HaCon

Traffic • Software • Service

HAFAS ReST API

Access to HAFAS journey planner systems

Version 1.23.23

22. 06. 2018

Michael Frankfurter
HaCon Ingenieurgesellschaft mbH
Lister Straße 15
30163 Hannover
Germany

Contents

HAFAS ReST API is a software solution of HaCon and will be continuously improved, thus content of the document and written realisation of features may change without further notice.

1 The Interface

1.1 Introduction

1.1.1 Interface Overview

The public interface is implemented as a ReST[1] (**Re**presentational **S**tate **T**ransfer) interface which provides different methods for the different functionalities of the journey planner, which are the following services:

- Location services
  - Location name
  - Nearby
- Trip
  - Scroll
- Interval trip search
- Reconstruction
- GIS Route
- Station board services:
  - DepartureBoard
  - ArrivalBoard
- JourneyDetail
- JourneyMatch
- Train Search
- Print to web gateway
- Real time archive gateway
- Time table info
- XSD
- Status

While Location, Trip, Interval, ArrivalBoard and DepartureBoard services can be called directly, the JourneyDetail and Scroll services can only be called by a reference given in a re-

[1] See http://rest.elkstein.org/ for a tutorial on ReST interfaces.

sult of the Trip, DepartureBoard or ArrivalBoard service. The Reconstruction service can only be called by a reference given in a result from a Trip request or other means. The XSD service can be called directly to download the XSD files with response specification of a certain service. The same is true for the Status service.

The system implements read-only GET requests which are called by given service URLs and multiple GET parameters to specify the requested journey planner information. The parameter values need to be UTF-8 URL encoded. The result of each request will be delivered either as XML or JSON response. If the URL parameter encoding is not correct, the behaviour of the system might deliver unexpected results.

From now on it is assumed, that you have been provided with a base URL of the HAFAS system. The following documentation of the different requests been described based on this given base URL <*baseurl*>.

1.2 General principles

There are some general principles which are valid for the different services which are described in this section.

1.2.1 Coordinates

Coordinates are always in the WGS84 system, represented as decimal degrees in the interval -90 to 90 for the latitude (lat) and -180 to 180 for the longitude (long).

1.2.2 Date and time formats

Dates are always represented in the format YYYY-MM-DD. This applies both for request parameters as for dates in responses. Times are always represented in the format hh:mm[:ss] in 24h nomenclature. Input of seconds is optional. Please note that seconds are ignored by the underlying HAFAS system, i.e. a departure time specified as 14:37:52, for example, and is interpreted as 14:37:00.

1.2.3 Stateless service vs. data dependency

All services of the provided interface are stateless as it is required for a ReST protocol. But this has its limitation concerning the journey planner's timetable data. As soon as the timetable data is exchanged (in most cases daily on weekdays), IDs of stops/stations are not necessary valid anymore. The same applies for reference URLs provided by the Trip service to retrieve JourneyDetails. The storage of stop/station IDs and reference URLs to JourneyDetails for a longer period except the current user session is not recommended. Any usage of these IDs or URLs beyond the lifetime of the current session is on your own risk and might cause undetermined behaviour.

1.2.4 Route index

A route is the list of stops/stations where a vehicle like a train or bus stops. Every stop/station on a route has its own index which can be used as a reference. This index is also used to identify distinctively if the same stop/station if it is contained several times in one route.

1.2.5 Real-time information

Real-time information will be included in the service as far as it is. It is always delivered in addition to the planned departures and arrivals.

1.2.6 Versioning

Due to enhancements of the API the input parameters and the results can change over time. Different Versions of the API will be available at the same time.

The requested version can be specified by using the version number in the path info:

http://<baseUrl>/<version>/<servicename>

The version part is optional. If it is omitted, the latest version will be used. Be aware that omitting the version can break your client when a new API version is introduced. If your client always requires a special version of the API (v2 for example), your URL would look like this: http://<baseUrl>/v2/<servicename>

1.2.7 Response Format

The interface returns responses either in XML (default) or JSON format.

