. Input and Output
5. HTTP Parameter Macros
6. Analysis Tools

# Introduction – EFA Interface

## About the Interface

- The EFA Intermodal Journey Planner provides an older XML and a recent JSON interface (called rapidJSON interface).
- It is controlled via a number of different HTTP requests (get unless otherwise indicated), and HTTP parameters.
- It is stateless and modular.
- Each request is matching one functionality of the EFA. Examples for this are the journey planner, the departure board, a stop sequence or the request for elements which can be displayed on an interactive map.

# Introduction – EFA Interface Version

---

- The response of the JSON interface is versioned. Which version you are receiving is shown in the response under version.
- The http parameter `version` lets you request a specific version e.g. `version=10.4.15.5`.
- If the parameter is missing the latest version is returned.

```
{  
  "version": "10.4.15.5",  
  "systemMessages": [  

```

# Introduction – HTTP Requests

## Requests and Functionality

---

### Basic Requests

- SystemInfo-Request: system information
- StopFinder-Request: stop search
- ServingLines-Request: line search
- LineStop-Request: passed stop

### Basic Journey-Planning Functionality

- Trip-Request: journey planning
- DM-Request: departures from a stop

### Print Products

- STT-Request: timetable of a stop
- TTB-Request: timetable of a line
- ROP-Request: maps of a route
- Check-MultiStopTimetable-Request: multi stop timetable available?

### Advanced Journey-Planning Functionality

- TripStopTimes-Request: stop sequence with times (including realtime)
- StopSeqCoord-Request: stop sequence with coordinates
- MapRoute-Request: maps for the route

---

### Map Requests

- Coord-Request : object coordinates
- GeoObject-Request: route coordinates

### Optional Functionality

- AddInfo-Request

### Advanced Realtime Functionality

- GetRTForTrip-Request

### Monitoring

- Journey-Request
- JourneyCheck-Request

### For internal Use

- StopList-Request: list of stops
- LineList-Request: list of lines

# Introduction – HTTP Requests Requests and Functionality

---

## Ticket Shop Functionality

- StopZone-Request: tariff information of a stop
- TicketDetails-Request: ticket details
- TariffScope-Request: tariff authorities and zones

See the following slides to get an overview of combining the HTTP requests.

# Journey Planner

① Point Search:  
StopFinder-Request

Central Station

West Station

**Search**

③ Print (PDF):  
TripRelation-Request



② Journey Planner:  
Trip-Request

11:02 - 11:35  
U1 > S3

11:25 - 11:41  
RB 351 

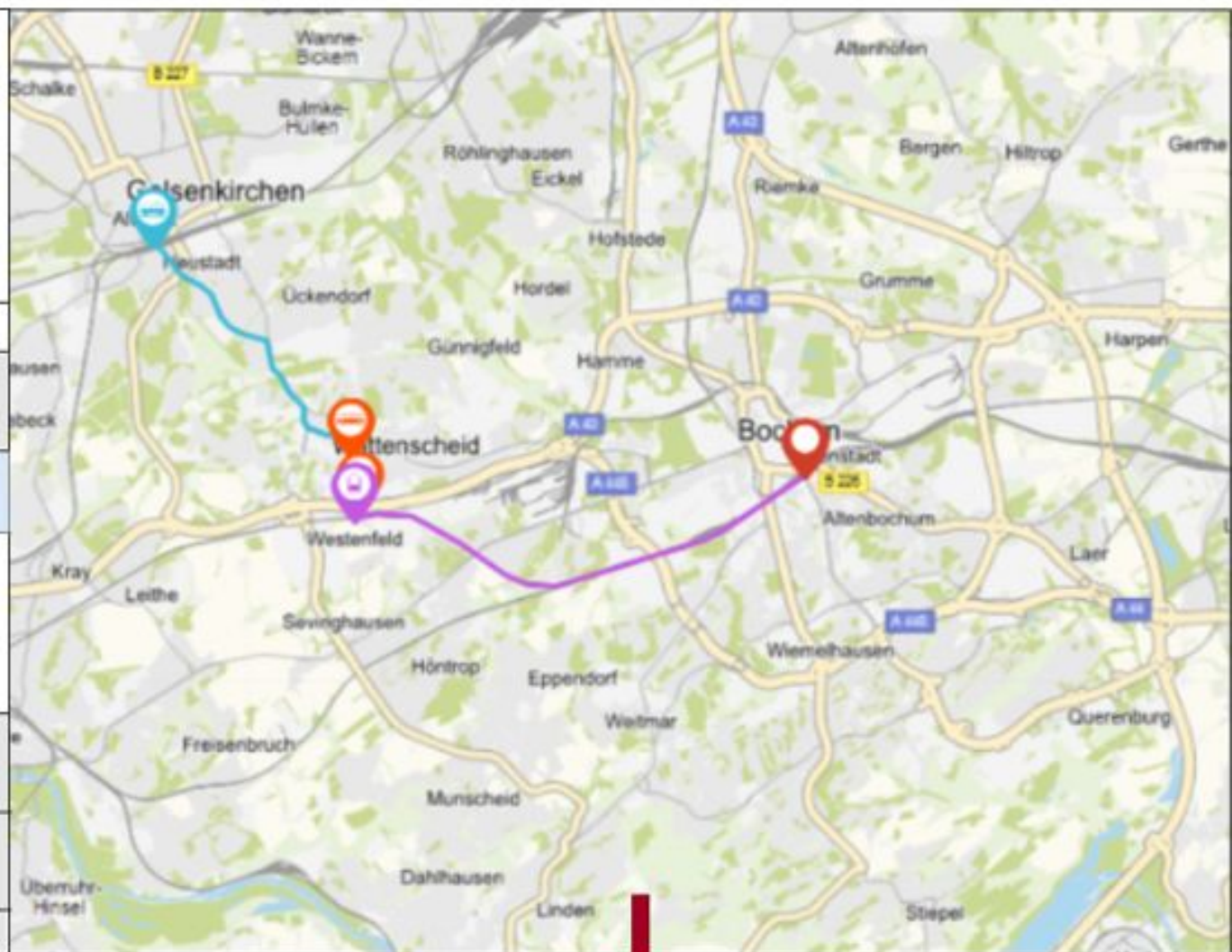
④ Passed Stops:  
TripStopTimes-Request

11:25 ● Central Station  
● First Stop  
RB ● Second Stop  
● Third Stop  
11:41 ● West Station

11:32 - 12:02  
U1 > S5

12:02 - 12:32  
U1 > S3

12:32 - 13:05



⑤ Line Sequence and Passed Stops:  
StopSeqCoord-Request

# Advanced Journey Planner

1 Journey Maps:  
MapRoute-Request

Origin Street 5

Destination 3

**Suchen**

**Journey Details**

8:40	Origin Street 5	
	Walk	
8:43	East Station	
8:43	East Station	
	U U5 Somewhere	
9:02	West Station	
	More time to change	
9:11	West Station	
	S S1 Nowhere	
9:25	South Station	
9:25	South Station	
	Walk	
9:37	Destination 3	

**Alternative Journeys**

from East Station to West Station

8:53	9:12
9:03	9:22
9:13	9:32
9:23	9:42
9:33	9:52
9:43	10:02

**More Time to Change**

- arrive earlier
- depart later

2 Leg Alternatives:  
LegTT-Request

3 Reschedule Legs:  
MoveTrip-Request






# Departure Board

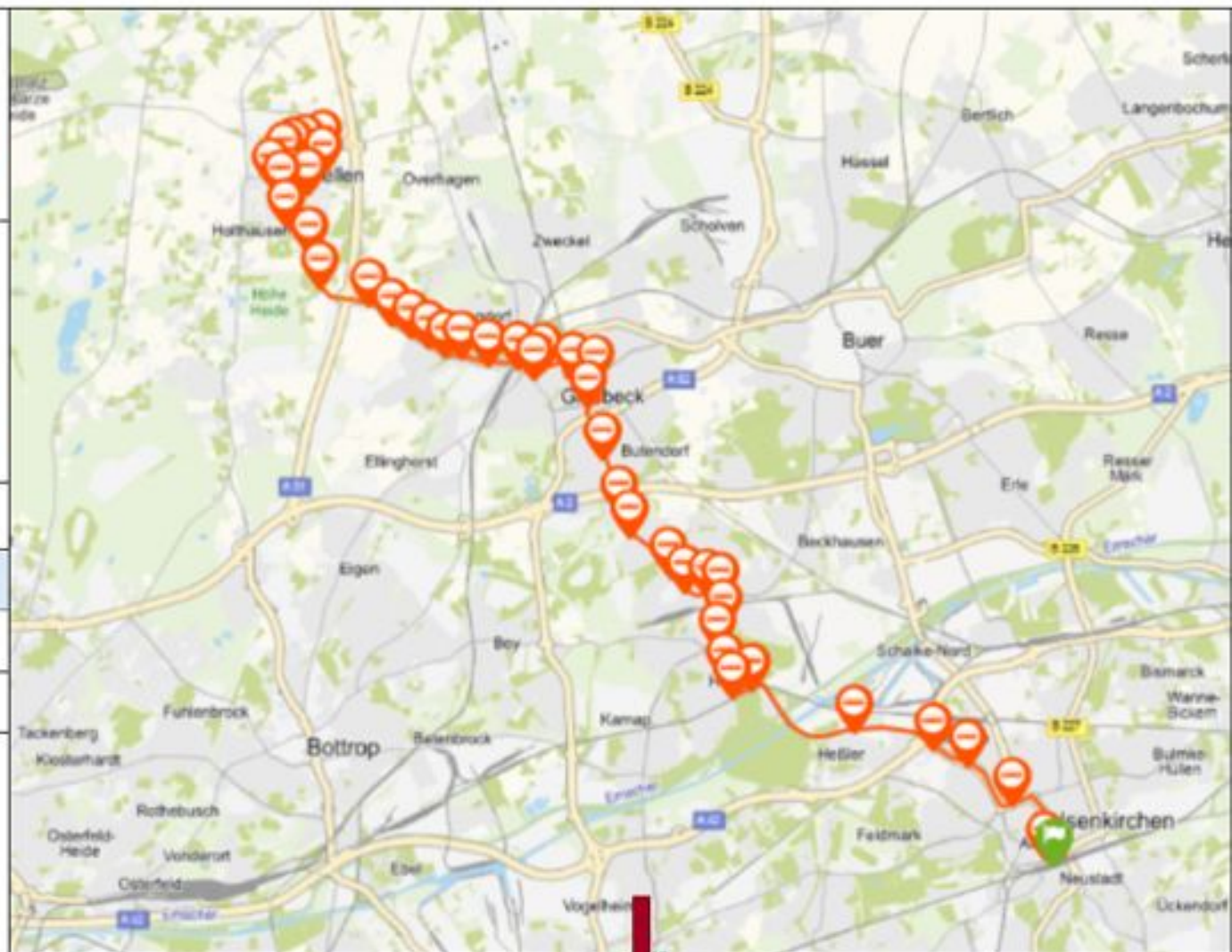
1 Point Search:  
StopFinder-Request

2 Line Search:  
ServingLines-Request

3 Departure Board:  
DM-Request

4 Passed Stops:  
TripStopTimes-Request

Central Station
<b>Submit</b>
S1 towards East Station U5 towards West Station RB 351 towards North
<b>Select</b>
11:02 U5 East Station
11:05 S1 West Station 
11:28 RB 351 North
11:35 S1 West Station
11:42 U5 East Station
<ul style="list-style-type: none"><li> First Stop</li><li> Second Stop</li><li> Third Stop</li><li> Fourth Stop</li></ul>



5 Line Sequence and Passed Stops:  
StopSeqCoord-Request

# Introduction – Use Cases Print Products

- JSON output includes base64 encoded stream
- Refer to a stop, a line or a passed stop of a line

## Starting Point: Stop Search

- Stop search with the StopFinder-Request
- If a serving line required: use ServingLines-Request and `mode=odv` to request serving lines of the previously identified stop

## Starting Point: Line Search

- Line search with the ServingLines-Request and `mode=line`
- If a passed stop is required: use LineStop-Request to request passed stops of the previously identified line

The image shows a web form with two input fields and a submit button. The first input field contains the text "station / stop / address / point of interest" and has a location pin icon on the right. Below it is the word "Or" and a second input field containing the text "line". A blue "Submit" button is positioned below the second input field. Below the form is a blue rectangular box with the text "STOP TIMETABLE" at the top and a white button labeled "Line and stop search" inside.

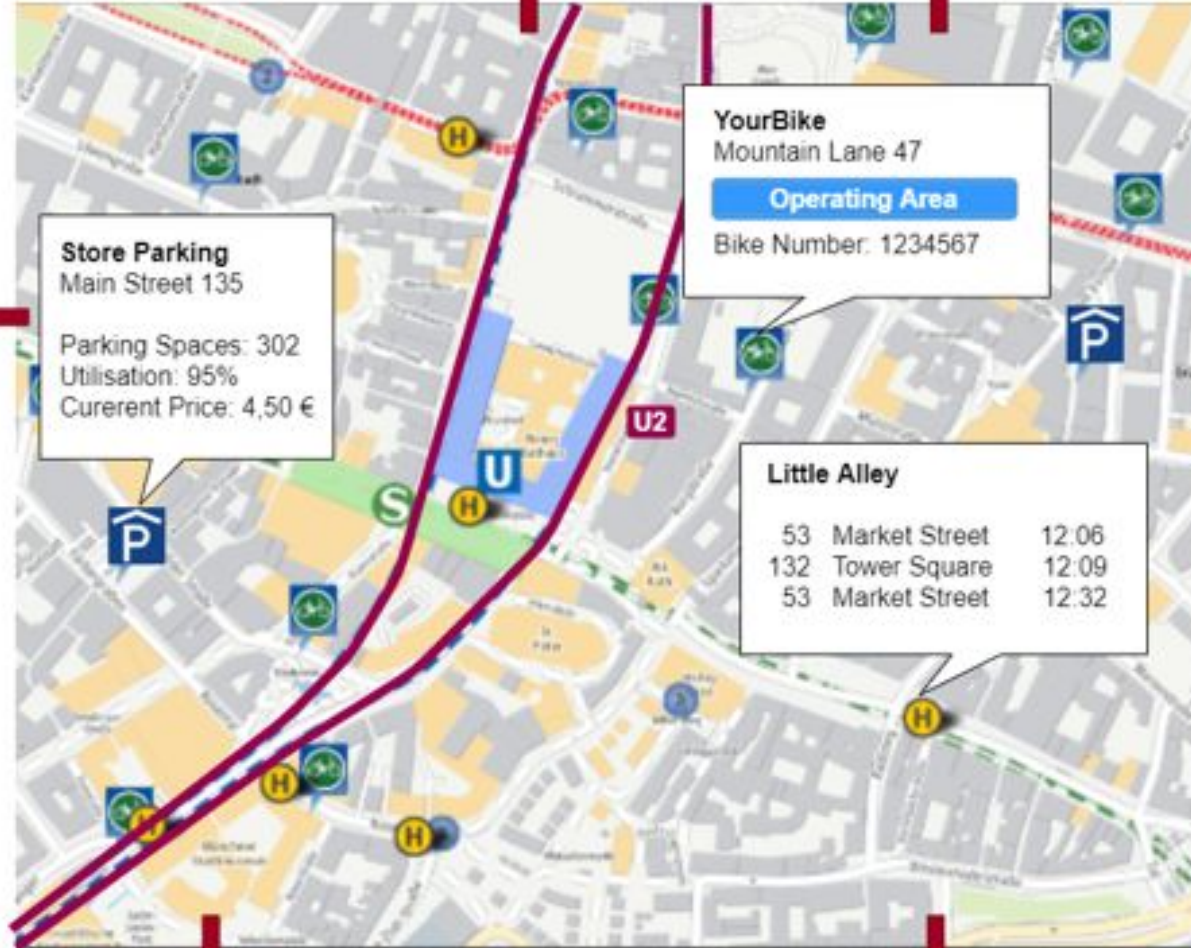
# Interactive Map

3 Lines:  
GeoObject-Request

5 Sharer Details:  
ModeShare-Request

6 Operating Areas:  
OpArea-Request

7 Park Object Details:  
ParkObject-Request



**Map View**

**Route Network**

- S1
- S2
- U1
- U2
- U3

Stops

Fine Arts

Restaurants

Sights

**Bike-Sharing**

- MyBike
- YourBike

**Parking**

- Car Park
- P&R

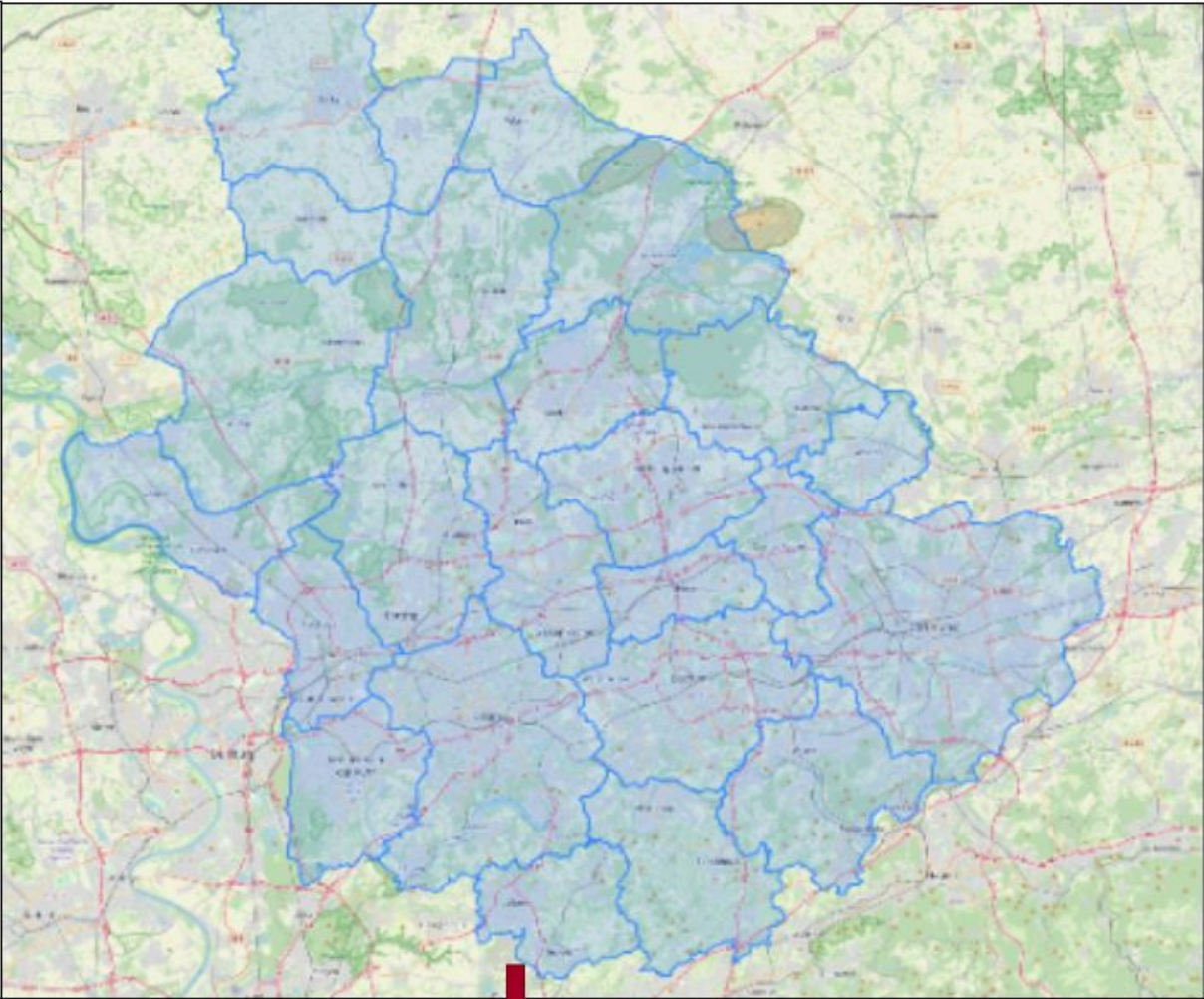
1 Map Objects (Stops, POI (with hierarchy), Sharer, Park Objects):  
CoordInfo-Request

4 Departure Board:  
DM-Request

2 Sharing Operator:  
OpArea-Request

# Ticket-Shop

Einzelticket Erwachsene Preisstufe: B Dauer: 2 h Preis: 3,50 €
Preisstufe B
Starthaltestelle Hauptbahnhof
Geltungsbereich 18072
Gültigkeitsbeginn 04.08.2022 10:30
Gültigkeitsende 04.08.2022 12:30
<b>Kaufen</b>



- 1 Punktsuche:  
StopFinder-Request
- 2 Tarifliche Information:  
StopZone-Request
- 3 Ticketdetails:  
TicketDetails-Request

4 Tarifgebiete und -zonen:  
Tariff-Scope-Request

# Introduction – Input and Output Request

A request is structured as follows:

```
http://server:port/virt_dir/request?HTTP_parameters
```

## Example (Trip-Request)

```
http://osm.demo.mentz.net/training/XML_TRIP_REQUEST2?commonMacro=trip
```

Requests are GET requests unless otherwise noted.