If XML is requested, the response will have the namespace **`hafas_rest_v1`**.

In order to request a JSON response you have to append the following parameter to each call of the interface: **`format=json`**. If JSONP is needed you can append an additional parameter to specify the name of callback function, the JSON object will be wrapped by a function call with this name: **`jsonpCallback=mycallback.`**

The JSON content is generated by converting the XML content to JSON automatically. The conversion is done by the following simple rules:

- Element names become object properties
- Text (PCDATA) becomes an object property with name "$"
  <a>foo becomes { "a": { "$" : "foo" } }
- Nested elements become nested properties
  <a>foo<c>foo</c>
  becomes
  { "a": { "b" : { "$": "foo" }, "c": { "$": "foo"} } }
- If there are multiple elements with the same name, the JSON code contains an array for these elements.
  <a>foo1foo2

becomes
{ "a": { "b" : [{"$": foo1" }, {"$": "foo2" }] } }

- Attribute names become object properties
  foo2
  becomes
  { "a": { "atb" : "foo1", "$" : "foo2" } }

The following example shows a trip in XML response and the resulting conversion to JSON:

XML:

```
<Trip xmlns="hafas_rest_v1">
      <Leg name="Expressbuss 830" type="LOC" id="830"
            direction="Göteborg Nils Ericsonterminal">
      <Origin name="Stockholm Cityterminalen" type="ST" id="7400622"
            routeIdx="0" time="08:05" date="2011-12-18" />
      <Destination name="Göteborg Nils Ericsonterminal" type="ST"
            id="7420483" routeIdx="12" time="15:25"
            date="2011-12-18" />
      </Leg>
</Trip>
```

JSON:

```
"Trip": {
      "Leg": {
            "name": "Expressbuss 830",
            "type": "LOC",
            "id": "830",
            "direction": "Göteborg Nils Ericsonterminal",
            "Origin": { "name": "Stockholm Cityterminalen",
                  "type": "ST", "id": "7400622", "routeIdx": "0",
                  "time": "08:05", "date": "2011-12-18" },
            "Destination": { "name": "Göteborg Nils Ericsonterminal",
                  "type": "ST", "id": "7420483", "routeIdx": "12",
                  "time": "15:25", "date": "2011-12-18" }
      }
}
```

1.2.8 Authentication

Every client using the API needs to pass a valid authentication key in every request.

The following parameter has to be appended to the URL: `accessId=<your_key_here>`.

Please contact the operating company in order to request an authentication key.

1.2.9 Languages

The journey planer supports multiple languages. The language can be specified by the optional URL parameter `lang=<code>`. The default language is defined by the underlying HAFAS system is used if no language parameter is delivered. The language code has to be lower case.

The supported languages depend on the plan data of the HAFAS system.

The chosen language only influences the returned Notes in the ReST responses.

Code	Language	Code	Language	Code	Language
de	German	fr	French	no	Norwegian
da	Danish	hu	Hungarian	pl	Polish
en	English	it	Italian	sv	Swedish
es	Spanish	nl	Dutch	tr	Turkish

1.2.10 Request tracking

If the request tracking is enabled in your installation, you can add the parameter requestId to all of your services. The value will occure in any log as well as in the response like this:

/trip?...&requestId=123456789

```
<?xml version="1.0" encoding="UTF-8"?>
<TripList serverVersion="1.5" dialectVersion="1.23" requestId="123456789"
xmlns="hafas_rest">
...
</TripList>
```

If no requestId is provided, an installation wide unique one is created and added to the logs and response.

1.2.11 Barrier free information

Barrier free information is provided as codes. The following table shows possible values and their meaning.

Code	Description
baim_1	Boarding and alighting possible. Vehicle and stop are fully accessible for people having walking disabilities.
baim_2	Boarding and alighting possible with the help of the crew. Vehicle, stop or both are limited accessible for people having

	walking disabilities.
baim_3	Boarding and alighting requires advance notification. Vehicle, stop or both are limited accessible for people having walking disabilities.
baim_4	Boarding and alighting not possible. Vehicle, stop or both are not accessible for people having walking disabilities.
baim_info	Informational note needs to be presented.
baim_rsv	Reservation note needs to be presented.

1.2.12 Capacity information

Capacity information is provided as codes. The following table shows possible values and there meaning.

Code	Description
cap1st_10	No information available for first class.
cap1st_11	First class low to moderate capacity utilization expected.
cap1st_12	First class high capacity utilization expected.
cap1st_13	First class very high capacity utilization expected.
cap2nd_10	No information available for second class.
cap2nd_11	Second class low to moderate capacity utilization expected.
cap2nd_12	Second class high capacity utilization expected.
cap2nd_13	Second class very high capacity utilization expected.

2 Services

The list of services below describes all services provided by the HAFAS ReST API. If a service is available in your installation depends on the package you licenced.

Some services take parameters which depend on customer specific HAFAS server settings, like predefined filters. You need to ckeck your delivery package notes for those if any.

2.1 Service overview

The service overview will provide a list of all services available via the proxy. Each list item is clickable and will lead to a WADL of the service choosen.

Request: `<baseurl>/`

2.2 Service description

Each service endpoint provides an online self-description in the form of a WADL file. They are available through the Service overview or using this scheme:
`<baseurl>/service?wadl`

Example to get the description for the Trip search service: `<baseurl>/trip?wadl`

2.3 Location Service

There are two different types of the location service which can be used to get a list of locations using different input parameters.

The root element for this response is `LocationList` (see also 3.1 for further details).

2.3.1 Stop weight

Each station or stop is assigned a weight value which indicates how “busy” this station is. The higher the value, the more “busy” the station is.

The calculation is based on the product classes. For each product class operating at this stop, the frequency how often this product class operates is rated between 0 and 3 where 0 means this product isn’t operating and 3 means that this product operates at a high frequency. Then an individual weight is calculated for product class by multiplying the frequency rating with a certain factor. These factors take into account that traffic by trains for example weighs higher than traffic by busses. The weight for the station is then the sum over all individual weights for each product class.

2.3.2 Location.name Service

The `location.name` service can be used to perform a pattern matching of a user input and to retrieve a list of possible matches in the journey planner database. Possible matches might be stops/stations, points of interest and addresses. For reasons of backward compatibility the service name `location` can be used as an alias for `location.name`.

The result is a list of possible matches (locations) where the user might pick one entry to perform a trip request with this location as origin or destination or to ask for a departure board or arrival board of this location (stops/stations only)

2.3.2.1 Request Parameters

Name	Use	Range	Default	Description
accessId	Mandatory	-	-	Access ID for identifying the requesting client.
input	Mandatory	-	-	Search for that token
maxNo	Optional	1-1000	10	Maximum number of returned stops
type	Optional		ALL	Type filter for location types: ALL: search in all existing location pools S: Search for stations only A: Search for addresses only P: Search for POI's only SA: Search for stations and addresses SP: search for stations and POIs AP: search for addresses and pois
products	Optional	-	-	Decimal value defining the product classes to be included in the search. It represents a bitmask combining bit number of a product as defined in the HAFAS raw data file zugart. If, for example, you would like to search for local and regional trains only, and your HAFAS raw data file *zugart* states regional trains are product class 2 and local trains are class 3, you need a bit-mask where bits 2 and 3 are set. Calculation is then: 2^2 + 2^3 = 12 which would be the parameter value for "products".
coordLong	Optional	See 1.2.1	-	Longitude of centre coordinate
coordLat	Optional	See 1.2.1	-	Latitude of centre coordinate
r	Optional	1-10000	1000	Search radius around the given coordinate in meter.
refineId	Optional	-	-	In case of an refinable location, this value takes the ID of the refinable one of a previous result.