# Introduction – Input and Output Configuration of the Training System

The following HTTP parameters are set automatically for every request via configuration or parameter injection:

- `outputFormat=rapidJSON` (activates the JSON API)
- `coordOutputFormat=WGS84 [dd.ddddd]` (coord format set to WGS 84)
- `locationServerActive=1` (activates EFALocationServer for locality search)

Note: Parameter injection works only for requests with HTTP parameters

HTTP parameter macros are HTTP parameters which combine several HTTP parameters. They are defined in the EFA configuration.

## Advantages:

- Shorter URLs
- Same set of standard HTTP parameters for each request
- (Standard) parameters can be changed without changing the user interface
- Send more than one HTTP parameter by a HTML input element (e.g. checkbox, drop down list)

---

## Analyze the Request Parameters

The following tools are useful to analyze the request parameters. Use this for debugging or to have a closer look at the demo Journey Planner:

`https://efademo.mentz.net/sl3+/trip`

Fiddler (freeware):

- Web debugging proxy, which is logging the HTTP(S) traffic
- <https://www.telerik.com/fiddler>

Browser Developer Tools:

- Analyze HTML/CSS
- JavaScript debugger
- Analysis of performance, headers, requests,...

---

## Analyze the JSON Response

The following tools are useful to analyze the JSON response

JSONView (addon for Chrome/Firefox):

- Formatting of JSON

# Common Functionality

# Common Functionality – Table of Contents



1. Error Handling
2. Date and Time

---

Error messages are handled by the array `systemMessages`.

- `code` is not unique!
- `error` provides a description of the error
- `type` can be message or error
- `module` indicates on which EFA module the error occurred

Refer to document *EFA9-10\_Errorcodes\_V1.1\_en.docx*.

```
{
  version: "10.2.8.6",
  - systemMessages: [
    - {
      code: -8020,
      error: "origin: no matches",
      type: "error",
      module: "BROKER"
    },
    - {
      code: -8030,
      error: "destination: no input value",
      type: "error",
      module: "BROKER"
    }
  ]
}
```

```
systemMessages: [
  - {
    type: "warning",
    module: "itp-monomodal",
    code: -10015,
    text: "itp"
  }
]
```

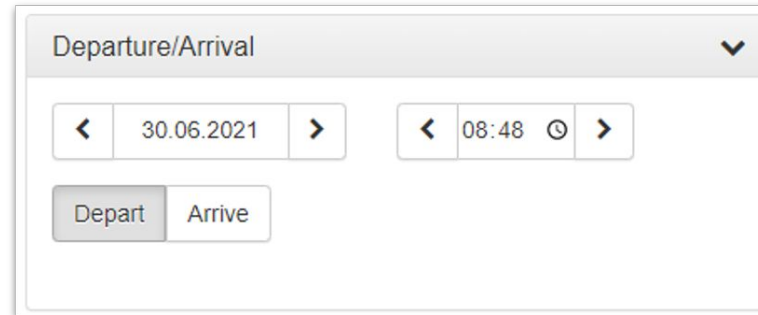
# Common Functionality – Date and Time Input and JSON Output

## Input

- The input of date and time is optional. If not requested differently the current date and time of the server are requested.
- The input date and time corresponds to the server date and time.

## JSON Output

- The output corresponds to UTC format (ISO 8601)
- One date and time refers to the scheduled (planned) time, the other date and time can contain realtime information (estimated).
- From EFA 10.5 on distinction between times of original timetable (baseTimetable) and times with daily updates to the timetable are available



```
version: "10.2.8.6",
systemMessages: [ ],
- journeys: [
  - {
    rating: 0,
    isAdditional: true,
    interchanges: 0,
    - legs: [
      - {
        duration: 360,
        - origin: {
          id: "10000566",
          name: "Belfast City Centre, Europa Buscentre",
          type: "stop",
          + coord: [...],
          + parent: {...},
          + departureTimePlanned: "2018-04-02T12:54:00Z",
          + departureTimeEstimated: "2018-04-02T12:54:00Z",
          + properties: {...}
        },

```

# Common Functionality – Date and Time Parameters to choose a Date

Parameter	Description	Format
itdDate	year, month and day	YYYYMMDD   JJMMDD
itdDateDay	day	DD   D
itdDateMonth	month	MM   M
itdDateYear	year	YYYY   YY
itdDateYearMonth	year and month	YYYYMM
itdDateDayMonthYear	day, month and year	DDMMYYYY   DDMMYY   DDxMMxYYYY*

\* x stands for any separator

## Examples for the Use of different Date Parameters

```
http://osm.demo.mentz.net/training/XML_TRIP_REQUEST2
?ext_macro=trip&type_origin=any&name_origin=10000566
&type_destination=1&name_destination=10000011 &itdDate=20210402
```

```
http://osm.demo.mentz.net/training/XML_TRIP_REQUEST2
?ext_macro=trip&type_origin=any&name_origin=10000566
&type_destination=1&name_destination=10000011 &itdDateDay=2&itdDateMonth=4&itdDateYear=2021
```

# Common Functionality – Date and Time Parameters to choose a Time

Parameter	Description	Format
itdTime	hour and minute	HHMM   HH:MM   HH.MM   HHMMa*   HHMMh**   HHMMp*
itdTimeHour	hour	HH   H
itdTimeMinute	minute	MM  M
timeOffset	offset to the current time (in minutes)	MM  M
itdTimeAMPM	Time is am or pm*	am   pm

\* Anglo-American format: „am“ and „pm“

\*\* 24-hour-Format

## Examples for the Use of different Time Parameters

```
http://osm.demo.mentz.net/training/XML_TRIP_REQUEST2?commonMacro=trip&type_origin=any&name_origin=5000350&type_destination=1&name_destination=5006052 &itdTime=1654
```

```
http://osm.demo.mentz.net/training/XML_TRIP_REQUEST2?commonMacro=trip&type_origin=any&name_origin=5000350&type_destination=1&name_destination=5006052 &itdTimeHour=17&itdTimeMinute=30
```

SystemInfo-Request

# SystemInfo-Request – Input and Output

---

Request to get information about the EFA system and validity information of the data.

## Request

`http://osm.demo.mentz.net/training/XML_SYSTEMINFO_REQUEST?commonMacro=system`

These parameters are given by parameter injection or configuration:

- `outputFormat=rapidJSON` (activates the JSON API)

Remember: Parameter injection works only for request with HTTP parameters.

```
{
  "version": "10.4.15.5",
  "ptKernel": {
    "appVersion": "10.4.17.7 build 01.09.2021 08:26:17",
    "dataFormat": "EFA10_04_00",
    "dataBuild": "2021-09-15T06:05:52Z"
  },
  "validity": {
    "from": "2021-08-01",
    "to": "2022-02-28"
  }
}
```

StopFinder-Request

# StopFinder-Request – Table of Contents

1. Input and Output
2. Locality Search
3. Locality Input
4. Nearby Stops
5. Default Texts
6. Optional Parameters

# StopFinder-Request – Input and Output Request

The EFALocationServer is the responsible module for the locality search. The StopFinder-Request is used to search a locality and get its unique ID. All other requests require a unique ID or coordinate as locality input.

## Request

```
http://osm.demo.mentz.net/training/XML_STOPFINDER_REQUEST?commonMacro=stopfinder
```

## Part of the StopFinder-Request

- Error Handling (as described per Common Functionality)
- Date and Time (Stops can be removed or added. Thus it's a good idea if locality search has the same date as journey planning.)

# StopFinder-Request – Input and Output JSON Output

## Example

[http://osm.demo.mentz.net/training/XML\\_STOPFINDER\\_REQUEST?commonMacro=stopfinder&type\\_sf=any&name\\_sf=stuttgart staatsgalerie](http://osm.demo.mentz.net/training/XML_STOPFINDER_REQUEST?commonMacro=stopfinder&type_sf=any&name_sf=stuttgart%20staatsgalerie)

## JSON Output

`locations` is an array of localities. The locality is specified in more detail by:

- `id` (unique id)
- `name` / `disassembledName`
- `coord` (coordinate)
- `type` (value: stop, poi, address, street, locality)
- `productClasses` – array of modes of transport which pass this stop
- `parent` – information about the locality or (in case of a stop point) the stop and locality
- `properties` (additional information)

Additionally in StopFinder-Request:

- `matchQuality` (quality)
- `isBest` (true for best match)

```
locations: [
  - {
    id: "de:08111:6024",
    isGlobalId: true,
    name: "Stuttgart, Staatsgalerie",
    disassembledName: "Staatsgalerie",
    - coord: [
      48.78275,
      9.18737,
    ],
    type: "stop",
    matchQuality: 1000,
    isBest: true,
    - productClasses: [
      3,
      5,
    ],
    - parent: {
      id: "placeID:8111000:51",
      name: "Stuttgart",
      type: "locality",
    },
    + assignedStops: [1],
    + properties: {2},
  },
],
```

# StopFinder-Request – Locality Search

## Mandatory Parameters

These parameters are given by parameter injection or configuration:

- `outputFormat=rapidJSON` (activates the JSON API)
- `coordOutputFormat=WGS84 [dd.ddddd]` (coord format set to WGS 84)
- `locationServerActive=1` (activates EFALocationServer for locality search)

Note: Parameter injection works only for requests with HTTP parameters.

Further customer specific parameters could be included in an HTTP parameter macro **`commonMacro=stopfinder`**.

# StopFinder-Request – Locality Search

## Mandatory Parameters

### **name\_<usage>**

Search string/name of the locality (e.g. stop, POI, address) or coordinate (e.g. via click on the interactive map).

### **type\_<usage> = any | coord**

Tighter specification of the locality. For EFALocationServer the value is always `any`. For coordinate input the value is `coord`.

### Parameter Suffix for Locality Input

The parameter suffix `<usage>` for StopFinder-Request is `sf`. Thus parameters are named `name_sf` and `type_sf`.

# StopFinder-Request – Locality Search

## Excursus: Parameter Suffix for Locality Input

Some requests require more than one locality, e.g. journey planning requires an origin and a destination. To distinguish the parameters, they have a suffix `<usage>`. The suffix differs from request to request. Some examples:

- `origin` (Trip-Request, PS-Request)
- `Destination` (Trip-Request, PS-Request)
- `via` (Trip-Request)
- `dm` (DepartureMonitor-Request)
- ...

# StopFinder-Request – Locality Search

## Search Criteria/Filters

Find the right search criteria/filters is a design task. It should be included in an HTTP parameter macro **commonMacro=stopfinder** in configuration. It could include:

- `anyMaxSizeHitList=30` (maximum size of hit list, criterion: match quality)
- `anySigWhenPerfectNoOtherMatches=1` (no other result for perfect matches)
- Other search criteria defined by customer (e.g. region filter, preference of regions, preference of stops served by certain modes of transports)

# StopFinder-Request – Locality Search Selection from a List

## Challenge

Search the stop *Stuttgart Fernsehturm*. Use the StopFinder-Request XML\_STOPFINDER\_REQUEST. Remember: The parameter suffix `<usage>` is `sf`.

# StopFinder-Request – Locality Search Selection from a List

## Solution

[http://osm.demo.mentz.net/training/XML\\_STOPFINDER\\_REQUEST?commonMacro=stopfinder&type\\_sf=any&name\\_sf=stuttgart fernsehturm](http://osm.demo.mentz.net/training/XML_STOPFINDER_REQUEST?commonMacro=stopfinder&type_sf=any&name_sf=stuttgart%20fernsehturm)

## The result is a list!

- If a locality search finds more than one hit for the input, the array `locations` contains more than one element.
- The best matching hit is marked by the element `isBest` with value `true`.
- Use the unique ID `id` or (in case of `isGlobalId=true`) `properties/stopID` for locality input. Or `id` if global IDs are required.

## Challenge

Select the best matching hit.

```
locations: [
  - {
    id: "de:08111:2564",
    isGlobalId: true,
    name: "Stuttgart, Fernsehturm",
    disassembledName: "Fernsehturm",
    - coord: [
      48.75627,
      9.18836,
    ],
    type: "stop",
    matchQuality: 1000,
    isBest: true,
    - productClasses: [
      5
    ],
    - parent: {
      id: "placeID:8111000:51",
      name: "Stuttgart",
      type: "locality",
    },
    - properties: {
      stopID: "5002564"
    },
  },
  - {
    id: "de:08111:6128",
    isGlobalId: true,
    name: "Stuttgart, Ruhbank (Fernsehturm)",
    disassembledName: "Ruhbank (Fernsehturm)",
    - coord: [
      48.75334,
```

# StopFinder-Request – Locality Search Selection from a List

## Solution

[http://osm.demo.mentz.net/training/XML\\_STOPFINDER\\_REQUEST?commonMacro=stopfinder&type\\_sf=any&name\\_sf=5002564](http://osm.demo.mentz.net/training/XML_STOPFINDER_REQUEST?commonMacro=stopfinder&type_sf=any&name_sf=5002564)

```
locations: [
  - {
    id: "de:08111:2564",
    isGlobalId: true,
    name: "Stuttgart, Fernsehturm",
    disassembledName: "Fernsehturm",
    - coord: [
      48.75627,
      9.18836,
    ],
    type: "stop",
    matchQuality: 1000,
    isBest: true,
    - productClasses: [
      5
    ],
    - parent: {
      id: "placeID:8111000:51",
      name: "Stuttgart",
      type: "locality",
    },
    - properties: {
      stopId: "5002564"
    },
  },
  - {
    id: "de:08111:6128",
    isGlobalId: true,
    name: "Stuttgart, Ruhbank (Fernsehturm)",
    disassembledName: "Ruhbank (Fernsehturm)",
    - coord: [
      48.75627,
      9.18836,
    ],
  },
]
```

# StopFinder-Request – Locality Search

## Sort Order of the List

---

If the list of hits is presented to the user for selection, it should be sorted. Best way is to do it in configuration.

**anyResSort\_<usage> = <Name>**

EFALocationServer can sort the results. Sort order is defined in configuration. This parameter chooses the sorter.

Example parameter: `anyResSort_sf=solingen`

```
[ResultSorter11]
Name             solingen
Criterion1      REGION
# Criterion2     OBJECTTYPE
# Criterion3     QUALITY
# Criterion4     ALPHABETICAL
ObjectTypeOrder STOP,DIVASINGLEHOUSE,POINAME,PLACEINT,DIVAADDR,DIVASTREET
RegionOrder     "32"
```

# StopFinder-Request – Locality Search Filters (Examples)

## `anyObjFilter_<usage>`

- The locality search may be limited to certain types of objects using this filter parameter.
- The value of the parameter is a bit mask.
- The individual object types can be combined.

Example: If the search space for the start point should be limited to bus stops and points of interests (2 + 32), the filter should be set to `anyObjFilter_origin=34`.

## Challenge

Search for stops *Stuttgart Bad Cannstatt* using the StopFinder-Request.

Value	Description
0	complete search area
1	locations
2	stop IDs and alias names of stops
4	streets
8	addresses
16	Intersections
32	POIs IDs and alias names of POIs
64	post codes

# StopFinder-Request – Locality Search Filters (Examples)

## Solution

```
http://osm.demo.mentz.net/training/XML_STOPFINDER_REQUEST?commonMacro=stopfinder&type_sf=any&name_sf=stuttgart bad  
cannstatt&anyObjFilter_sf=2
```

# StopFinder-Request – Locality Input Unique ID

The unique ID determined by StopFinder-Request may be used as an input for any request.

Use the following parameters:

- `name_<usage>`
- `type_<usage> = any`

Alternative: coordinates.

# StopFinder-Request – Locality Input

## Coordinate Input

- A coordinate is entered by the parameter `type_usage=coord` and `name_usage=<x>:<y>:<coordinate format>:<free text>`.
- The value of `name_usage` is composed of three required values and an optional value separated by colon. `<x>` and `<y>` are the x- and the y-coordinate.
- `<coordinate format>` describes the coordinate format. Pre-defined default format for the training server is `WGS84[dd.ddddd]`.
- The last value `<free text>` is optional. If no free text is available the system tries to snap to the nearest street.

### Examples

```
http://osm.demo.mentz.net/training/XML_STOPFINDER_REQUEST?commonMacro=stopfinder&type_sf=coord& name_sf=9.23:48.80:WGS84[dd.ddddd]
```

```
http://osm.demo.mentz.net/training/XML_STOPFINDER_REQUEST?commonMacro=stopfinder&type_sf=coord& name_sf=9.23:48.80:WGS84[dd.ddddd]:A nice place
```

# StopFinder-Request – Nearby Stops

- Some requests require a uniquely identified stop, e.g. timetable.
- In the case of addresses or POIs an additional step is necessary: the selection of a nearby stop (e.g. the one with the shortest distance).
- More complex stops or stations, for example a central station, can be modelled by several stops.
- The array of stops is called `assignedStops`. Its default maximum length is 10.
- To choose a nearby stop the unique ID can be used.

## Example

[http://osm.demo.mentz.net/training/XML\\_STOPFINDER\\_REQUEST?commonMacro=stopfinder&type\\_sf=coord&name\\_sf=9.23:48.80:WGS84\[dd.ddddd\]](http://osm.demo.mentz.net/training/XML_STOPFINDER_REQUEST?commonMacro=stopfinder&type_sf=coord&name_sf=9.23:48.80:WGS84[dd.ddddd])

```
locations: [
- {
  id: "coord:1027489:5759044:MRCV:Bad Cannstatt, Gasteiner Straße 15:0",
  name: "Bad Cannstatt, Gasteiner Straße 15",
  disassembledName: "Gasteiner Straße 15",
+ coord: [2],
  buildingNumber: "15",
  type: "address",
+ parent: {3},
- assignedStops: [
+ {12},
+ {12},
+ {12},
+ {12},
- {
  id: "de:08111:33",
  isGlobalId: true,
  name: "Stuttgart Augsburger Platz",
  disassembledName: "Augsburger Platz",
  type: "stop",
- coord: [
    48.80569,
    9.23035,
  ],
- parent: {
  name: "Stuttgart",
  type: "locality",
},
  distance: 911,
  duration: 13,
- productClasses: [
    3,
    5,
  ],
  connectingMode: 100,
- properties: {
  stopId: "5000033"
},
},
},
].
```

# StopFinder-Request – Nearby Stops

---

## Suppress the Search for Nearby Stops

StopFinder-Request searches by default for (nearby) stops. In some cases nearby stops are not required, e.g. for the autosuggest list.

Suppress the search to improve performance!

**doNotSearchForStops\_<usage> = 1**

Prevents the search for nearby stops. It should be used to increase the performance whenever the routing should not consider public transport.