2.3.2.2 Adding a question mark at the end of the input to receive more results

Although you can specify the maxNo parameter to define how many results you would like to receive, there are cases when you receive much less results. This is the case if the pattern matching found a 100% match with the input string. Then no more results are returned because the algorithm assumes that this result is all the user expects.
This is the default behavior but in many cases it may be of interest to see the other results which may not represent a 100% match but are still quite close. To enable this output, add a question mark "?" at the end of the input string. Then the service will also return the other results up to the specified maximum number of results.

2.3.2.3 Usage of coordinates (coordLong and coordLat)

If coordLong and coordLat are provided, the pattern matching is done in the respective area. Depending upon the setup of your HAFAS system, a selective or a highlighting filter is used. In case of a selective filter, only locations in the specified area are returned. In case of a highlighting filter, the pattern matching score of locations in the specified area is increased but other locations outside that region will also be returned if those locations have a high pattern matching score.

Please note that a location.name request that comprises coordinates is computation-intensive and should be used seldom if at all.

Please refer to your project manager to discuss your options for this service as well as alternative solutions that could support your use case.

If you want to find every stop or POI around a center coordinate, please consider the Location.nearbystops service (see section 2.3.3).

2.3.2.4 Refinable Locations

Some locations in a result might not be fully resolved and so are refinable. This is indicated by the attribute **`refinable`** being true. To ease of use, the service **`link`** for the refinement is created along this result in the **`links`** section:

```
<CoordLocation name="1716 Schwarzsee, Ättenbergstrasse" type="ADR"
id="A=2@O=1716 Schwarzsee, Ättenbergstras-
se@X=7307502@Y=46683563@U=103@b=990017935@B=1@p=1413523432@"
lon="7.307502" lat="46.683563" refinable="true">
  <links>
    <link rel="refine"
href="http://demo.hafas.de/sbb/restproxy/location.name?input=1716
Schwarzsee, Ättenbergstrasse&refineId=A=2@O=1716 Schwarzsee, Ätten-
bergstras-
se@X=7307502@Y=46683563@U=103@b=990017935@B=1@p=1413523432@&type=A"/
>
  </links>
</CoordLocation>
```

In the case you want to create the refinement link on your own, you need to do the following:

- Put the name of the refinable location into the `input` parameter

- Put the id of the refinable location into the `refineId` parameter
- Set the `type` parameter accordingly
    - S in case of an station
    - A in case of a location
    - P in case of a POI

2.3.2.5 Example

Request:

```
<baseurl>/location.name?input=oslo
```

Result:

```
<LocationList xmlns="hafas_rest_v1">
    <StopLocation id="A=1@O=Oslo S@X=10755332@Y=59910200@U=70
        @L=007600100@B=1@p=1400139960@" name="Oslo S"
        lon="10.755332" lat="59.9102"/>
    <StopLocation id="A=1@O=Oslo S- avst@X=10712157@Y=59877623@U=70
        @L=000100124@B=1@p=1400139960@" name="Oslo S- avst"
        lon="10.712157" lat="59.877623"/>
    <StopLocation id="A=1@O=Oslo Lufthavn@X=11096913@Y=60193280@U=70
        @L=007600220@B=1@p=1400139960@" name="Oslo Lufthavn"
        lon="11.096913" lat="60.19328"/>
    <CoordLocation name="Oslo," type="ADR" lon="10.542252"
        lat="60.151588"/>
    <CoordLocation name="Oslo, Dråga" type="ADR" lon="10.790768"
        lat="59.897678"/>
    <CoordLocation name="Oslo, Bøgata" type="ADR" lon="10.781068"
        lat="59.912933"/>
    <CoordLocation name="Oslo, Grinda" type="ADR" lon="10.75126"
        lat="59.965241"/>
</LocationList>
```

2.3.3 Location.nearbystops Service

The `location.nearbystops` service returns a list of stops around a given center coordinate (within a radius of 1000m). The returned results are ordered by their distance to the centre coordinate.