## Example

[http://osm.demo.mentz.net/training/XML\\_STOPFINDER\\_REQUEST?commonMacro=stopfinder&type\\_sf=coord&name\\_sf=9.23:48.80:WGS84\[dd.ddddd\]&doNotSearchForStops\\_sf=1](http://osm.demo.mentz.net/training/XML_STOPFINDER_REQUEST?commonMacro=stopfinder&type_sf=coord&name_sf=9.23:48.80:WGS84[dd.ddddd]&doNotSearchForStops_sf=1)

```
locations: [  
  - {  
    id: "coord:1027489:5759044:MRCV:Bad Cannstatt, Gasteiner Straße 15:0",  
    name: "Bad Cannstatt, Gasteiner Straße 15",  
    disassembledName: "Gasteiner Straße 15",  
    - coord: [  
      48.79978,  
      9.23009,  
    ],  
    buildingNumber: "15",  
    type: "address",  
    - parent: {  
      id: "placeID:8111000:1500000003",  
      name: "Bad Cannstatt",  
      type: "locality",  
    },  
  },  
],
```

# StopFinder-Request – Default Text

---

Do not send any “default text” to the EFA system. EFALocationServer gives its best to propose localities. For a default text this does not make any sense and the results will confuse the user. If no program driven is possible use this parameter:

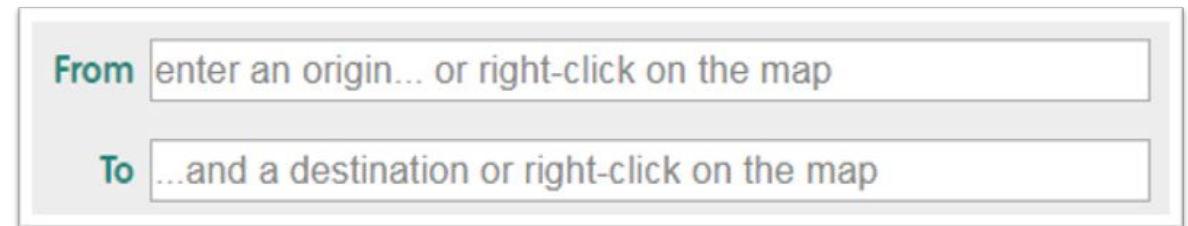
## **nameDefaultText\_<usage>**

The value of this parameter is a text which is not considered for the locality search of the input given in `name_<usage>`.

### Example

```
http://osm.demo.mentz.net/training/XML_STOPFINDER_REQUEST?commonMacro=stopfinder&type_sf=any&name_sf=enter an origin... or right-click on the map
```

```
http://osm.demo.mentz.net/training/XML_STOPFINDER_REQUEST?commonMacro=stopfinder&type_sf=any&name_sf=enter an origin... or right-click on the map&nameDefaultText_sf=enter an origin... or right-click on the map
```



From

To

# Trip-Request

# Trip-Request – Table of Contents



1. Input and Output
2. Extension of Date and Time
3. Connection Options
4. Realtime

# Trip-Request – Input and Output Request

The Trip-Request calculates journeys to a given origin and destination. Date, time, via locality and travel options are optional. The output includes travel options, optionally with realtime information.

## Request

```
http://osm.demo.mentz.net/training/XML_TRIP_REQUEST2?commonMacro=trip
```

## Part of the Trip-Request

- Error Handling (as described per Common Functionality)
- Date and Time (as described per Common Functionality)
- Locality Input (as described per StopFinder-Request)

# TripRequest – Input and Output

## Mandatory Parameters

---

### Parameters for the Interface

These parameters are given by parameter injection or configuration:

- `outputFormat=rapidJSON` (activates the JSON API)
- `coordOutputFormat=WGS84 [dd.ddddd]` (coord format set to WGS 84)
- `locationServerActive=1` (activates EFALocationServer for locality search)

Note: Parameter injection works only for requests with HTTP parameters.

### Mandatory Parameters for Trip-Request

The Trip-Request requires an origin and destination input as described per StopFinder-Request.

These parameters should be included in the HTTP parameter macro

**commonMacro=trip:**

- `deleteAssignedStops_origin/deleteAssignedStops_destination` (no nearby stops)
- `genC=0, genP=0, genMaps=0` (prevents output of coordinate sequences, path descriptions and generation of additional pdf files)

### Optional customer specific parameters:

Can be added to **commonMacro=trip**, e.g.:

- `useUT=1` (enables unified tickets)
- `useRealtime=1` (enables realtime)

# Trip-Request – Input and Output

## Parameter Suffix for Locality Input

### Parameter Suffix for Locality Input

The parameter suffixes `<usage>` are `origin`, `destination` and `via`. The `via` location is optional.

### Challenge

Calculate a journey from the stop *Stuttgart Schwabstraße* to the stop *Stuttgart Feuersee*.

Hint: Trip calculation takes only place if the origin and destination are identified.

# Trip-Request – Input and Output

## Parameter Suffix for Locality Input

### Solution

**Step 1: Determine unique IDs for origin and destination, e.g. by StopFinder-Request:**

```
http://osm.demo.mentz.net/training/XML_STOPFINDER_REQUEST?commonMacro=stopfinder&type_sf=any&name_sf=stuttgart schwabstraße
```

**Step 2: Trip calculation with the Trip-Request:**

```
http://osm.demo.mentz.net/training/XML_TRIP_REQUEST2?commonMacro=trip&type_origin=any&name_origin=5006052&type_destination=any&name_destination=5006221
```

# Trip-Request – Input and Output

## JSON Output

### Journeys

`journeys` is an array of journey options. Each journey option contains information about :

- the number of `interchanges`
- one or more `legs`

And optionally about the :

- `fares`
- and `daysOfService`.

```
journeys: [
  - {
    rating: 0,
    isAdditional: false,
    interchanges: 0,
    - legs: [
      + {6}
    ],
    + fare: {1},
    + daysOfService: {1},
  },
  + {6},
  + {6},
  + {5},
],
```



# Trip-Request – Input and Output

## JSON Output

---

### Leg

A leg includes:

- duration
- origin and destination localities
- transportation (mode of transport)
- optionally the distance and customer specific properties may be included
- optionally `footPathInfo` (interchange footpath, e.g. stairs, elevators)

Public transport legs additionally contain:

- `stopSequence` (sequence of passed stops)

The passed stops have the structure of `location` as described for `StopFinder-Request`.

```
legs: [  
  - {  
    duration: 120,  
    + origin: {13},  
    + destination: {13},  
    + transportation: {9},  
    + stopSequence: [2],  
    + infos: [1],  
  }  
],
```

# Trip-Request – Input and Output

## JSON Output

And optionally:

- `infos` (status updates)
- `hints` (additional information)
- `isRealtimeControlled` (information if the vehicle is realtime controlled)

Unreduced response includes additionally:

- `coords` (array of path coordinates)
- For individual transport turn instructions (`pathDescription`) are available.

# Trip-Request – Input and Output

## JSON Output

---

### Location

The location (e.g. origin, destination) is specified as described for StopFinder-Request.

Additionally:

- departureTimePlanned/  
arrivalTimePlanned (scheduled departure/arrival time)
- departureTimeEstimated/  
arrivalTimeEstimated (realtime information)

```
origin: {
  isGlobalId: true,
  id: "de:08111:32:2:1",
  name: "Stuttgart Uff-Kirchhof",
  disassembledName: "Uff-Kirchhof",
  type: "platform",
+ coord: [2],
  niveau: 1,
+ parent: {9},
+ productClasses: [2],
  departureTimePlanned: "2021-11-04T12:37:00Z",
  departureTimeEstimated: "2021-11-04T12:37:00Z",
- properties: {
  WheelchairAccess: "true",
  AREA_NIVEAU_DIVA: "1",
  areaGid: "de:08111:32:2",
  area: "2",
  platform: "1",
},
}.
```

# Trip-Request – Input and Output

## JSON Output

### Transportation

transportation includes information about the mode of transport (public or individual).

Public transport types contain:

- id (unique ID)
- Name/disassembledName
- number
- description
- operator
- destination
- properties (includes information about further products available (TTB, STT, ROP))

```
transportation: {
  id: "vvs:10001: :R:j21",
  name: "S-Bahn S1",
  disassembledName: "S1",
  number: "S1",
  description: "Herrenberg - Stuttgart - Plochingen - Kirchheim (T)",
  - product: {
    id: 0,
    class: 1,
    name: "S-Bahn",
    iconId: 2,
  },
  + operator: {2},
  + destination: {3},
  + properties: {7},
},
```

# Trip-Request – Input and Output

## JSON Output

---

### Product

product is valid for both public and individual transport:

- class (mode of transport)
- name (description of mode of transport)
- iconId (unique identifier for the icon)

```
transportation: {
  id: "vvs:10001: :R:j21",
  name: "S-Bahn S1",
  disassembledName: "S1",
  number: "S1",
  description: "Herrenberg - Stuttgart - Plochingen - Kirchheim (T)",
  - product: {
    id: 0,
    class: 1,
    name: "S-Bahn",
    iconId: 2,
  },
  + operator: {2},
  + destination: {3},
  + properties: {7},
},
```

```
transportation: {
  - product: {
    class: 99,
    name: "footpath",
    iconId: 99
  }
},
```

# Trip-Request – Input and Output JSON Output

## Infos

infos is an array of ICS messages:

- priority – values: here always normal
- url, urlText – link and text for the link
- subtitle, content – title and content of the message

```
infos: [
  - {
    priority: "normal",
    id: "23782_Translink",
    version: "1",
    urlText: "Easter Holiday Information (some disruption to routes due to parades etc)>>>>",
    url: "http://jpincident.translink.co.uk:80/ics/XSLT_CM_SHOWADDINFO_REQUEST?infoID=23782_Translink&seqID=1",
    content: "<a href='http://www.translink.co.uk/Services/Metro-Service-Page/metro-travel-updates1/'>http://www.translink.co.uk/Services/Metro-Service-Page/metro-travel-updates1/</a>",
    subtitle: "Easter Holiday Information (some disruption to routes due to parades etc)>>>>",
    - properties: {
      providerCode: "Translink"
    }
  },
  + {...}
],
```

# Trip-Request – Input and Output

## JSON Output

---

### Hints

`hints` is an array of hints for the service.

```
hints: [  
  - {  
    content: "Service 511b: Bank holidays"  
  }  
],
```

# Trip-Request – Input and Output

## JSON Output

### Footpath Info

footPathInfo includes information about foot paths to reach a stop e.g. the duration, the position relative to the leg (BEFORE, AFTER, IDEST) and an array of descriptive elements footPathElem:

- type - STAIRS, RAMP, ESCALATOR, ELEVATOR, LEVEL
- level - LEVEL, UP, DOWN
- levelFrom and levelTo specifies the change of level in case of level=UP|DOWN

```
footPathInfo: [  
  - {  
    position: "IDEST",  
    duration: 360,  
    - footPathElem: [  
      - {  
        description: "",  
        type: "LEVEL",  
        levelFrom: 0,  
        levelTo: 0,  
        level: "LEVEL",  
        + origin: {...},  
        + destination: {...}  
      }  
    ]  
  }  
],
```

```
itdTripDateTimeDepArr = dep | arr
```

Determines whether the journey should depart (`dep`) or arrive (`arr`) at the indicated time.

Default: `dep`.

## Challenge

On April 28th you are currently near the stop *Stuttgart Schwabstraße* and want to meet your friends later at 17:00 at the stop *Stuttgart Feuersee*. When do you have to leave?

## Solution

**Step 1: Determine unique IDs for origin and destination, e.g. by StopFinder-Request:**

```
http://osm.demo.mentz.net/training/XML_STOPFINDER_REQUEST?commonMacro=stopfinder&type_sf=any&name_sf=stuttgart schwabstraße
```

**Step 2: Trip calculation with the Trip-Request:**

```
http://osm.demo.mentz.net/training/XML_TRIP_REQUEST2?commonMacro=trip&type_origin=any&name_origin=5006052&type_destination=any&name_destination=5006221&itdDateDay=28&itdDateMonth=4&itdTime=1700&itdTripDateTimeDepArr=arr
```

The basic parameters are perfect for the calculation of a trip, but for more complex requests, they are not always sufficient. Especially for people with special transport needs. Therefore, additional parameters offer further control. This makes it possible, for example, to calculate trips for people with reduced mobility, heavy baggage etc.

Different types of options:

- Common options
- Options valid for public transport
- Advanced mobility options for public transport
- Options valid for individual transport

# Trip-Request – Connection Options

## Common Options

### **calcNumberOfTrips**

Specifies the number of trips which are calculated.

*Value:* Integer

*Default:* 4 + walk only trip + alternative trips

Alternative trips are identified by `isAdditional=true`. They are recommended journey options which do not match the preferred search criteria, e.g. a fast journey with a couple of interchanges if you prefer fewer interchanges.

# Trip-Request – Connection Options Common Options

`calcOneDirection = 1`

Prevents EFA from calculating one journey before the requested departure.

The screenshot shows a mobile application interface for a trip request. At the top, there are radio buttons for 'Departure' (selected) and 'Arrival', with a 'NOW' button to the right. Below this, there are input fields for 'Pick time' and 'Pick date'. The 'Pick time' field shows '14:54' and is highlighted with a red box. The 'Pick date' field shows '05/11/2021'. Below the input fields is a blue bar with icons for refresh, print, and share. Below the blue bar is a table of connection options. The table has two columns: 'Earlier' and 'Later'. The first row in the 'Earlier' column is highlighted with a red box and shows a departure time of '14:50' and an arrival time of '15:13'. The second row shows a departure time of '14:57' and an arrival time of '15:13'. The third row shows a departure time of '14:57' and an arrival time of '15:14'. Each row also shows the travel mode (walking, bus, train) and the price level (M) and cost (From EUR 3.30).

Earlier	Later
14:50 15:13 Walking > Bus 100	23 Min. Price-level M From EUR 3.30
14:57 - 15:13 Walking > Train U4 > Walking > Bus 100	16 Min. Price-level M From EUR 3.30
14:57 - 15:14 Walking > Train U4 > Walking	17 Min. Price-level M From EUR 3.30

# Trip-Request – Connection Options

## Options valid for public Transport

To use these parameters, the options for public transport `ptOptionsActive=1` must be enabled.

### **useProxFootSearch=1**

Taking account of nearby stops. To differentiate between origin and destination stop, the parameter name can be supplemented using the extension `Origin` or `Destination`.

### **maxChanges**

Maximum number of changes in one trip. Trips with more than the specified changes will be discarded for the trip request.

Values: 0–9 (Default: 9)

# Trip-Request – Connection Options

## Options valid for public Transport

### **routeType**

Specifies the criterion according to which the trip request should be optimized:

Value/Criterion	Description
leastinterchange	Connections with least interchanges
<b>leasttime</b>	Fastest connections
leastwalking	Connections with least footpaths

# Trip-Request – Connection Options

## Options valid for public Transport: Transport Selection

### `exclMOT_<ID>`

- This parameter causes the means of transport with the identification number <ID> to be excluded.
- To exclude multiple transports the parameter can be used multiple times.
- This parameter does not require a value. The means of transport with the identification number <ID> is excluded if the corresponding parameter is passed.
- It is necessary activate this feature with the parameter `excludedMeans=checkbox`. All modes are included initially.
- Altering to parameter `exclMOT_<ID>` means of transports can be excluded with the parameter `excludedMeans=<ID>`.

Note: The means of transport and their IDs are customer specific. The table shows the standard assignment.

ID	Mode of Transport
0	train
1	commuter railway
2	underground train
3	city rail
4	tram
5	city bus
6	regional bus
7	coach
8	cable car
9	Boat
10	transit on demand

ID	Mode of Transport
11	other
12	airplane
13	regional train
14	national train
15	international train
16	high-speed train
17	rail replacement train
18	shuttle train
19	Bürgerbus

# Trip-Request – Connection Options

## Options valid for public Transport: Transport Selection

### `inclMOT_<ID>`

- This parameter causes the means of transport with the identification number <ID> to be included by the system.
- If several means of transport are taken into account, the parameter can be used multiple times.
- This parameter does not require a value. The means of transport with the identification number is <ID> included when the corresponding parameter is passed.
- By default, the means of transports, when using the transportation inclusion, are disabled and will not be considered.
- Analog to the exclusion of transports. It is necessary to activate this functionality with the parameter `includedMeans=checkbox`.
- Altering to this parameter means of transports can be included with the parameter `includedMeans=<ID>`.

Note: The means of transport and their IDs are customer specific. The table shows the standard assignment.

ID	Mode of Transport
0	train
1	commuter railway
2	underground train
3	city rail
4	tram
5	city bus
6	regional bus
7	coach
8	cable car
9	Boat
10	transit on demand

ID	Mode of Transport
11	other
12	airplane
13	regional train
14	national train
15	international train
16	high-speed train
17	rail replacement train
18	shuttle train
19	Bürgerbus

# Trip-Request – Connection Options

## Options valid for public Transport: Transport Selection

### Challenge

Calculate a trip from *Stuttgart Schwabstraße* to *Stuttgart Feuersee* without using the commuter railway (ID = 1).

### Hint:

```
http://osm.demo.mentz.net/training/XML_TRIP_REQUEST2?commonMacro=trip&type_origin=any&name_origin=5006052&type_destination=any&name_destination=5006221
```

# Trip-Request – Connection Options

## Options valid for public Transport: Transport Selection

### Solution

```
http://osm.demo.mentz.net/training/XML_TRIP_REQUEST2?commonMacro=trip&type_origin=any&name_origin=5006052&type_destination=any&name_destination=5006221&excludedMeans=1
```

`ptOptionsActive=1` is not necessarily required for exclusion/inclusion of means of transport.

# Trip-Request – Connection Options

## Options valid for public Transport: accessibility Options

---

Accessibility options require data about the stop structure. Do not use these parameter if this data is no available.

To use these parameters, the accessibility options `impairedOptionsActive=1` must be enabled.

Caution! The parameters activate the described functionality with any value. Even with `false`.

### **lowPlatformVhcl**

This parameter activates the exclusive use of low platform vehicles.

### **noElevators**

This parameter excludes elevators from journey planning.

### **noEscalators**

This parameter excludes escalators from journey planning.

### **noSolidStairs**

This parameter excludes stairs from journey planning.

---

### **wheelchair**

This parameter includes only vehicles with wheelchair access in journey planning. Vehicles are low platform or have either a lift for wheelchair or ramp.

# Trip-Request – Connection Options

## Options valid for individual Transport

To use these parameters, the options for individual transport `itOptionsActive=1` must be enabled.

### `trITMOT`

This parameter indicates the means of transport from the starting point to the first stop and from the destination stop to the destination point. The following values are possible:

Alternative: The means of transport can be set separately for origin and destination with the parameters `trITDepMOT` and `trITArrMOT`.

### `trITMOTvalue<ID>`

The value of the parameter indicates the maximum time from the starting point to the first stop and also from the destination stop to the destination point.

*Value:* The time is specified in minutes (Default: 10)

Alternative:

The times can be set separately for origin and destination with the parameters `trITDepMOTvalue<ID>` and `trITArrMOTvalue<ID>`.

ID	Mode
100	footpath
101	bike & ride
102	take your bike along
103	kiss & ride
104	park & ride
105	taxi

# Trip-Request – Connection Options

## Options valid for public Transport and individual Transport

### **changeSpeed**

- Sets the speed for interchange paths, when `ptOptionsActive=1`.
- Sets the speed for the path from the starting point to the departure stop and the speed for the path from the destination stop to the destination, if `itOptionsActive=1`.
- Values:

PT: `normal` (e.g. 100), `slow` (e.g. 50), `fast` (e.g. 200)

$\text{interchange time [min]} = (\text{time from interchange matrix [min]} * \text{parameter value}) / 100$

IT: `normal` (e.g. 100), `slow` (e.g. 200), `fast` (e.g. 50)

$\text{speed [km/h]} = (\text{default speed [km/h]} * \text{parameter value}) / 100$

# Trip-Request – Connection Options

Options valid for public Transport and individual Transport

## Challenge

Calculate a trip from *Stuttgart Rothenbergstraße 5* to *Stuttgart Schwabstraße 25*. You have your heavy luggage with you. Search for an suitable connection.