2.3.3.1 Request Parameters

Name	Use	Range	Default	Description
accessId	Mandatory	-	-	Access ID for identifying the requesting client.

Name	Use	Range	Default	Description
originCoordLat	Mandatory	See 1.2.1	-	Latitude of centre coordinate
originCoord-Long	Mandatory	See 1.2.1	-	Longitude of centre coordinate
r	Optional	1-10000	1000	Search radius around the given coordinate in meter.
maxNo	Optional	1-1000	10	Maximum number of returned stops
type	Optional		S	Type filter for location types: S: Search for stations only P: Search for POI's only SP: search for stations and POIs
products	Optional	-	-	Decimal value defining the product classes to be included in the search. It represents a bitmask combining bit number of a product as defined in the HAFAS raw data file zugart. If, for example, you would like to search for local and regional trains only, and your HAFAS raw data file *zugart* states regional trains are product class 2 and local trains are class 3, you need a bitmask where bits 2 and 3 are set. Calculation is then: 2^2 + 2^3 = 12 which would be the parameter value for "products".

2.3.3.2 Example

Request: Search for stations around the coordinate

```
<baseurl>/location.nearbystops?originCoordLong=10.755332&
originCoordLat=59.9100200&maxNo=2
```

Result:

```
<LocationList xmlns="hafas_rest_v1">
    <StopLocation id="A=1@O=Oslo S@X=10755332@Y=59910200@u=0@U=70
        @L=7600100@" name="Oslo S" lon="10.755332" lat="59.9102"/>
```

```
    <StopLocation id="A=1@O=Nydalen st@X=10758982@Y=59915405@u=0
        @U=70@L=7621273@" name="Nydalen st" lon="10.758982"
        lat="59.915405"/>
</LocationList>
```

2.4 Trip service

The `trip` service calculates a trip from a specified origin to a specified destination. These might be stop/station IDs or coordinates based on addresses and points of interest validated by the location service or coordinates freely defined by the client.

2.4.1 Request Parameters

Name	Use	Range	Default	Description
accessId	Mandatory	-	-	Access ID for identifying the requesting client.
originId	Mandatory if originExtId and coordinate aren't specified	See 2.3.2 or 2.3.3	-	Specifies the station/stop ID (location reconstruction context) of the origin for the trip. Such ID can be retrieved from the location.name or location.nearbystops services.
originExtId	Mandatory if originId and coordinate aren't specified			Specifies the external station/stop ID of the origin for the trip. Such ID can be retrieved from the location.name or location.nearbystops services
originCoordLat	Mandatory if ID isn't specified	See 1.2.1 and 2.3.2 or 2.3.3	-	Latitude of station/stop coordinate of the trip's origin. The coordinate can be retrieved from the location.name or location.nearbystops services.
originCoordLong	Mandatory if ID isn't specified	See 1.2.1 and 2.3.2 or 2.3.3	-	Longitude of station/stop coordinate of the trip's origin. The coordinate can be retrieved from the location.name or location.nearbystops services.

destId	Mandatory if destExtId and coordinate aren't specified	See 2.3.2 or 2.3.3	-	Specifies the station/stop ID (location reconstruction context) of the destination for the trip. Such ID can be retrieved from the location.name or location.nearbystops services.
destExtId	Mandatory if destId and coordinate aren't specified			Specifies the external station/stop ID of the destination for the trip. Such ID can be retrieved from the location.name or location.nearbystops services
destCoordLat	Mandatory if ID isn't specified	See 1.2.1 and 2.3.2 or 2.3.3	-	Latitude of station/stop coordinate of the trip's destinationn. The coordinate can be retrieved from the location.name or location.nearbystops services.
destCoordLong	Mandatory if ID isn't specified	See 1.2.1 and 2.3.2 or 2.3.3	-	Longitude of station/stop coordinate of the trip's destination. The coordinate can be retrieved from the location.name or location.nearbystops services.