# Trip-Request – Connection Options

Options valid for public Transport and individual Transport

## Solution (Example):

### Step 1: locality search

```
http://osm.demo.mentz.net/training/XML_STOPFINDER_REQUEST?  
commonMacro=stopfinder&type_sf=any&name_sf=stuttgart  
rothenbergstraße 5
```

```
http://osm.demo.mentz.net/training/XML_STOPFINDER_REQUEST?commonMacro=stopfinder&type_sf=any&name_sf=stuttgart schwabstraße 25
```

# Trip-Request – Connection Options

## Options valid for public Transport and individual Transport

### Step 2: trip calculation with advanced options

```
http://osm.demo.mentz.net/training/XML_TRIP_REQUEST2?commonMacro=trip&type_origin=any&name_origin=streetID:1500000870:5:8111000:51:Rotenbergstra%C3%9Fe:Stuttgart:Rotenbergstra%C3%9Fe::Rotenbergstra%C3%9Fe:70190:ANY:DIVA_SINGLEHOUSE:1024014:5761338:MRCV:osm:0&type_destination=any&name_destination=streetID:1500000505:25:8111000:51:Schwabstra%C3%9Fe:Stuttgart:Schwabstra%C3%9Fe::Schwabstra%C3%9Fe:70197:ANY:DIVA_SINGLEHOUSE:1019541:5764173:MRCV:osm:0  
ptOptionsActive=1&itOptionsActive=1&changeSpeed=slow&routeType=LEAST  
INTERCHANGE
```

# Trip-Request – Realtime

---

## useRealtime=1

Activates the realtime output.

origin and destination of the delayed leg include not only the scheduled time (departureTimePlanned/arrivalTimePlanned) but also the estimated time (departureTimeEstimated/arrivalTimeEstimated).

```
origin: {
  isGlobalId: true,
  id: "700000001710",
  name: "Dundonald, Ulster Hospital",
  type: "platform",
  + coord: [...],
  + parent: {...},
  departureTimePlanned: "2018-04-08T09:32:00Z",
  departureTimeEstimated: "2018-04-08T09:38:00Z",
  + properties: {...}
},
destination: {
  isGlobalId: true,
  id: "700000001803",
  name: "Belfast City Centre, Donegall Square West",
  type: "platform",
  + coord: [...],
  + parent: {...},
  arrivalTimePlanned: "2018-04-08T09:49:00Z",
  arrivalTimeEstimated: "2018-04-08T09:59:00Z",
  + properties: {...}
},
```



DepartureMonitor-Request

# DepartureMonitor-Request – Table of Content



1. Input and Output
2. Design Variants
3. Optional Parameters
4. Realtime

# DepartureMonitor-Request – Input and Output Request

Next departures of a stop group, several nearby stop groups or a stop point.

## Request

`http://osm.demo.mentz.net/training/XML_DM_REQUEST?`

## Part of the DepartureMonitor-Request

- Error Handling (as described per Common Functionality)
- Date and Time (as described per Common Functionality)
- Locality Input (as described per StopFinder-Request)
- Transport Selection (as described per Trip-Request)

## Parameter Suffix for Locality Input

The parameter suffix `<usage>` is `dm`.

# DepartureMonitor-Request – Input and Output

## Mandatory Parameters

---

### Parameters for the Interface

These parameters are given by parameter injection or configuration:

- `outputFormat=rapidJSON` (activates the JSON API)
- `coordOutputFormat=WGS84 [dd.ddddd]` (coord format set to WGS 84)
- `locationServerActive=1` (activates EFALocationServer for locality search)

Note: Parameter injection works only for requests with HTTP parameters.

### Mandatory Parameters for DepartureMonitor-Request

The DM-Request requires an origin input as described per StopFinder-Request.

These parameters should be included in the HTTP parameter macro

**commonMacro=dm:**

- `mode=direct`
- `useAllStops=1` (all stop points, includes stop points which cannot be reached by a walk, e.g. some underground stop points)

- 
- `lsShowTrainsExplicit=1` (enables trains)
  - `useProxFootSearch=0` (no alternative stops)

Optionally customer specific Parameters can be included, e.g.:

- `useRealtime=1` (activates realtime)

# DepartureMonitor-Request – Input and Output Mandatory Parameters

## Challenge

Calculate the departure board for *Stuttgart Feuersee* at 11:15.

# DepartureMonitor-Request – Input and Output

## Mandatory Parameters

### Solution

```
http://osm.demo.mentz.net/training/XML_DM_REQUEST?commonMacro=dm&type_dm=any&name_dm=5006221&itdTime=1115
```

# DepartureMonitor-Request – Input and Output

## JSON Output

### Locations

`locations` contains the locality for the departure board. It includes a list of nearby stops (`assignedStops`) which may include one (for stop) or more (for addresses, POIs,..) stops.

### List of Departures

`stopEvents` is an array of departures. It includes:

- `location` – locality of departure
- `departureTimePlanned /Estimated`– departure time
- `transportation` – information about the mode of transport
- `infos` (optionally) - array of ICS messages

```
{
  version: "10.2.8.6",
  systemMessages: [ ],
+ locations: [...],
- stopEvents: [
  - {
    + location: {...},
      departureTimePlanned: "2018-04-12T10:15:00Z",
    + transportation: {...}
  },
+ {...},
+ {...},
+ {...},

```

# DepartureMonitor-Request - Design Variants

There are two options to display a departure board for POIs or addresses:

- A combined departure board which displays the departures of nearby stop groups
- Select a nearby stop (see StopFinder-Request) and display the departure board for this stop.

## Challenge

Get a combined departure board for *Stuttgart Schwabstraße 22*.

**DEPARTURES**

From München, Sonnenstraße 22

Departure Search options

Departure  Arrival NOW

**Sendlinger Tor** (303 m)

14:52	62 Rotkreuzplatz U towards Rotkreuzplatz U	🕒
14:53	U6 Klinikum Großhadern towards Klinikum Großhadern	🕒
14:54	U2 Messestadt Ost towards Messestadt Ost	🕒

more departure times

**Karlsplatz (Stachus)** (255 m)

14:52	U4 Arbellapark towards Arbellapark	🕒
14:52	17 Maxmonument towards Maxmonument	🕒
14:54	U2 Messestadt Ost towards Messestadt Ost	🕒

more departure times

**DEPARTURES**

From Sendlinger Tor, München

Departure Search options

Departure  Arrival NOW

**Sendlinger Tor**

14:52	62 Rotkreuzplatz U towards Rotkreuzplatz U	🕒
14:53	U6 Klinikum Großhadern towards Klinikum Großhadern	🕒
14:54	U2 Messestadt Ost towards Messestadt Ost	🕒

more departure times

# DepartureMonitor-Request - Design Variants

## Solution

### Step 1: Locality search with the StopFinder-Request

[http://osm.demo.mentz.net/training/XML\\_STOPFINDER\\_REQUEST?commonMacro=stopfinder&type\\_sf=any&name\\_sf=stuttgart schwabstraße 22](http://osm.demo.mentz.net/training/XML_STOPFINDER_REQUEST?commonMacro=stopfinder&type_sf=any&name_sf=stuttgart%20schwabstra%C3%9Fen%2022)

### Step 2: Request the departure board with the ID of the address

[http://osm.demo.mentz.net/training/XML\\_DM\\_REQUEST?commonMacro=dm&deleteAssigendStops\\_dm=1&type\\_dm=any&name\\_dm=streetID:1500000505:22:8111000:51:Schwabstra%C3%9Fen:Stuttgart:Schwabstra%C3%9Fen::Schwabstra%C3%9Fen:70197:ANY:DIVA\\_SINGLEHOUSE:1019512:5764035:MRCV:osm:0](http://osm.demo.mentz.net/training/XML_DM_REQUEST?commonMacro=dm&deleteAssigendStops_dm=1&type_dm=any&name_dm=streetID:1500000505:22:8111000:51:Schwabstra%C3%9Fen:Stuttgart:Schwabstra%C3%9Fen::Schwabstra%C3%9Fen:70197:ANY:DIVA_SINGLEHOUSE:1019512:5764035:MRCV:osm:0)

```
stopEvents: [
  - {
    - location: {
      id: "de:08111:6052:3:2",
      isGlobalId: true,
      name: "Stuttgart Schwabstraße",
      disassembledName: "Pos. 2",
      type: "platform",
      pointType: "POSITION",
      + coord: [2],
      + properties: {3},
      - parent: {
        id: "de:08111:6052",
        isGlobalId: true,
        name: "Stuttgart Schwabstraße",
        disassembledName: "Schwabstraße",
        type: "stop",
        + parent: {2},
        - properties: {
          stopId: "5006052"
        },
      },
    },
    departureTimePlanned: "2021-11-08T10:19:00Z",
    + transportation: {10},
    + infos: [1],
    + properties: {1},
  },
],
```

```
- {
  - location: {
    id: "de:08111:6221:1:2",
    isGlobalId: true,
    name: "Stuttgart Feuersee",
    disassembledName: "2",
    type: "platform",
    pointType: "TRACK",
    + coord: [2],
    + properties: {3},
    - parent: {
      id: "de:08111:6221",
      isGlobalId: true,
      name: "Stuttgart Feuersee",
      disassembledName: "Feuersee",
      type: "stop",
      + parent: {2},
      - properties: {
        stopId: "5006221"
      },
    },
  },
  departureTimePlanned: "2021-11-08T10:19:00Z",
  + transportation: {9},
  + hints: [3],
  + properties: {1},
},
```

## Challenge

Request a departure board for *Stuttgart Schwabstraße 22*. Select your preferred stop, e.g. *Stuttgart Schwabstraße*.

# DepartureMonitor-Request - Design Variants

## Solution

### Step 1: Locality search with the StopFinder-Request

[http://osm.demo.mentz.net/training/XML\\_STOPFINDER\\_REQUEST?commonMacro=stopfinder&type\\_sf=any&name\\_sf=stuttgart schwabstraÙe 22](http://osm.demo.mentz.net/training/XML_STOPFINDER_REQUEST?commonMacro=stopfinder&type_sf=any&name_sf=stuttgart schwabstraÙe 22)

### Step 2: Request the departure board with the ID of the stop *Stuttgart SchwabstraÙe*

[http://osm.demo.mentz.net/training/XML\\_DM\\_REQUEST?commonMacro=dm&deleteAssignedStops\\_dm=1&type\\_dm=any&name\\_dm=5006052&doNotSearchForStops\\_dm=1](http://osm.demo.mentz.net/training/XML_DM_REQUEST?commonMacro=dm&deleteAssignedStops_dm=1&type_dm=any&name_dm=5006052&doNotSearchForStops_dm=1)

```
locations: [
- {
  id: "streetID:150000505:22:8111000:51:SchwabstraÙe:Stutt
  name: "Stuttgart, SchwabstraÙe 22",
  disassembledName: "SchwabstraÙe 22",
+ coord: [2],
  streetName: "SchwabstraÙe",
  buildingNumber: "22",
  type: "singlehouse",
  matchQuality: 1000,
  isBest: true,
+ parent: {3},
- assignedStops: [
  - {
    id: "de:08111:6052",
    isGlobalId: true,
    name: "Stuttgart SchwabstraÙe",
    disassembledName: "SchwabstraÙe",
    type: "stop",
+ coord: [2],
+ parent: {2},
    distance: 16,
    duration: 0,
+ productClasses: [3],
    connectingMode: 100,
- properties: {
      stopId: "5006052"
    },
  },
- {
  id: "de:08111:2207",
  isGlobalId: true,
  name: "Stuttgart Schwab-/ReinsburgstraÙe",
  disassembledName: "Schwab-/ReinsburgstraÙe",
  type: "stop",
- coord: [
```

```
locations: [
- {
  id: "de:08111:6052",
  isGlobalId: true,
  name: "Stuttgart, SchwabstraÙe",
  disassembledName: "SchwabstraÙe",
+ coord: [2],
  type: "stop",
  matchQuality: 100000,
  isBest: false,
+ parent: {3},
+ assignedStops: [1],
+ properties: {2},
}
],
stopEvents: [
- {
+ location: {9},
  departureTimePlanned: "2021-11-08T10:45:00Z",
+ transportation: {10},
+ infos: [1],
+ properties: {1},
},
+ {6},
+ {6},
+ {5},
```

**IsShowTrainsExplicit = 1**

Includes trains in the list of routes.

**limit**

Maximum number of departures. By default up to 40 departures within a maximum of 2 days are displayed.

# DepartureMonitor-Request - Realtime

---




## useRealtime=1

Activates the realtime output.

This parameters could be part of the HTT parameter macro

`commonMacro=dm`.

A realtime controlled `stopEvent` includes not only the scheduled departure time (`departureTimePlanned`) but also an estimated departure time (`departureTimeEstimated`).

Stop: Dundonald, Ulster Hospital		<input type="checkbox"/> Later
10:15	 505 / Belfast City Centre, Europa Buscentre	
10:17 Exp'd: 10:18	 4a / Dundonald, Ballybeen Rank Road	
10:32	 4a / Belfast City Centre, Donegall Square West	

```
departureTimePlanned: "2018-04-08T09:17:00Z",  
departureTimeEstimated: "2018-04-08T09:18:00Z",
```

ServingLines-Request

# ServingLines-Request – Table of Content



1. Input and Output
2. Direct Line Search
3. Line Search via Stop Search
4. Optional Parameters
5. Line Input (unique ID)

# ServingLines-Request – Input and Output Request

The ServingLines-Request is used for line search. It provides line search via stop search and direct line search.

## Request

```
http://osm.demo.mentz.net/training/XML_SERVINGLINES_REQUEST?commonMacro=servinglines
```

## Example

```
http://osm.demo.mentz.net/training/XML_SERVINGLINES_REQUEST?commonMacro=servinglines&mode=odv&type_sl=stopID&name_sl=de:08111:6221
```

# ServingLines-Request – Input and Output Request

## Part of the Request

- Error Handling (as described per Common Functionality)
- Locality Input (as described per StopFinder-Request)

## Parameter Suffix for Locality Input

The parameter suffix `<usage>` for ServingLines-Request is `s1`.

# ServingLines-Request – Input and Output

## JSON Output / Mandatory Parameters

### Lines

The response provides an array of `lines`.

### Mandatory Parameters

These parameters are given by parameter injection or configuration:

- `outputFormat=rapidJSON` (activates the JSON API)
- `coordOutputFormat=WGS84[dd.ddddd]` (coord format set to WGS 84)
- `locationServerActive=1` (activates EFALocationServer for locality search)

Note: Parameter injection works only for requests with HTTP parameters.

Further customer specific parameters could be included in an HTTP parameter macro `commonMacro=servinglines`.

### mode

Search mode: line search via locality search or direct line search.

Values are `odv` (localities) or `line`.

```
lines: [
  - {
      id: "ddb:92T01: :H:j21",
      name: "S-Bahn S1",
      disassembledName: "S1",
      number: "S1",
      + product: {4},
      + operator: {3},
      + destination: {3},
      + properties: {6},
    },
  + {8},
  + {9},
  + {9},
```

`mode=line` required

## **lineName**

Search string: name of the searched line.

## Challenge

Search for S-Bahn line S2.

# ServingLines-Request – Direct Line Search

## Solution

```
http://osm.demo.mentz.net/training/XML_SERVINGLINES_REQUEST?commonMa  
cro=servinglines&mode=line&lineName=S2
```

# ServingLines-Request – Line Search via Stop Search

`mode=odv` required

**`name_sl`**

ID of the stop.

**`type_sl = stopID`**

Type of locality is `stopID`.

## Solution

Which lines stop at *Stuttgart Feuersee*?

# ServingLines-Request – Line Search via Stop Search



## Solution

```
http://osm.demo.mentz.net/training/XML_SERVINGLINES_REQUEST?commonMa  
cro=servinglines&mode=odv&type_sl=stopID&name_sl=de:08111:6221
```

# ServingLines-Request – Optional Parameters

## **lineReqType**

Presentation type – works as a bit mask to combine presentation types

Example: `lineReqType=5` -> 1 + 4

-> Departure Monitor and Timetable

Value	Description
1	Departure Monitor (DM)
2	Stop Timetable (STT)
4	Timetable (TTB)
8	Route Maps
16	Station Timetable

## **mergeDir = 1**

By default both, inbound and outbound, are taken into account. This parameter merges the directions, thus only inbound is returned if both are available.

## **IsShowTrainsExplicit = 1**

By default no services for trains are returned, if not switched on by this parameter.

# ServingLines-Request – Line Input (unique ID)

The unique ID determined by ServingLines-Request may be used as an input for any request that requires a line. Therefore HTTP parameter **line** is used.

## Example

```
line=ddb:92T01: :R:j21
```

LineStop-Request

# LineStop-Request – Table of Content

1. Input and Output
2. Additional Information

# LineStop-Request – Input and Output Request

The LineStop-Request returns the stops of a line.

## Request

```
http://osm.demo.mentz.net/training/XML_LINESTOP_REQUEST?commonMacro=linestop
```

## Example

```
http://osm.demo.mentz.net/training/XML_LINESTOP_REQUEST?commonMacro=linestop&line=mvv:01002:E:H:s21
```

## Part of the LineStop-Request

- Error Handling (as described per Common Functionality)
- Line Input (as described per ServingLines-Request)

# LineStop-Request – Input and Output JSON Output / Mandatory Parameters

## Locations

`locationSequence` is an array of locations in the well known format.

## Mandatory Parameters

These parameters are given by parameter injection or configuration:

- `outputFormat=rapidJSON` (activates the JSON API)
- `coordOutputFormat=WGS84[dd.ddddd]` (coord format set to WGS 84)

Note: Parameter injection works only for requests with HTTP parameters.

Further customer specific parameters could be included in an HTTP parameter macro `commonMacro=linestop`.

## `line`

Line of which the coordinate sequence is requested. Value: unique route ID.

```
locationSequence: [
  - {
    isGlobalId: true,
    id: "de:09174:6950",
    name: "Altomünster",
    type: "stop",
    - parent: {
      id: "placeID:9174111:1",
      name: "Altomünster",
      type: "locality",
    },
    - properties: {
      stopId: "1006950"
    },
  },
  + {6},
  + {6},
]
```

# LineStop-Request – Additional Information



**allStopInfo = 1**

Provides additional information, e.g. areas and platforms.

STT-Request

TTB-Request

# ROP-Request

# ROP-Request – Table of Contents



1. Input and Output
2. LVP (Route Map)
3. SPA (Stop Map)
4. Optional Parameters

# ROP-Request – Input and Output Request

The ROP-Request provides route maps: LVPs (Linienverlaufsplan - route maps) and SPAs (Stadtplanausschnitt – street maps).

## Request

`http://osm.demo.mentz.net/training/XML_ROP_REQUEST?`

## Example

`http://osm.demo.mentz.net/training/XML_ROP_REQUEST?commonMacro=rop&line=ddb:92T01: :R:j22&reqType=LVP&mode=line`

# ROP-Request – Input and Output Request

## Part of the Request

- Error Handling (as described per Common Functionality)
- Locality Input (as described per StopFinder-Request)
- Line Input (as described per ServingLines-Request)

# ROP-Request – Input and Output

## JSON Output / Mandatory Parameters

---

### Lines

The response provides a base 64 encoded PDF in a `stream`.

### Mandatory Parameters

These parameters are given by parameter injection or configuration:

- `outputFormat=rapidJSON` (activates the JSON API)

Note: Parameter injection works only for requests with HTTP parameters.

These parameters should be included in the HTTP parameter macro

#### **commonMacro=rop:**

- `command=direct`
- `useAllStops=1` (all stop points, includes stop points which cannot be reached by a walk, e.g. some underground stop points)
- `lsShowTrainsExplicit=1` (enables trains)

```
{
  version: "10.5.17.3",
  location: [ ],
  - download: {
    url: "/avv_web/ROP/20221206-045728/STOP_-1.pdf",
    size: 1,
  },
  stream:
  "JVBERi0xLjQKJeTjz9IKMSAwIG9iagpbL1BERi9JbWFnZUIvSW1hZ2VDI
}
```

# ROP-Request – Input and Output Mandatory Parameters

**reqType = LVP | SPA**

Activates LVPs or SPAs.