via	Optional	-	-	Complex structure to define multiple vias and parameters. Multiple vias are separated by ; This structure is build like this: viaId|waittime|viastatus|products *viaId*: id or extId of the via, mandatory *waittime*: waiting time spent at via station in minutes, optional *viastatus*: one of EXR (boarding and alighting nececessary), NER (boarding not necessary), NXR (alighting not necessary), NEXR (boarding and alighting not necessary), optional but defaults to EXR *products*: products used at the via, optional Example 1: Just define three vias to be passed by extId: via=801234;801235;801236 Example 2: Two vias having a wait time of 10 and 20 minutes: via=801234|10;801235|20 Example 3: One via without waittime but NEXR: via=801234||NEXR
viaId	Optional	See 2.3.2 or 2.3.3		ID of a station/stop used as a via for the trip. Specifying a via station forces the trip search to look for trips which must pass through this station. Such IDs can be retrieved from the location.name or location.nearbystops services. If via is used, viaId and viaWaitTime are having no effect.
viaWaitTime	Optional	See 1.2.2	0	Defines the waiting time spent at via station in minutes. If via is used, viaId and viaWaitTime are having no effect.

avoid	Optional	-	-	Complex structure to define multiple avoids and parameters. Multiple avoids are separated by ; This structure is build like this: avoidId|avoidstatus *avoidId*: id or extId of the avoid, mandatory *avoidstatus*: one of NPAVM (do not run through if this is a meta station), NPAVO (do not run through), NCAVM (do not change if this is a meta station), NCAVO (do not change), optional but defaults to NCAVM Example: Just define three avoids by extId: avoid=801234;801235;801236
avoidId	Optional	See 2.3.2 or 2.3.3		ID of a station/stop to be avoided as transfer stop for the trip. Such IDs can be retrieved from the location.name or location.nearbystops services. If avoid is used, avoidId has no effect.
changeTimePercent	Optional	0 - 500	100	Configures the walking speed when changing from one leg of the journey to the next one. It extends the time required for changes by a specified percentage. A value of 200 doubles the change time as initially calculated by the system. In the respones, change time is presented in full minutes. If the calculation based on changeTimePercent does not result in a full minute, it is rounded using “round half up” method.
minChangeTime	Optional	-	-	Minimum change time at stop in minutes.
maxChangeTime	Optional	-	-	Maximum change time at stop in minutes.

addChangeTime	Optional	-	-	This amount of minutes is added to the change time at each stop.
maxChange	Optional	0-11		Max no of changes.
date	Optional	See 1.2.2	Current server date	Sets the departure date for the search.
time	Optional	See 1.2.2	Current server time	Sets the departure time for the search.
searchForArrival	Optional	0 or 1	0	If set, the date and time parameters specify the arrival time for the trip search instead of the departure time.
numF	Optional	0 to 6	5	Minimum number of trips after the search time. Sum of **numF** and **numB** has to be less or equal 6. Please see the section below about the search algorithm for more details.
numB	Optional	0 to 6	0	Minimum number of trips before the search time. Sum of **numF** and **numB** has to be less or equal 6. Please see the section below about the search algorithm for more details.
products	Optional	-	-	Decimal value defining the product classes to be included in the search. It represents a bitmask combining bit number of a product as defined in the HAFAS raw data file *zugart*. If, for example, you would like to search for local and regional trains only, and your HAFAS raw data file *zugart* states regional trains are product class 2 and local trains are class 3, you need a bitmask where bits 2 and 3 are set. Calculation is then: 2^2 + 2^3 = 12 which would be the parameter value for "products".