**mode = line | odv**

Computation modus is `line` or `odv`. Mode `line` is required if LVP or SPA is requested via line input. Mode `odv` is required if they are requested via stop, point or coordinate.

# ROP-Request – LVP

## Mandatory Parameters

`reqType=LVP` and `mode=line` are required.

### **line**

Unique route ID or

```
<network>:<DIVALne>:<supplement>:<direction>:<project>:<motType>:<type>:<stopSequence>:<lineVersion>
```

### Example

```
http://osm.demo.mentz.net/training/XML_ROP_REQUEST?commonMacro=rop&line=ddb:92T01: :R:j22&reqType=LVP&mode=line
```

# ROP-Request – LVP

## Optional Parameters

**printOutsPerStop = 1 | 2**

Adds 1 or 2 copies (depending on the value of the parameter) of the SPA for each passed stop.

**spaOut = bothdirections | mainroutefirst**

Controls the output of SPAs (see parameter `printOutsPerStop`, which is required for this functionality). `bothdirections` returns the SPAs of inbound route, if the inbound route differs from outbound. `mainroutefirst` returns the SPAs of the main route first and then of the secondary route.

**subReqType = overview**

Removes detailed maps and add only overview map to PDF.

# ROP-Request – SPA

## Line Input: mandatory Parameters

### Line Input

`reqType=SPA` and `mode=line` are required.

### **line**

Unique route ID or

```
<network>:<DIVALne>:<supplement>:<direction>:<project>:<motType>:<type>:<stopSequence>:<lineVersion>
```

### Example

```
http://osm.demo.mentz.net/training/XML_ROP_REQUEST?commonMacro=rop&line=ddb:92T01: :R:j22&reqType=SPA&mode=line
```

# ROP-Request – SPA

## Line Input: optional Parameters

**printOutsPerStop = 1 | 2**

Adds 1 or 2 copies (depending on the value of the parameter) of the SPA for each passed stop. The parameter is only required if value 2 is needed as 1 is the default.

**spaOut = bothdirections | mainroutefirst**

Controls the output of SPAs (see parameter `printOutsPerStop`, which is required for this functionality). `bothdirections` returns the SPAs of inbound route, if the inbound route differs from outbound. `mainroutefirst` returns the SPAs of the main route first and then of the secondary route.

# ROP-Request – SPA

## Line Input: mandatory Parameters

### Input of Stop, Point or Coordinate

`reqType=SPA` and `mode=odv` are required.

### Stop

**stopID**

Unique route ID of the stop.

### POI

**poiID**

ID of the POI.

**omc**

OMC of the POI.

# ROP-Request – SPA

## Line Input: mandatory Parameters

### Coordinate

**namecoord**

Coordinate system.

**xcoord**

X coordinate.

**ycoord**

Y coordinate.

### Example

```
http://osm.demo.mentz.net/training/XML_ROP_REQUEST?commonMacro=rop&st  
topID=de:09761:100&reqType=SPA&mode=odv
```

# ROP-Request – SPA

## Optional Parameters

### **spaSkip**

Suppresses elements of the SPA, depending on value.

Note: This functionality requires additional configuration of the ITKernel.

Value:

`index` – No short distance destinations

`map` – No map.

`no` – No short distances and no map

Check-MultiStopTimetable-Request  
t

## 1. Input and Output

# Check-MultiStopTimetable-Request – Input and Output Request

The Check-Multistoptimetable-Request checks if there are multi stop timetables available for a stop

## Request

```
http://osm.demo.mentz.net/training/XML_CHECK_MULTISTOPTIMETABLE_REQUEST?
```

## Example

```
http://osm.demo.mentz.net/training/XML_CHECK_MULTISTOPTIMETABLE_REQUEST?type_cmstt=any&name_cmstt=5006221
```

## Part of the Check-MultiStopTimetable-Request

- Locality Input (as described per StopFinder-Request)
- Error Handling (as described per Common Functionality)

# Check-MultiStopTimetable-Request – Input and Output Request



## Parameter Suffix for Locality Input

The parameter suffix `<usage>` is `cmstt`.

# Check-MultiStopTimetable-Request – Input and Output JSON Output

---

## Predefined Stop Timetables

`predefinedStopTimetables` is an array of stop timetable. A stop contains multi stop timetable if the array is not empty.

# Check-MultiStopTimetable-Request – Input and Output

## Mandatory Parameters

These parameters are given by parameter injection or configuration:

- `outputFormat=rapidJSON` (activates the JSON API)
- `locationServerActive=1` (activates EFALocationServer for locality search)

Note: Parameter injection works only for requests with HTTP parameters.

Check-MultiStopTimetable-Request requires Locality input but has no further HTTP parameters.

Coord-Request

# Coord-Request – Table of Contents



1. Input and Output
2. Filters
3. Bounding Box
4. Radial Search
5. Optional Parameters

# Coord-Request – Input and Output Request

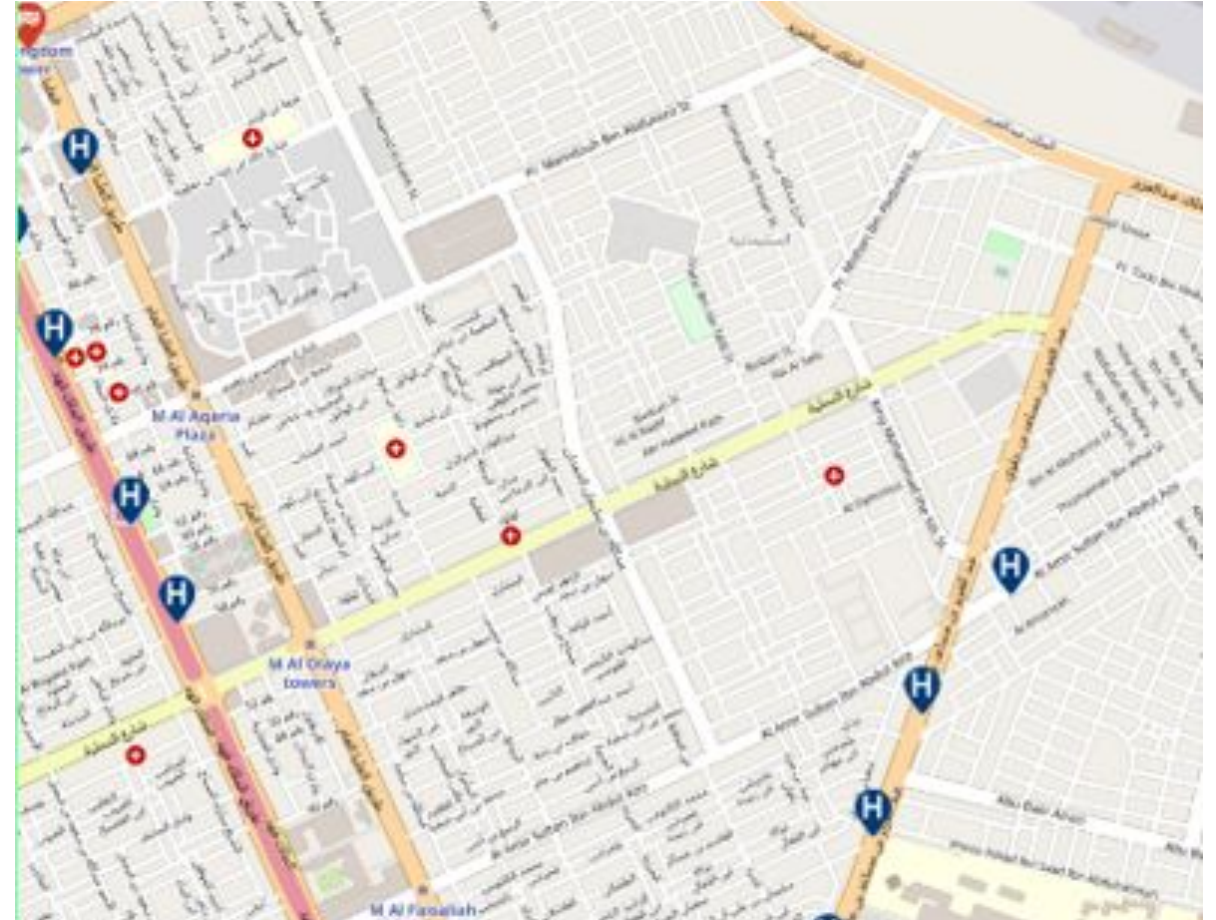
Request coordinates of objects, e.g. stops or POIs.

## Request

```
http://osm.demo.mentz.net/training/XML_COORD_REQUEST  
?commonMacro=coord
```

## Example

```
http://osm.demo.mentz.net/training/XML_COORD_REQUEST  
?commonMacro=coord&boundingBox=&boundingBoxLU=9.15:4  
8.77:WGS84[dd.ddddd]&boundingBoxRL=9.10:48.82:WGS84[  
dd.ddddd]&type_1=STOP&inclFilter=1
```



# Coord-Request – Input and Output

## JSON Output

### Locations

`locations` is an array of objects found. They have a `name`, an `id` and a `coordinate` (`coord`).

### Properties

`properties` provides some information relevant for Coord-Request:

- `distance` (distance from the centre coordinate)
- `STOP_MAJOR_MEANS` (Icon ID)

### List of Transports

The array `productClasses` provides the list of means of transport.

```
locations: [
  + {8},
  + {8},
  - {
    id: "de:08111:2429",
    isGlobalId: true,
    name: "Paul-Lincke-Straße",
    type: "stop",
    - coord: [
      48.7843,
      9.13179,
    ],
    - parent: {
      id: "placeID:8111000:51",
      name: "Stuttgart",
      type: "locality",
    },
    - productClasses: [
      5
    ],
    - properties: {
      distance: 1961,
      STOP_GLOBAL_ID: "de:08111:2429",
      STOP_NAME_WITH_PLACE: "Stuttgart Paul-Lincke-Straße",
      STOP_MAJOR_MEANS: "3",
      STOP_MEANS_LIST: "107,201",
      STOP_MOT_LIST: "5",
      STOP_TARIFF_ZONES:vvs: "1",
    },
  },
],
```

# Coord-Request – Input and Output

## Mandatory Parameters

These parameters are given by parameter injection or configuration:

- `outputFormat=rapidJSON` (activates the JSON API)
- `coordOutputFormat=WGS84 [dd.ddddd]` (coord format set to WGS 84)

Note: Parameter injection works only for requests with HTTP parameters.

Further customer specific parameters could be included in an HTTP parameter macro **`commonMacro=coord`**.

# Coord-Request – Input and Output

## Mandatory Parameters

### Filters

To activate the filters `inclFilter=1` is required.

#### `type_<filter index>`

With this parameter a certain type of point can be chosen. Several point types can be chosen by using this parameter multiple times. Hereby for each point type another index `<filter index>` is assigned. There are the following types:

Point Types	Description
ANY	All points
BUS_POINT	Bus stops
ENTRANCE	Entrances (e.g. vor suburban train stops)
GIS_AREA	GIS-Areas
GIS_POINT	GIS-Points
LINE	Services, that cross the street segment passed by the coordinate <code>coord</code>
POI_AREA	Area-POIs (important area points)
POI_POINT	Point-POIs (important points)
STOP	Stops
STREET	Streets
AREA	Stop-Areas

It is possible to provide a bounding box and only objects inside bounding box are calculated. Therefore the following parameters must be requested.

**boundingBox=1**

Enables the bounding box. No value must be provided.

**boundingBoxLU** and **boundingBoxRL**

Left upper and right lower coordinate.

**Value:** <x>:<y>:<coordinate system>

## Example

```
http://osm.demo.mentz.net/training/XML_COORD_REQUEST?commonMacro=coord&boundingBox=&boundingBoxLU=9.15:48.77:WGS84[dd.ddddd]&boundingBoxRL=9.10:48.82:WGS84[dd.ddddd]&type_1=STOP&inclFilter=1
```

An alternative to the bounding box is the radial search. The following parameters are needed:

## **coord**

This parameter specifies the middle coordinate, which is the focus of the search for the points.

**Value:** `<x>:<y>:<coordinate system>`

## **radius\_<filter index>**

With this filter the radius in which the search should be done can be given in meters. The focus of the search is the middle coordinate `coord`.

The radius of each point that is found by `type_<filter index>` can be specified separately by the parameter `radius_<filter index>`. To activate the filters `inclFilter=1` is required.

## Challenge

Calculate stops in a radius of 500 meters for the center coordinate

9.15:48.77:WGS84 [dd.ddddd] .

## Solution

```
http://osm.demo.mentz.net/training/XML_COORD_REQUEST?commonMacro=coord&type_1=STOP&inclFilter=1&radius_1=500&coord=9.15:48.77:WGS84[dd.dddd]
```

## **max**

This parameter is the maximum number of object that are to be determined and displayed. The objects closest to the center are chosen. By default there is no limit.

Value: Integer

## **deadline**

Date for which the stops are valid.

*Value:* <JJJJ><MM><TT>, default: date of the server.

## **purpose**

With this parameter the finding of points (POI) can be reduced to points with a specific purpose. This means that certain groups of important points can be treated separately.

Hint: Using the purpose, it is possible to assign several POIs different kinds of configurations. This is done through different sections with the name of the purpose in the configuration file of the EFAITKernel.

This functionality is identical to the restriction of the search space by using the draw class with the parameter `inclDrawClasses_<filter index>`.

# GeoObject-Request

# GeoObject-Request – Table of Contents

1. Input and Output
2. Optional Parameters
3. Bounding Box

# GeoObject-Request – Input and Output Request

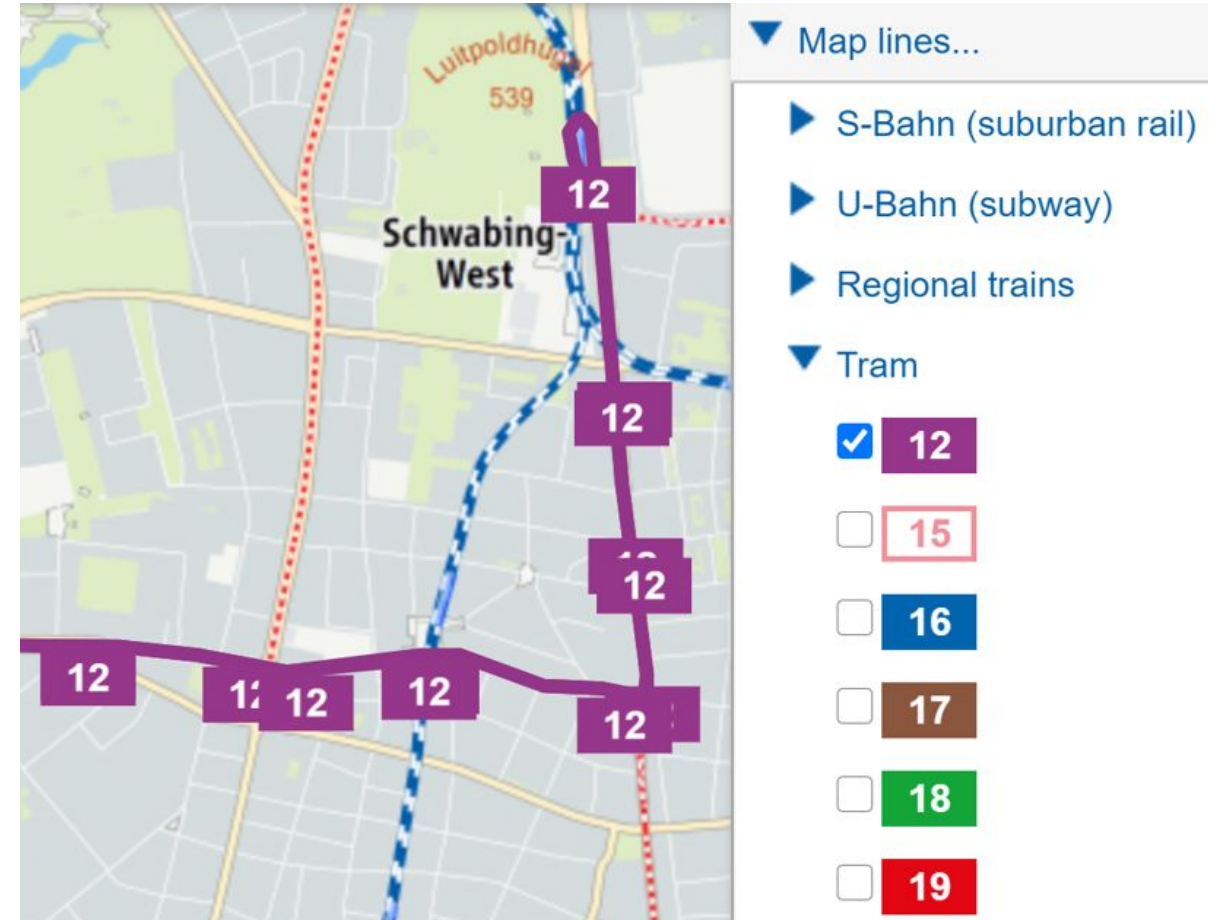
Generate a sequence of coordinates and of all passed stops of a provided service. The coordinate sequence and the points can be shown, for example on an interactive map.

## Request

```
https://osm.demo.mentz.net/training/XML_GEOOBJECT_REQUEST?commonMacro=geobj
```

## Example

```
http://osm.demo.mentz.net/training/XML_GEOOBJECT_REQUEST?commonMacro=geobj&line=vvs:10002:%20:R:j21
```



# GeoObject-Request – Input and Output

## JSON Output

---

### Transportation

The element `transportation` contains apart of the usual objects:

- `coords` (array with the coordinate sequence of the service)
- `locationSequence` (optional list of passed stops)

```
transportations: [  
  - {  
    id: "vvs:10002: :R:j21",  
    name: "S-Bahn S2",  
    disassembledName: "S2",  
    number: "S2",  
    description: "Filderstadt - Flughafen/Messe - Stuttgart - Schorndorf",  
    + product: {4},  
    + operator: {2},  
    + destination: {3},  
    + properties: {5},  
    + coords: [8],  
    + locationSequence: [29],  
  }  
],
```

# GeoObject-Request – Input and Output JSON Output

## Coordinates

The coordinate sequence is found in `coords`.

## Stop

The passed stops are part of the

`locationSequence`:

- `name` – name
- `coord` – coordinate
- `productClasses` – list of transports

Part of `properties`:

- `STOP_MAJOR_MEANS` – Icon

```
coords: [  
  - [  
    - [ 24.68033,  
        46.70126,  
      ],  
    - [ 24.68116,  
        46.70087,  
      ],  
  ],
```

```
- locationSequence: [  
  - {  
    isGlobalId: true,  
    id: "23620301",  
    name: "Dirab_01",  
    type: "platform",  
    + coord: [2],  
    + parent: {4},  
    - productClasses: [  
      5  
    ],  
    - properties: {  
      STOP_MEANS: "32",  
      STOP_MAJOR_MEANS: "3",  
    },  
  },  
  - {  
    isGlobalId: true,  
    id: "23620201",
```

# GeoObject-Request – Input and Output

## Mandatory Parameters

These parameters are given by parameter injection or configuration:

- `outputFormat=rapidJSON` (activates the JSON API)
- `coordOutputFormat=WGS84 [dd.ddddd]` (coord format set to WGS 84)

Note: Parameter injection works only for requests with HTTP parameters.

Further customer specific parameters could be included in an HTTP parameter macro **`commonMacro=geobj`**.

# GeoObject-Request – Input and Output

## Mandatory Parameters

### **line**

Line of which the coordinate sequence is requested.

**Value:** unique route ID or <network>:<DIVA line>:<supplement>:<direction>  
:<project>:<mot type>:<type>::<stop sequence>:<line version>

<stopSequence> has to be set to 1 to request the stop sequence.

### Example

```
http://osm.demo.mentz.net/training/XML_GEOOBJECT_REQUEST?commonMacro  
=geoobj&line=vvs:10002:%20:R:j21
```

## **filterDate**

This parameter provides the possibility to get the coordinate sequence and passed stops of one specific date. The format is YYYYMMDD.

Hint: This can be useful if the line has different routes e.g. for weekends.

Analog to the CoordInfo-Request it is also possible to provide a bounding box and only coordinate sequence and passed stops inside bounding box is calculated. Therefore the following parameters must be requested.

**boundingBox=1**

Enables the bounding box. No value must be provided.

**boundingBoxLU** and **boundingBoxRL**

Left upper and right lower coordinate.

**Value:** <x>:<y>:<coordinate system>

# TripStopTimes-Request

# TripStopTimes-Request – Table of Contents



1. Input and Output
2. Optional Parameters

# TripStopTimes-Request – Input and Output Request

The TripStopTimes-Request is used to get the stop sequence of a journey - including arrival and departure times.

## Request

```
http://osm.demo.mentz.net/training/XML_TRIPSTOPTIMES_REQUEST?commonMacro=tripstoptimes
```

## Example

```
http://osm.demo.mentz.net/training/XML_TRIPSTOPTIMES_REQUEST?commonMacro=tripstoptimes&tripCode=963&stopID=5006052&time=1432&date=20211109&line=vvs:10006:%20:R:j21vvs:10006:%20:R:j21
```

14:45 ○ München Hbf  
↓ Footpath: 98 m, 2 Min.

14:47 ○ Tram), Bus, U, Hauptbahnhof (S, München plat. 1)  
S-Bahn S7  
towards Kreuzstraße

^ 5 stops, 9 Min.

14:48 ○ Karlsplatz (Stachus)  
14:50 ○ Marienplatz  
14:52 ○ Isartor  
14:53 ○ Rosenheimer Platz  
14:56 ○ Ostbahnhof plat. 4

# TripStopTimes-Request – Input and Output JSON Output

---

## Stop Sequence

The array `locationSequence` contains stops in the well known format.

```
transportation: {
  id: "vvs:10006: :R:j21vvs",
  - product: {
    class: 100,
    iconId: 100,
  },
  - properties: {
    tripCode: 0,
    lineDisplay: "LINE",
  },
  - locationSequence: [
    - {
      isGlobalId: true,
      id: "de:08111:6052:1:2",
      name: "Stuttgart Schwabstraße",
      disassembledName: "Gleis 2",
      type: "platform",
      pointType: "TRACK",
      + coord: [2],
      niveau: -300,
      + parent: {9},
      + productClasses: [3],
      + properties: {9},
      departureTimePlanned: "2021-11-09T13:27:00Z",
    },
    + {13},
    + {13},
    + {13},
  ],
}
```

# TripStopTimes-Request – Input and Output Mandatory Parameters

These parameters are given by parameter injection or configuration:

- `outputFormat=rapidJSON` (activates the JSON API)
- `coordOutputFormat=WGS84 [dd.ddddd]` (coord format set to WGS 84)

Note: Parameter injection works only for requests with HTTP parameters.

Further customer specific parameters could be included in an HTTP parameter macro **`commonMacro=tripstoptimes`**.

# TripStopTimes-Request – Input and Output Mandatory Parameters

Parameters to identify the line of which the coordinate sequence and stops are requested: Take the parameters from the response of the Trip- or DM-Request.

## line

Unique route ID or

<network>:<DIVALine>:<supplement>:<direction>:<project>:<motType>:<type>::<stopSequence>:<lineVersion>

## stopID

ID of the origin stop.

## tripCode

Key of the journey.

## date

Date of the departure <YYYYMMDD>.

## time

Time of the departure <HHMM>.

```
journeys: [
  - {
    rating: 0,
    isAdditional: false,
    interchanges: 0,
    - legs: [
      - {
        duration: 120,
        - origin: {
          isGlobalId: true,
          id: "de:08111:6052:1:2",
          name: "Stuttgart Schwabstraße",
          disassembledName: "Gleis 2",
          type: "platform",
          pointType: "TRACK",
          + coord: [2],
          niveau: -300,
          - parent: {
            isGlobalId: true,
            id: "de:08111:6052",
            name: "Stuttgart Schwabstraße",
            disassembledName: "Schwabstraße",
            type: "stop",
            + parent: {3},
            - properties: {
              stopId: "5006052"
            },
            + coord: [2],
            niveau: 0,
          },
          + productClasses: [3],
          departureTimePlanned: "2021-11-09T13:27:00Z",
          departureTimeEstimated: "2021-11-09T13:27:00Z",
          + properties: {8},
        },
        + destination: {13},
        - transportation: {
          id: "vvs:10006: :R:j21",
          name: "S-Bahn S6",
          disassembledName: "S6",
          number: "S6",
          description: "Stuttgart - Leonberg - Weil der Stadt",
          + product: {4},
          + operator: {2},
          + destination: {3},
          - properties: {
            trainName: "S-Bahn",
            trainType: "S",
            trainNumber: "7942",
            isROP: true,
            tripCode: 063,
            timetablePeriod: "Fahrplan 2021",
            lineDisplay: "LINE",
          },
        },
      },
    ],
  },
  ..
]
```

# TripStopTimes-Request – Input and Output

## Mandatory Parameters

### Challenge

Get the stop sequence of the *S-Bahn S2*, which departs at November 9th, 2021, 15:00 from *Stuttgart Schwabstraße*.

### Hint:

```
http://osm.demo.mentz.net/training/XML_DM_REQUEST?commonMacro=dm&type_dm=any&name_dm=5006052
```

# TripStopTimes-Request – Input and Output Mandatory Parameters

## Solution

[http://osm.demo.mentz.net/training/XML\\_TRIPSTOPTIMES\\_REQUEST?commonMacro=tripstoptimes &tripCode=96&stopID=5006052&time=1500&date=20211109&line=ddb:92T02:%20:H:j21](http://osm.demo.mentz.net/training/XML_TRIPSTOPTIMES_REQUEST?commonMacro=tripstoptimes &tripCode=96&stopID=5006052&time=1500&date=20211109&line=ddb:92T02:%20:H:j21)

For a departure board often only the next stops are required, not the previous...

```
locations: [1],
stopEvents: [
  - {
    - location: {
      id: "de:08111:6052:1:1",
      isGlobalId: true,
      name: "Stuttgart Schwabstraße",
      disassembledName: "1",
      type: "platform",
      pointType: "TRACK",
      + coord: [2],
      + properties: {3},
      - parent: {
        id: "de:08111:6052",
        isGlobalId: true,
        name: "Stuttgart Schwabstraße",
        disassembledName: "Schwabstraße",
        type: "stop",
        + parent: {2},
        - properties: {
          stopId: "5006052"
        },
      },
    },
    departureTimePlanned: "2021-11-09T14:00:00Z",
    - transportation: {
      id: "ddb:92T02: :H:j21",
      name: "S-Bahn S2",
      disassembledName: "S2",
      number: "S2",
      + product: {4},
      + operator: {3},
      + destination: {3},
      - properties: {
        trainName: "S-Bahn",
        trainType: "S",
        trainNumber: "7241",
        tripCode: 96,
        lineDisplay: "LINE",
      },
      + origin: {3},
    },
    - infos: [
```

# TripStopTimes-Request – Optional Parameters

**useRealtime = 0|1**

Provide realtime arrival/departure times in response.

**tStOTType**

Filters the stop sequence.

Value	Description
ALL	All stops
NEXT	Next stops relative to the given stop (see parameter stopID)
PREVIOUS	Previous stops relative to the given stop (see parameter stopID)

StopSeqCoord-Request

# StopSeqCoord-Request – Table of Contents



1. Input and Output
2. Optional Parameters

# StopSeqCoord-Request – Input and Output

## Request

The StopSeqCoord-Request is used to get the stop sequence and the track of an option from departure board.

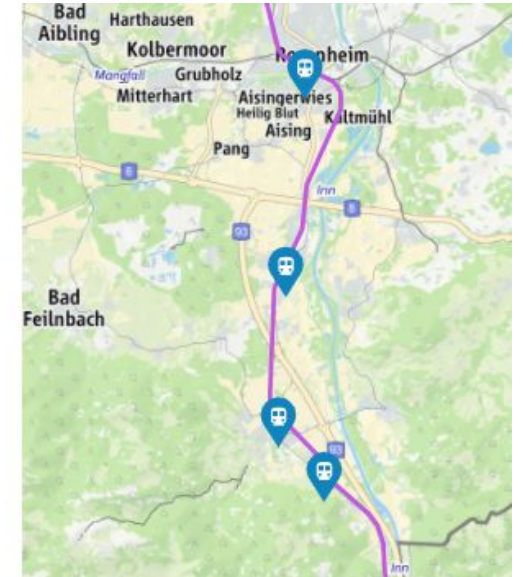
### Request

[http://osm.demo.mentz.net/training/XML\\_STOPSEQCOORD\\_REQUEST?commonMacro=stopseqcoord](http://osm.demo.mentz.net/training/XML_STOPSEQCOORD_REQUEST?commonMacro=stopseqcoord)

### Example

[http://osm.demo.mentz.net/training/XML\\_STOPSEQCOORD\\_REQUEST?commonMacro=stopseqcoord&tStOTType=NEXT&tripCode=96&stopID=5006052&time=1500&date=20211109&line=ddb:92T02:%20:H:j21](http://osm.demo.mentz.net/training/XML_STOPSEQCOORD_REQUEST?commonMacro=stopseqcoord&tStOTType=NEXT&tripCode=96&stopID=5006052&time=1500&date=20211109&line=ddb:92T02:%20:H:j21)

- 14:51 ○ Ostbahnhof
- 15:05 ○ Grafing Bahnhof
- 15:11 ○ Aßling
- 15:16 ○ Ostermünchen
- 15:21 ○ Großkarolinenfeld
- 15:27 ○ Rosenheim
- 15:36 ○ Raubling
- 15:40 ○ Brannenburg
- 15:43 ○ Flintsbach
- 15:49 ○ Oberaudorf
- 15:54 ○ Kiefersfelden
- 15:58 ○ Kufstein



# StopSeqCoord-Request – Input and Output

## JSON Output

---

### Transportation

The element `transportation` contains apart of the usual objects:

- `coords` (array with the coordinate sequence of the service)
- `locationSequence` (optional list of passed stops)

```
transportation: {
  id: "ddb:92T02: :H:j21",
  name: "S-Bahn S2",
  disassembledName: "S2",
  number: "S2",
+ product: {4},
+ operator: {3},
+ destination: {3},
+ properties: {4},
+ locationSequence: [10],
+ coords: [1],
},
```

# StopSeqCoord-Request – Input and Output

## Mandatory Parameters

These parameters are given by parameter injection or configuration:

- `outputFormat=rapidJSON` (activates the JSON API)
- `coordOutputFormat=WGS84 [dd.ddddd]` (coord format set to WGS 84)

Note: Parameter injection works only for requests with HTTP parameters.

Further customer specific parameters could be included in an HTTP parameter macro **`commonMacro=stopseqcoord`**.

# StopSeqCoord-Request – Input and Output

## Mandatory Parameters

Parameters to identify the line of which the coordinate sequence and stops are requested: Take the parameters from the response of the Trip- or DM-Request.

### line

Unique route ID or

<network>:<DIVALine>:<supplement>:<direction>:<project>:<motType>:<type>::<stopSequence>:<lineVersion>.

### stop

ID of the stop.

### tripCode

Key of the journey.

### date

Date of the departure <YYYYMMDD>.

### time

Time of the departure <HHMM>.

```
journeys: [
  - {
    rating: 0,
    isAdditional: false,
    interchanges: 0,
    - legs: [
      - {
        duration: 120,
        - origin: {
          isGlobalId: true,
          id: "de:08111:6052:1:2",
          name: "Stuttgart Schwabstraße",
          disassembledName: "Gleis 2",
          type: "platform",
          pointType: "TRACK",
          + coord: [2],
          niveau: -300,
          - parent: {
            isGlobalId: true,
            id: "de:08111:6052",
            name: "Stuttgart Schwabstraße",
            disassembledName: "Schwabstraße",
            type: "stop",
            + parent: {3},
            - properties: {
              stopId: "5006052"
            },
            + coord: [2],
            niveau: 0,
          },
          + productClasses: [3],
          + departureTimePlanned: "2021-11-09T13:27:00Z",
          + departureTimeEstimated: "2021-11-09T13:27:00Z",
          + properties: {8},
        },
        + destination: {13},
        - transportation: {
          id: "vvs:10006: :R:j21",
          name: "S-Bahn S6",
          disassembledName: "S6",
          number: "S6",
          description: "Stuttgart - Leonberg - Weil der Stadt",
          + product: {4},
          + operator: {2},
          + destination: {3},
          - properties: {
            trainName: "S-Bahn",
            trainType: "S",
            trainNumber: "7942",
            isROP: true,
            + tripCode: 063,
            + timetablePeriod: "Fahrplan 2021",
            lineDisplay: "LINE",
          },
        },
      },
    ],
  },
  ..
]
```

# StopSeqCoord-Request – Input and Output

## Mandatory Parameters

### Challenge

Get the stop sequence and coordinates of the *S-Bahn S2*, which departs at November 9th, 2021, 15:00 from *Stuttgart Schwabstraße*.

### Hint:

```
http://osm.demo.mentz.net/training/XML_DM_REQUEST?commonMacro=dm&type_dm=any&name_dm=5006052
```

# StopSeqCoord-Request – Input and Output

## Mandatory Parameters

### Solution

```
http://osm.demo.mentz.net/training/XML_STOPSEQCOORD_REQUEST?commonMacro=stopseqcoord&tStOTType=NEXT&tripCode=96&stopID=5006052&time=1500&date=20211109&line=ddb:92T02:%20:H:j21
```

# StopSeqCoord-Request – Optional Parameters

## tStOTType

Filters the stop and coordinate sequence.

Value	Description
ALL	All stops/coordinates
NEXT	Next stops/coordinates relative to the given stop (see parameter stopID)
PREVIOUS	Previous stops/coordinates relative to the given stop (see parameter stopID)

# MapRoute-Request

AddInfo-Request

1. Input and Output
2. Filters
3. Additional Information / Reduction of the Response

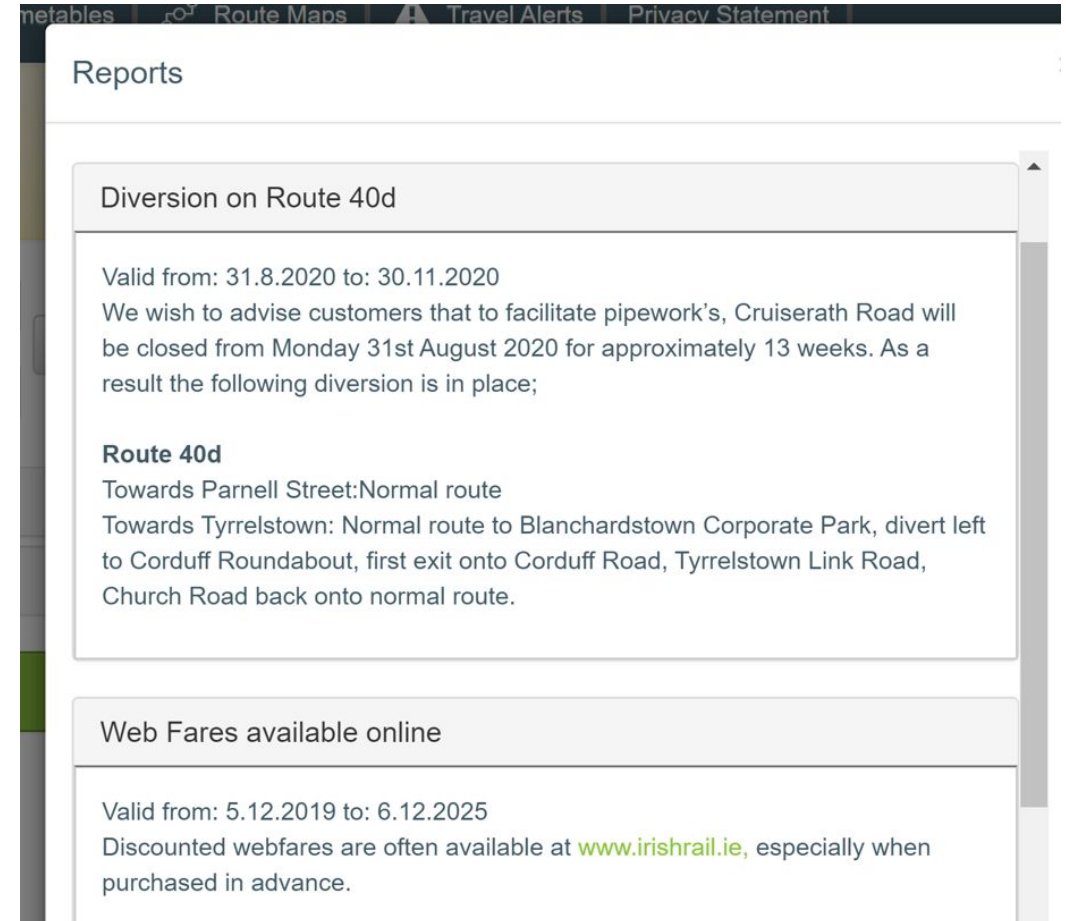
# AddInfo-Request – Input and Output Request

The AddInfo-Request is used to get the travel alerts.

Due to performance reasons it is recommendable to filter the messages. The filter criteria are determined by the HTTP parameters.

## Request

```
http://osm.demo.mentz.net/training/XML_ADDINFO_REQUEST?commonMacrot=addinfo
```



The screenshot shows a web browser window with a navigation bar containing links for 'netables', 'Route Maps', 'Travel Alerts', and 'Privacy Statement'. The main content area is titled 'Reports' and contains two alert cards. The first card is titled 'Diversion on Route 40d' and contains the following text: 'Valid from: 31.8.2020 to: 30.11.2020', 'We wish to advise customers that to facilitate pipework's, Cruiserath Road will be closed from Monday 31st August 2020 for approximately 13 weeks. As a result the following diversion is in place;', and a sub-section 'Route 40d' with directions: 'Towards Parnell Street: Normal route' and 'Towards Tyrrelstown: Normal route to Blanchardstown Corporate Park, divert left to Corduff Roundabout, first exit onto Corduff Road, Tyrrelstown Link Road, Church Road back onto normal route.' The second card is titled 'Web Fares available online' and contains the text: 'Valid from: 5.12.2019 to: 6.12.2025' and 'Discounted webfares are often available at [www.irishrail.ie](http://www.irishrail.ie), especially when purchased in advance.'

# AddInfo-Request – Input and Output JSON Output

---

## Infos

The response contains an array of `current` travel alerts, optionally an array of `historic` messages and an object which contains:

- `affected lines`
- `affected trains`
- `affected stops`

```
infos: {  
  + current: [150],  
  - affected: {  
    + lines: [606],  
    + stops: [102],  
  },  
},
```

# AddInfo-Request – Input and Output JSON Output

## Travel Alert

- Each travel alert has a `type`, `id` and `priority`.
- Use the objects `url` and `urlText` as a teaser. The content can be quite long.
- The objects `subtitle` and `content` contain the title and the content of the alert in HTML format.
- Information about the provider of the message and the source system is included in `properties`.
- The object `timestamp` includes information about creation date and time, the last modification, the validity and publishing period.
- The object `affected` informs about affected lines or stops.

```
{
  type: "lineInfo",
  id: "10480",
  version: 1,
  priority: "normal",
  + timestamps: {3},
  urlText: "Ludwigsburg: Umleitung der Linie 424 wegen Bauarbeiten.",
  url: "http://ics.efa-bw.de:80/cm/XSLT_CM_SHOWADDINFO_REQUEST?infoID=10480&seqID=1",
  content: "Aufgrund der Bauma\u2666nahme an der Kreuzung Wilhelmstra\u2666e/ Arsenalstra\u2666e in Ludwigsburg.",
  subtitle: "Ludwigsburg: Umleitung der Linie 424 wegen Bauarbeiten.",
  title: "Linie 424",
  - properties: {
    providerCode: "LVL",
    publisher: "EMS",
    sourceSystemID: "VVS",
    additionalContent: "Ludwigsburg: Umleitung der Linie 424 wegen Bauarbeiten.",
    htmlText: "<div>Aufgrund der Bauma&szlig;nahme an der Kreuzung Wilhelmstra&szlig;e/ Arsenalst",
    wmlText: "Linie 424",
    smsText: "Ludwigsburg: Umleitung der Linie 424 wegen Bauarbeiten.",
    speechText: "Linie 424",
    - source: {
      id: "VVS",
      name: "VVS EMS",
      type: "Testsystem",
    },
    mot: "bus",
    timetableChange: "lines",
  },
  - affected: {
    + lines: [2]
  },
},
```

# AddInfo-Request – Input and Output

## Mandatory Parameters

These parameters are given by parameter injection or configuration:

- `outputFormat=rapidJSON` (activates the JSON API)
- `coordOutputFormat=WGS84 [dd.ddddd]` (coord format set to WGS 84)

Note: Parameter injection works only for requests with HTTP parameters.

Further customer specific parameters could be included in an HTTP parameter macro **`commonMacro=addinfo`**.

# AddInfo-Request – Filters

## Filter for Publication or Validity Status

### **filterPublicationStatus**

Currently active (`current`) or expired (`history`) messages.

### **filterPublished = 1**

Only messages which are currently published.

# AddInfo-Request – Filters

## Filter for Publication or Validity Status

### **filterPublicationStatus**

Currently active (*current*) or expired (*history*) messages.

### **filterPublished = 1**

Only messages which are currently published.

### **filterValid = 1**

Only messages which are currently valid.

### **filterDateValid**

Messages which are active for the given day <DD-MM-YYYY>. The filter can be sent multiple times for messages which are active on several dates.

### **filterValidIntervalStart** and **filterValidIntervalEnd**

Messages which are active in the given interval <DD-MM-YYYY>.

```

filterValidIntervalStart: "2020-08-18T12:00:00Z",
filterValidIntervalEnd: "2020-08-18T12:00:00Z",
timestamps: {
  creation: "2020-08-18T12:00:00Z",
  lastModification: "2020-08-18T12:00:00Z",
  - availability: {
    from: "2020-08-17T23:01:00Z",
    to: "2020-11-29T23:59:59Z",
  },
  - validity: [
    - {
      from: "2020-08-30T23:01:00Z",
      to: "2020-11-30T00:00:00Z",
    }
  ],
},

```

# AddInfo-Request – Filters

## Filter for Message Type

### **filterInfoType**

For several types use the parameter multiple times. Available message types:

- areaInfo
- stopInfo
- stopBlocking
- lineInfo
- lineBlocking
- routeInfo
- routeBlocking
- generalInfo
- bannerInfo
- trafficInformation

# AddInfo-Request – Filters

## Filter for Priority

### **filterPriority**

Filters for messages with the priority:

- `veryLow`
- `low`
- `high`
- `veryHigh`

# AddInfo-Request – Filters

## Filter for Mode of Transport or Operator

### **filterMOTType**

Mode of transport. For several modes use the parameter multiple times.

### **itdLPxx\_selOperator**

Messages which affect the services of a certain operator. The parameter value is the operator code. For several operators use the parameter multiple times.

# AddInfo-Request – Filters

## Filter for Network and Services

The following parameters can be used multiple times. For the supplement the character `_` needs to be replaced by a space.

### **itdLPxx\_selLine**

Diva line, e.g. `itdLPxx_selLine=6040D`.

### **filterPartialNet**

Messages which affect services that match the network.

### **filterPNLineSup**

Messages which affect services that match the network, DIVA line and supplement `<network>:<DIVA line>:<supplement>`, e.g. `filterPNLineSup=irl:6040D: .`

# AddInfo-Request – Filters

## Filter for Services

### **filterPNLineDir**

Messages which affect services that match the subnet, DIVA line and direction

<network>:<DIVA line >:<supplement>:<direction>, e.g. `filterLineDir=irl:6040D: :H`

Hint: H is inbound, R is outbound.

### **line**

Messages which affect services that match the subnet, DIVA line, supplement, direction and project <network>:<DIVA line >:<supplement>:<direction>:<project>, e.g.

`line=irl:6040D: :H:_`.

# AddInfo-Request – Filters

## Filter for Stops

### **passedStops = 1**

Messages which affect all passed stops of a service. See filter by service.

Hint: The selection of a service is required, e.g.

```
itdLPxx_selLine=6040D&passedStops=1.
```

### **itdLPxx\_selStop**

ID of a stop. The parameter can be used multiple times.

# AddInfo-Request – Filters

## Filter for Localities

### **filterOMC**

Messages which affect one or more municipalities determined by the OMC

<OMC>:<OMC>:....

### **filterOMC\_PlaceID**

Messages which affect a locality given by the OMC and place ID <OMC>:<place ID>. For more than one locality the parameter is used multiple times.

# AddInfo-Request – Filters

## Filter for Provider and Source

---

### **filterProviderCode**

Provider code. For messages of several providers use the parameter multiple times.

### **filterSourceSystemName**

Source system ID. For messages which have been entered to several source systems use the parameter multiple times.

```
  subtitle: DIVERSION ON R  
- properties: {  
  providerCode: "NTA",  
  - source: {  
    id: "ICSIRL",  
    name: "ICSIRL",  
    type: "MDVCMS",  
  },  
},  
- affected: {
```

# AddInfo-Request – Filters

## Filter for Message ID

### **filterInfoID**

ID of the message. For several messages use the parameter multiple times.

# AddInfo-Request – Additional Information / Reduction of the Response

Add required or remove not required elements from the output:

**filterShowLineList = 1 | 0**

Removes the list of affected lines.

**filterShowStopList = 1 | 0**

Removes the list of affected stops.

**filterShowPlaceList = 1 | 0**

Removes the list of affected localities.

# Journey-Request

# Journey-Request

1. Input and Output
2. Relevant Information for Monitoring

# Journey-Request – Input and Output Request

The Journey-Request finds all lines for journeys from an origin to a destination within a time interval. The passed stops are. Response data can be used as an input for JourneyCheck-Request.

## Request

```
http://osm.demo.mentz.net/training/XML_JOURNEY_REQUEST?commonMacro=journey
```

## Part of the Trip-Request

- Error Handling (as described per Common Functionality)
- Date and Time (as described per Common Functionality)
- Locality Input (as described per StopFinder-Request)
- Connection Options (as described per Trip-Request)

# Journey-Request – Input and Output

## Mandatory Parameters

### Parameters for the Interface

These parameters are given by parameter injection or configuration:

- `outputFormat=rapidJSON` (activates the JSON API)
- `coordOutputFormat=WGS84 [dd.ddddd]` (coord format set to WGS 84)

Note: Parameter injection works only for requests with HTTP parameters.

### Mandatory Parameters for Journey-Request

The Journey-Request requires an origin and destination input as described per StopFinder-Request. Additionally a time interval.

These parameters should be included in the HTTP parameter macro **commonMacro=journey**:

- `deleteAssignedStops_origin/deleteAssignedStops_destination` (no nearby stops)
- `genC=0, genP=0, genMaps=0` (prevents output of coordinate sequences, path descriptions and generation of additional pdf files)

This parameters are equal to Trip-Request's parameter thus the HTTP parameter

**commonMacro=trip** might be used.

# Journey-Request – Input and Output

## Mandatory Parameters

### **interval**

Search interval. The request calculates possible connections within the time interval given by a date (described par Common Functionality) and the search interval.

*Value:* Integer [minutes]

# Journey-Request – Input and Output

## Parameter Suffix for Locality Input

### Parameter Suffix for Locality Input

The parameter suffixes `<usage>` are `origin`, `destination` and `via`. The `via` location is optional.

### Challenge

You want to monitor the journey from the stop *Stuttgart Schwabstraße* to the stop *Stuttgart Feuersee* from 7:30 to 8:30.

Hint: Calculation takes only place if the origin and destination are identified.

# Journey-Request – Input and Output

## Parameter Suffix for Locality Input

### Solution

**Step 1: Determine unique IDs for origin and destination, e.g. by StopFinder-Request:**

```
http://osm.demo.mentz.net/training/XML_STOPFINDER_REQUEST?commonMacro=stopfinder&type_sf=any&name_sf=stuttgart schwabstraße
```

**Step 2: Calculation with the Journey-Request:**

```
http://osm.demo.mentz.net/training/XML_JOURNEY_REQUEST?commonMacro=trip&type_origin=any&name_origin=5006052&type_destination=any&name_destination=5006221&itdTime=0730&interval=60
```

# Journey-Request – Input and Output JSON Output

## Journeys

`journeys` is an array of connection options. Each journey option contains information about:

- the `service` (`transportation`)
- the `stopSequence`

## Services

`transportation` includes information about the lines. Especially the `id`, which is required for monitoring

## Passed Stops

`stopSequence` is an array of stops passed by the line. For monitoring required information are:

- the `id`
- the `MonitoredStop` flag, which is part of the `properties` and which indicates if the stop is required for monitoring

```
{
  version: "10.4.18.18",
  - journeys: [
    - {
      + transportation: {9},
      + stopSequence: [2],
      departureTime: "07:27:00",
      arrivalTime: "07:29:00",
    },
    - {
      + transportation: {9},
      + stopSequence: [11].
    }
  ]
}
```

```
- transportation: {
  id: "vvs:10006: :R:j22",
  name: "S-Bahn S6",
  disassembledName: "S6",
  number: "S6",
  description: "Stuttgart - Leonberg - Weil der Stadt",
  + product: {4},
  + operator: {3},
  + destination: {3},
  + properties: {8},
},
- stopSequence: [
  - {
    isGlobalId: true,
    id: "de:08111:6052:1:2",
    name: "Stuttgart Schwabstraße",
    disassembledName: "Gleis 2",
    type: "platform",
    pointType: "TRACK",
    + coord: [2],
    niveau: -300,
    + parent: {9},
    + productClasses: [3],
    - properties: {
      WheelchairAccess: "true",
      MonitorStop: "true",
      stoppingPointPlanned: "Gleis 2",
      areaGid: "de:08111:6052:1",
      area: "1",
      platform: "2",
      platformName: "Gleis 2",
    },
  },
  + {11},
]
```

# Journey-Request – Input and Output

## Relevant Information for Monitoring

To do monitoring with the JourneyCheck request, the following information is required:

- The ID of the line to be monitored
- The stops which mark the beginning and the end of the line segment to be monitored. These stops are marked with the flag `MonitoredStop=true`.

JourneyCheck-Request

1. Input and Output
2. Stop Areas and Stop Points

# Journey-CheckRequest – Input and Output Request

The Journey-Request monitors journeys. Journeys are given by a series of lines and for each line two passed stops which mark the beginning and end of the monitored segment.

## Request

`http://osm.demo.mentz.net/training/XML_JOURNEYCHECK_REQUEST?`

## Part of the Trip-Request

- Error Handling (as described per Common Functionality)

# JourneyCheck-Request – Input and Output

## Mandatory Parameters

**job<x>JobId = <x>**

Creates a monitoring job x with the ID x.

*Value:* Integer [Index, starting with 1]

**job<x>MO<y>Line**

Adds lines for monitoring to job x. The parameter can be used multiple times, in case a journey is compounded of more than one line. Each line has an index y, starting with 1.

*Value:* Line ID

**job<x>MO<y>Stop<z>Id** and **job<x>MO<y>Stop<z>Gid**

Add the two passed stops, which mark the start and the end point of the line segment y to be monitored, to job x. Use z=1 one for the first, 2 for the second stop.

*Value:* Stop ID for *job<x>MO<y>Stop<z>Id* and Global Stop ID for *job<x>MO<y>Stop<z>Gid*

# JourneyCheck-Request – Input and Output

## Mandatory Parameters

**job<x>MO<y>Date**

Date of the monitoring.

*Value:* Date [Format: YYYYMMDD]

**job<x>MO<y>DepTime** and **job<x>MO<y>ArrTime**

Time interval of the monitoring. “DepTime” marks the beginning, “ArrTime” the end of the interval.

*Value:* Time [Format: HHMM]

# JourneyCheck-Request – Input and Output

## Mandatory Parameters

### Example

[http://osm.demo.mentz.net/training/XML\\_JOURNEYCHECK\\_REQUEST?job1JobId=1&job1MO1Line=vvs:10006:R:j22&job1MO1Stop1Id=5006052&job1MO1Stop1Gid=de:08111:6052&job1MO1Stop2Id=5006221&job1MO1Stop2Gid=de:08111:6221&job1MO1Date=20221017&job1MO1DepTime=0700&job1MO1ArrTime=0800](http://osm.demo.mentz.net/training/XML_JOURNEYCHECK_REQUEST?job1JobId=1&job1MO1Line=vvs:10006:R:j22&job1MO1Stop1Id=5006052&job1MO1Stop1Gid=de:08111:6052&job1MO1Stop2Id=5006221&job1MO1Stop2Gid=de:08111:6221&job1MO1Date=20221017&job1MO1DepTime=0700&job1MO1ArrTime=0800)

```
journeys: [
  - {
    - transportation: {
      id: "vvs:10006:R:j22",
      name: "S-Bahn S6",
      disassembledName: "S6",
      number: "S6",
      description: "Stuttgart - Leonberg - Weil",
      + product: {4},
      + operator: {3},
      + destination: {3},
      + properties: {8},
    },
    - stopSequence: [
      - {
        isGlobalId: true,
        id: "de:08111:6052:1:2",
        name: "Stuttgart Schwabstraße",
        disassembledName: "Gleis 2",
        type: "platform",
        pointType: "TRACK",
        + coord: [2],
        niveau: -300,
        - parent: {
          isGlobalId: true,
          id: "de:08111:6052",
          name: "Stuttgart Schwabstraße",
          disassembledName: "Schwabstraße",
          type: "stop",
          + parent: {3},
          - properties: {
            stopId: "5006052"
          },
        },
        + coord: [2],
        niveau: 0,
      },
      + productClasses: [3],
      + properties: {7},
    ],
  },
  - {
    isGlobalId: true,
    id: "de:08111:6221:1:2",
    name: "Stuttgart Feuersee",
    disassembledName: "Gleis 2",
    type: "platform",
    pointType: "TRACK",
    + coord: [2],
    niveau: -2,
    - parent: {
      isGlobalId: true,
      id: "de:08111:6221",
      name: "Stuttgart Feuersee",
      disassembledName: "Feuersee",
      type: "stop",
      + parent: {3},
      - properties: {
        stopId: "5006221"
      },
    },
    + coord: [2],
    niveau: 0,
  },
  + productClasses: [3],
  + properties: {7},
],
departureTime: "07:27:00",
arrivalTime: "07:29:00",
},
```

For a more detailed stop specification use:

**job<x>MO<y>Stop<z>Area**

Specifies the stop area.

*Value:* Stop Area

**job<x>MO<y>Stop<z>Point** and **job<x>MO<y>Stop<z>GidPoint**

Specifies the stop point.

*Value:* Stop Point resp. Global Stop Point

StopList-Request

# StopList-Request

1. Input and Output
2. Filters
3. Additional Information

# StopList-Request – Input and Output Request

The StopList-Request is used to get the stops. Filtering is possible and recommended. Used for analysis and data export.

**Note: This request should not be part of a user web interface due to performance reasons. Filtering can help preventing a browser crash.**

## Request

`http://osm.demo.mentz.net/training/XML_STOPLIST_REQUEST?commonMacro=stoplist` -> **Do not enter this in your browser!**

## Example

`http://osm.demo.mentz.net/training/XML_STOPLIST_REQUEST?commonMacro=stoplist&stopListOMC=8111000`

# StopList-Request – Input and Output

## JSON Output / Mandatory Parameters

### Locations

The response includes an array of `locations`. The structure is equal to former examples.

### Mandatory Parameters

These parameters are given by parameter injection or configuration:

- `outputFormat=rapidJSON` (activates the JSON API)
- `coordOutputFormat=WGS84[dd.ddddd]` (coord format set to WGS 84)

Note: Parameter injection works only for requests with HTTP parameters.

Further customer specific parameters could be included in an HTTP parameter macro `commonMacro=stoplist`.

```
locations: [
- {
  isGlobalId: true,
  id: "de:08111:2",
  name: "WaldburgstraÃŸe",
  type: "stop",
- parent: {
  id: "51",
  name: "Stuttgart",
  type: "locality",
  - properties: {
    omc: "8111000"
  },
},
- properties: {
  stopId: "5000002"
},
- coord: [
  48.72546,
  9.1074,
],
},
+ {7},
+ {7},
+ {7},
```

# StopList-Request – Input and Output Filters

## **stopListOMC**

OMC (municipality code).

## **stopListPlaceId**

ID of the place. Can be combined with `stopListOMC`.

## **stopListOMCPlaceId**

Combination of `stopListOMC` and `stopListPlaceId`. OMC and ID of the place are separated by colon.

## **rTN**

Only stops within the network given by parameter value.

# StopList-Request – Input and Output Filters

## **stopListSubnetwork**

Only stops served by services from the network given by parameter value.

## **fromstop and tostop**

Only stops with IDs within the intervall restricted by these parameters.

# StopList-Request – Input and Output

## Additional Information

Please take in mind that requesting additional information worsens the performance.

**servicingLines = 1**

Services of each stop.

**servicingLinesMOTType = 1**

Mayor means of transport of each stop. The combination with `servicingLinesMOTTypes=1` is not possible.

**servicingLinesMOTTypes = 1**

All means of transport of each stop. Separated by comma. The combination with `servicingLinesMOTType=1` is not possible.

# StopList-Request – Input and Output

## Additional Information

**tariffZones = 1**

Tariff zone of each stop.

# LinkedList-Request

# LineList-Request – Table of Content

1. Input and Output
2. Optional Parameters

# LineList-Request – Input and Output

## Request

The LineList-Request is used to get the lines. Used for analysis and data export.

### Request

```
http://osm.demo.mentz.net/training/XML_LINELIST_REQUEST?commonMacro=linelist
```

### Example

```
http://osm.demo.mentz.net/training/XML_LINELIST_REQUEST?commonMacro=linelist&lineListSubnetwork=vvs
```

# LineList-Request – Input and Output

## JSON Output / Mandatory Parameters

---

### Transportation

The array `transportations` includes a list of services in the well known format.

### Mandatory Parameters

These parameters are given by parameter injection or configuration:

- `outputFormat=rapidJSON` (activates the JSON API)
- `coordOutputFormat=WGS84[dd.ddddd]` (coord format set to WGS 84)

Note: Parameter injection works only for requests with HTTP parameters.

Further customer specific parameters could be included in an HTTP parameter macro `commonMacro=linelist`.

```
transportations: [  
  - {  
    id: "vvs:21010: :H:j21",  
    name: "Zahnradbahn 10",  
    disassembledName: "10",  
    number: "10",  
    description: "Marienplatz - Degerloch (Zahnradbahn Zacke)",  
    + product: {4},  
    + operator: {2},  
    + destination: {2},  
    + properties: {5},  
  },  
  + {9},  
  + {9},  
]
```

# LineList-Request – Input and Output

## Mandatory Parameters

Use one of these parameters to search for lines:

**lineListBranchCode**

Code of the branch.

**lineListNetBranchCode**

Network and optionally the code of the branch separated by colon.

**lineListSubnetwork**

Network.

# LineList-Request – Optional Parameters

**lineListMixedLines = 1**

Activates the search of composed services.

**mergeDir = 1**

Merges the inbound and outbound service. Thus only inbound services are listed. By default both are listed.

**lineReqType**

Presentation type – works as a bit mask to combine presentation types Example: `lineReqType=5` -> `1 + 4`  
-> Departure Monitor and Timetable

**lineListOmc**

OMC (municipality code).

Value	Description
1	Departure Monitor (DM)
2	Stop Timetable (STT)
4	Timetable (TTB)
8	Route Maps
16	Station Timetable

StopZone-Request

# StopZone-Request – Table of Content

1. Input and Output
2. Optional Parameters

# StopZone-Request – Input and Output Request

The StopZone-Request is used to get tariff information of a stop, e.g. authority and tariff zone. It is required for the direct sale of tickets.

## Request

```
http://osm.demo.mentz.net/training/XML_STOPZONE_REQUEST?commonMacro=stopZone
```

## Example

```
http://osm.demo.mentz.net/training/XML_STOPZONE_REQUEST?commonMacro=stopZone&name_szi=de:08111:6118&type_szi=any&itdDate=20220802
```

# StopZone-Request – Input and Output Request

## Part of the StopZone-Request

- Locality Input (as described per StopFinder-Request)
- Data and Time (parameter `itdDate` only)

## Parameter Suffix for Locality Input

The parameter suffix `<usage>` for StopZone-Request is `szi`. Thus parameters are named `name_szi` and `type_szi`.

# StopZone-Request – Input and Output

## Mandatory Parameters

### Parameters for the Interface

These parameters are given by parameter injection or configuration:

- `outputFormat=rapidJSON` (activates the JSON API)
- `locationServerActive=1` (activates EFALocationServer for locality search)

Note: Parameter injection works only for requests with HTTP parameters.

### Mandatory Parameters for StopZone-Request

The StopZone-Request requires an **origin input** as described per StopFinder-Request and the a date given by the parameter **itdDate**.

Further customer specific parameters could be included in an HTTP parameter macro **commonMacro=stopZone**.

# StopZone-Request – Input and Output JSON Output

---

## Tariff Info

The array `zoneInfos` includes information about:

- `tariff authority`
- `tariff zones`

For some customer additional information is available:

- `priceLevel`
- `areaNumber (relation ID)`

```
stops: [
  - {
    id: "20005613",
    network: "vrr",
    - zoneInfos: [
      - {
        authority: "vrr",
        - zones: [
          "260"
        ],
        tariffArea: "26",
        pricelevel: "A2",
        areaNumber: "110110",
        intersectionArea: false,
      }
    ],
  }
],
```

# StopZone-Request – Optional Parameters

## **authority**

Filter for tariff authority.

## **priceLevel**

For a given price level the detailed price level is calculated and the relation ID is returned.

TicketDetails-Request

# TicketDetails-Request – Table of Content



1. Input and Output
2. Optional Parameters

# TicketDetails-Request – Input and Output

## Request

The TicketDetails-Request is used to get detailed information of a ticket. It is required for the direct sale of tickets.

### Request

```
http://osm.demo.mentz.net/training/XML_TICKETDETAILS_REQUEST?commonM  
acro=ticketDetails
```

### Example

```
http://osm.demo.mentz.net/training/XML_TICKETDETAILS_REQUEST?commonM  
acro=ticketDetails&date=20210419&tariff=vrr&ticketID=110570B&time=09  
31
```

# TicketDetails-Request – Input and Output

## Mandatory Parameters

### Parameters for the Interface

These parameters are given by parameter injection or configuration:

- `outputFormat=rapidJSON` (activates the JSON API)

Note: Parameter injection works only for requests with HTTP parameters.

### Mandatory Parameters for TicketDetails-Request

Customer specific parameters could be included in an HTTP parameter macro  
**`commonMacro=ticketDetails`**.

# TicketDetails-Request – Input and Output

## Mandatory Parameters



### **ticketID**

ID of the ticket.

### **tariff**

The tariff authority is required because the ticket ID is only unique within a tariff authority.

### **date**

Date.

*Format:* YYYYMMDD

### **time**

Time.

*Format:* HHMM

## **relationID**

The relation ID is only required if relation IDs are included in the tariff model. The parameter causes output of additional ticket information.

TariffScope-Request

# TariffScope-Request – Table of Content



1. Input and Output
2. Optional Parameters

# TariffScope-Request – Input and Output Request

The TariffScope-Request is used to get the tariff authorities and zones of a relation ID. It is an optional request for the direct sale of tickets. It is required to request geographical information of the relation ID to display it on an interactive map.

## Request

```
http://osm.demo.mentz.net/training/XML_TARIFFSCOPE_REQUEST?commonMacro=tariffScope
```

## Example

```
http://osm.demo.mentz.net/training/XML_TARIFFSCOPE_REQUEST?commonMacro=tariffScope&date=20210419&net=vrr&relationID=180072
```

# TariffScope-Request – Input and Output

## Mandatory Parameters

### Parameters for the Interface

These parameters are given by parameter injection or configuration:

- `outputFormat=rapidJSON` (activates the JSON API)

Note: Parameter injection works only for requests with HTTP parameters.

### Mandatory Parameters for TariffScope-Request

Customer specific parameters could be included in an HTTP parameter macro **`commonMacro=tariffScope`**.

# TariffScope-Request – Input and Output

## Mandatory Parameters

### **relationID**

Tariff authorities and zones are requested for this relationID.

### **date**

Date.

*Format:* YYYYMMDD

### **time**

Time.

*Format:* HHMM

## **net**

Filter for tariff authority. Only information for the given tariff authority is returned.

## **investigateRelationID**

Checks if a relation ID is part of the requested region ID (*regionID*).  
This parameter is used multiple times for multiple relation IDs.

## **destRegion / startRegion**

Required to highlight the start and destination zone in the result.

Explanatory Real Time (licence  
needed)

# Explanatory Real Time

## Description

The so called “Explanatory Real Time Feature” shows the users why they are not seeing their usual connections, but alternative connections instead (cancellation of a trip, interchange not possible due to a delay of the feeding service etc.).

## Input Parameter

add to the Trip-Request the parameter `realDesc=2`

## Output

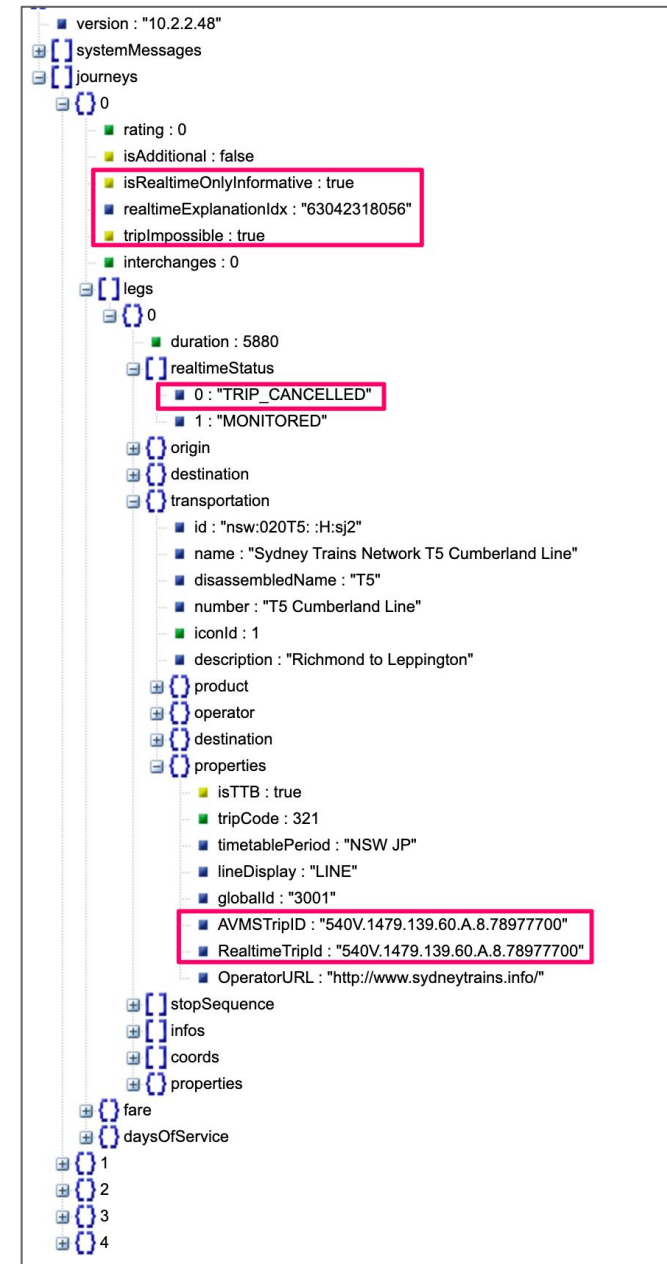
`isRealTimeOnlyInformative` - boolean

`tripImpossible` - boolean

`realtimeStatus` - enum:

“TRIP\_CANCELLED”, “EXTRA\_TRIP”, “EXTRA\_STOPS”,  
“DEVIATION”, “MONITORED”, “PROGNOSIS\_IMPOSSIBLE”,  
“OUTSIDE\_REALTIME\_WINDOW”,  
“REALTIME\_ONLY\_INFORMATIVE”

...



# Explanatory Real Time - Example from Gullivr App

## Example

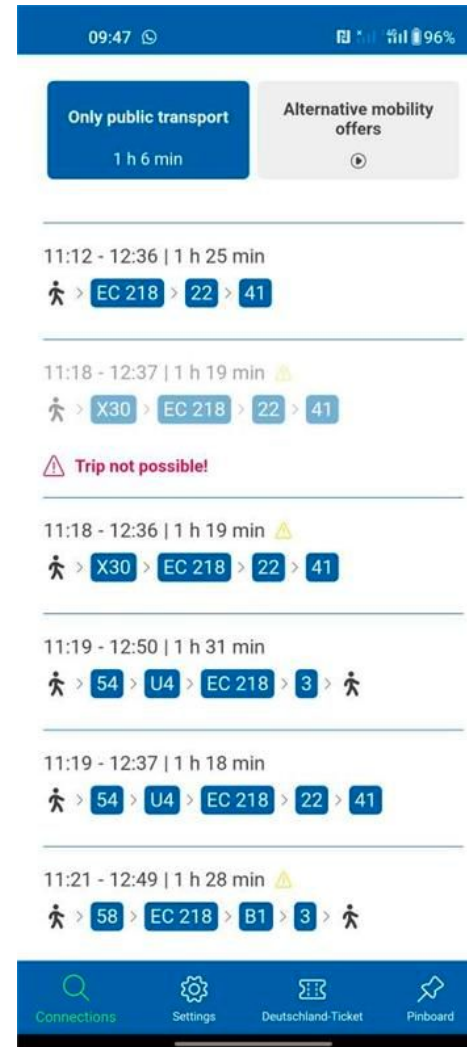
On the right, you can see an example where an interchange is not possible due to current real time situation (screenshots from our Gullivr App).

## Details

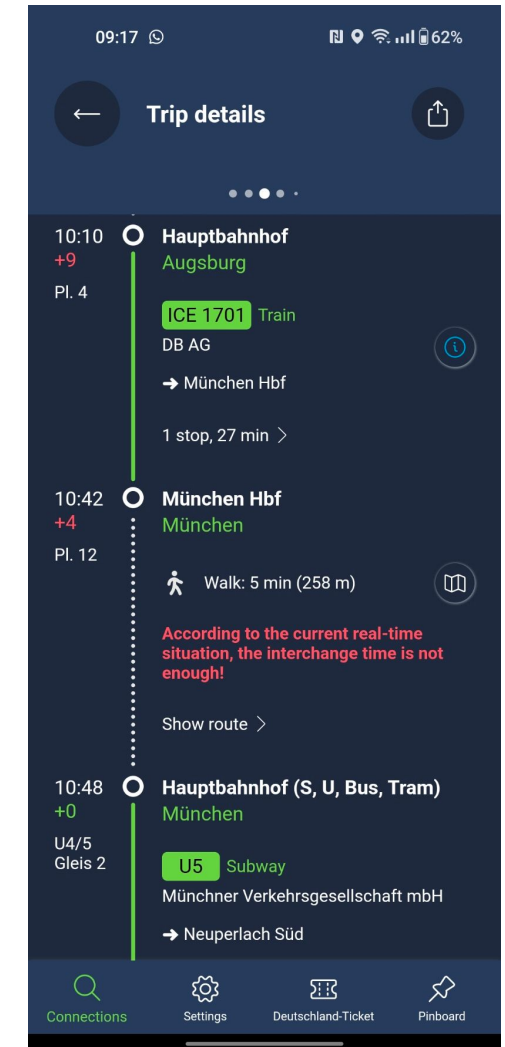
The train ICS 1701 arrives at Munich HBF with 4 minutes delay (planned arrival 10:42, estimated arrival 10:46). The interchange time to the connecting subway U4/5 is 5 minutes. As the subway departs at 10:48 there is not enough time for the 5 minutes interchange. The app highlights that by showing the customer: “According to the current real-time situation, the interchange time is not possible“

## What happens in the backend

Technically what happens in the background is that EFA does two calculations when the Parameter realDesc is active. The first one is the “normal” one considering real time and the second one triggers a calculation based on time table data only (no real time). By doing this EFA can compare the results and display the differences as this described here.



Trip results



Trip Detail View

# Explanatory Real Time - HTTP parameter options

Departure time according to schedule	status	Shown in API response with realDesc=0 (default)	Shown in API response with realDesc=1 (deprecated, do not use)	Shown in API response with realDesc=2	Shown in API response with realOnlyInfo=1
10:00	10:00	10:00	10:00	10:00	10:00
10:30	cancelled	-	10:30	10:30	10:30
11:00	cancelled	-	-	11:00	11:00
11:30	cancelled	-	-	11:30	11:30
12:00	12:00	12:00	12:00	12:00	-
12:30	12:30	12:30	12:30	12:30	-
13:00	13:00	13:00	13:00	13:00	-

# Additional remarks

The above shown use cases for cancellations can be either achieved

- by getting cancelled trips via the real time interfaces, and
- by means of blocking messages within EMS
  - either as an EMS agent, or
  - via imported disruptions alerts.

VMI-API

Vehicle location map interface  
(licence needed)

The VMI interface is an http/JSON interface. It is offered by the software systems DDIP and VISFeed.

The purpose of this interface is to deliver position information about all running vehicles. The information can for example be requested by a map which displays the information on an additional map layer.

Parameter Name	Value Type	Description
Operator	string	Operator filter. Journeys of other operators are filtered out
MOTCode	string (i.e. 1,2,3,4). Transport codes have to be separated by comma.	MOT code filter. Journeys with other MOT (Mean of transport) codes are filtered out.
JourneyIdentifier	string	JourneyIdentifier filter. Journeys with different keys are filtered out.

Parameter Name	Value Type	Description
MinX	int	Min X-Coord value of the filter rectangle
MaxX	int	Max X-Coord value of the filter rectangle
MinY	int	Min Y-Coord value of the filter rectangle
MaxY	int	Max Y-Coord value of the filter rectangle
CoordSystem	string (i.e. WGS84 - default, WG10)	Coordinate system of the filter rectangle coords. Also, the coordinates in the response will be transformed to this system.
ID	string	ID filter. If journey with this ID exists, returns a list with this single journey. If not, returns an empty list.

# VMI - Response if no ID was given as request parameter

Field Name	Value Type	Description	Meaning if not set
CurrentStop	string	ID of the current stop of the journey	No information about current stop available.
DayOfOperation	string	The day of operation	(always set)
Delay	int	Current delay in seconds	Delay unknown
DirectionText	string	Direction text	No direction text available.
ID	int	ID of the journey (may be used as parameter for further requests)	(always set)
JourneyIdentifier	int	external journey identifier	No external journey identifier given.

# VMI - Response if no ID was given as request parameter

Field Name	Value Type	Description	Meaning if not set
LineText	string	Text description of line, e.g. "S2"	No LineText available.
MOTCode	string	MOT code	MOTCode not available.
NextStop	string	ID of the next stop of the journey	No information about next stop available.
ProductIdentifier	string	Product ID type	No Product ID type available.
RealtimeAvailable	0/1	1 = available, 0 = not available	0, no realtime available
Timestamp	string	Timestamp of this journey's data.	(always set)
TimestampPrevious	string	Timestamp of the previous position	No previous position available. This is the first time a journey is reported.

# VMI - Response if no ID was given as request parameter

Field Name	Value Type	Description	Meaning if not set
X	double	X coordinate of the current position	No coordinate available for this journey.
XPrevious	double	X coordinate of the previous position	No previous position available. This is the first time a journey is reported.
Y	double	Y coordinate of the current position	No coordinate available for this journey.
YPrevious	double	Y coordinate of the previous position	No previous position available. This is the first time a journey is reported.

# VMI - Response if no ID was given as request parameter

```
{
  "ID": "defac1d2-1f51-4ad9-be45-94fa423a7ad1",
  "Operator": "DAN",
  "JourneyIdentifier": "11",
  "DayOfOperation": "24.03.2025",
  "Delay": 0,
  "MOTCode": 5,
  "X": "9.392079",
  "Y": "48.868526",
  "Timestamp": "2025-03-24T14:57:06+01:00",
  "XPrevious": "9.393186",
  "YPrevious": "48.863284",
  "TimestampPrevious": "2025-03-24T14:56:52+01:00",
  "RealtimeAvailable": 1,
  "LineText": "Bus 339",
  "LineNumber": "339",
  "CurrentStop": "5003642#0#3",
  "NextStop": "5006983#0#3",
  "DirectionText": "Winnenden Rems-Murr-Klinikum"
},
```

# VMI - Response if ID is given as request parameter

Field Name	Value Type	Description
RealtimeAvailable	0/1	0: Planned 1: Estimated
Delay	int	Current delay in seconds
DayOfOperation	string	The day of operation
IsAtStop	0/1	0: false 1: true
Distance	int	Distance to next stop (in meter)
JourneyIdentifier	int	External journey identifier
LineText	string	Display line text
DirectionText	string	Display direction text
CurrentStop	string	ID of the current (or last) stop

# VMI - Response If ID is given as request parameter

Field Name	Value Type	Description
NextStop	string	ID of the next stop
Operator	string	Operator name of this journey
ProductIdentifier	string	Product ID
MOTCode	int	Public transport type coded as follows:  <pre>TRAIN = 0; COMMUTERTRAIN = 1; UNDERGROUND = 2; URBANRAIL = 3; TRAM = 4; URBANBUS = 5; REGIONALBUS = 6; EXPRESSBUS = 7; FUNICULAR = 8; SHIP = 9; DEMANDBUS = 10; OTHER = 11; AIRPLANE = 12; TRAIN_LOCAL = 13; TRAIN_NATIONAL = 14; TRAIN_INTERNATIONAL = 15; TRAIN_HIGHSPEED = 16; RAIL_REPLACEMENT = 17; RAIL_SHUTTLE = 18;</pre>

# VMI - Response If ID is given as request parameter

```
[
  {
    "ID": "d83af0b9-3fbe-4789-b467-3805251a1a83",
    "JourneyIdentifier": "hmv:31625:_:H:j25:96",
    "DayOfOperation": "24.03.2025",
    "Delay": 300,
    "MOTCode": 5,
    "X": "9.331078",
    "Y": "49.237111",
    "Timestamp": "2025-03-24T15:13:17+01:00",
    "XPrevious": "9.331096",
    "YPrevious": "49.237073",
    "TimestampPrevious": "2025-03-24T15:12:53+01:00",
    "RealtimeAvailable": 1,
    "LineText": "Bus 625",
    "LineNumber": "625",
    "CurrentStop": "5400561#0#>Ost",
    "NextStop": "5400547#8#8",
    "DirectionText": "Neuenstadt über Oedheim"
  }
]
```

# Vielen Dank!

MENTZ

Mehr über uns erfahren Sie auf: [mentz.net](http://mentz.net)

---

MENTZ GmbH  
Grillparzerstraße 18  
81675 München

vertreten durch: Christoph Mentz, Geschäftsführer

---

[info@mentz.net](mailto:info@mentz.net)  
+49 89 41868-0