context	Optional	See 2.4	-	Defines the starting point for the scroll back or forth operation. Use the scrB value from a previous result to scroll backwards in time and use the scrF value to scroll forth.
poly	Optional	0 or 1	0	Enables/disables the calculation of the polyline for each leg of the trip.
passlist	Optional	0 or 1	0	Enables/disables the return of the passlist for each leg of the trip.
operators	Optional	All operator codes or names from HAFAS raw data file *betrieb*.	-	Only trips provided by the given operators are part of the result. To filter multiple operators, separate the codes by comma. If the operator should not be part of the trip, negate it by putting ! in front of it. E.g. filter for A and B operator: operators=A,B.
attributes	Optional	All attribute codes from HAFAS raw data.	-	Filter trips by one or more attribute codes of a journey. Multiple attribute codes are separated by comma. If the attribute should not be part of the trip, negate it by putting ! in front of it.
sattributes	Optional	All station attribute codes from HAFAS raw data.	-	Filter trips by one or more station attribute codes of a journey. Multiple attribute codes are separated by comma. If the attribute should not be part of the trip, negate it by putting ! in front of it.
lines	Optional	-	-	Only journeys running the given line are part of the result. To filter multiple lines, separate the codes by comma. If the line should not be part of the trip, negate it by putting ! in front of it. This filter needs extended line data of HAFAS 5.40 in the back end.

avoidPaths	Optional	One or more codes	-	Only path not having the given properties will be part of the result. Possible codes are Stairway SW Elevator EA Escalator ES Ramp RA Convey Belt CB E.g. use paths without ramp and stairway: avoidPaths=SW,RA. Please note: Attribute codes may vary in your installation.
originWalk	Optional	0 or 1	1	Enables/disables using footpaths in the beginning of a trip when searching from an address. To fine-tune the minimum and/or maximum distance to the next public transport station, provide these values separted by comma. These values are expressed in meters. Samples To enable walk, minimum distance should be zero meters, maximum distance should be 1000 meters set the parameter `originWalk=1,0,1000` If the default distance should be used, just put no value, e.g `1,,1500` to have walk enabled, default minimum and 1500 meters as maximum. **Other possible settings are** **Speed** < 100: faster = 100: normal (default) > 100: slower **Bee line calculation** 0 (default) or 1

originBike	Optional	0 or 1	1	Enables/disables using bike routes in the beginning of a trip when searching from an address. To fine-tune the minimum and/or maximum distance to the next public transport station or mode change point, provide these values separted by comma. These values are expressed in meters. Samples To enable bike, minimum distance should be zero meters, maximum distance should be 1000 meters set the parameter `originBike=1,0,1000`. If the default distance should be used, just put no value, e.g `1,,1500` to have bike enabled, default minimum and 1500 meters as maximum. **Other possible settings are** **Speed** < 100: faster = 100: normal (default) > 100: slower **Bee line calculation** 0 (default) or 1 **Vehicle mode** sharing, self (default) **Provider** enabled providers for vehicle mode, e.g. callabike, etc. Sample bikesharing using call-a-bike having max. of 2.5km default speed: `originBike=1,0,2500,100,0,sharing,callabike`

originCar	Optional	0 or 1	1	Enables/disables using car in the beginning of a trip when searching from an address. To fine-tune the minimum and/or maximum distance to the next public transport station, provide these values separted by comma. These values are expressed in meters. Samples To enable car, minimum distance should be 2000 meters, maximum distance should be 100 kilometers set the parameter `originCar=1,0,100000`. If the default distance should be used, just put no value, e.g `1,,100000` to have car enabled, default minimum and 100 kilometers as maximum. **Other possible options** **Speed** < 100: faster = 100: normal (default) > 100: slower **Bee line calculation** 0 (default) or 1 **Vehicle mode** sharing, self (default) **Provider** enabled providers for vehicle mode, e.g. car2go, drivenow, etc. Sample Carsharing using car2go having max. of 15km default speed: `originCar=1,0,15000,100,0,sharing,car2go`

originTaxi	Optional	0 or 1	1	Enables/disables using taxi rides in the beginning of a trip when searching from an address. To fine-tune the minimum and/or maximum distance to the next public transport station, provide these values separted by comma. These values are expressed in meters. Samples To enable taxi, minimum distance should be zero meters, maximum distance should be 1000 meters set the parameter `originTaxi=1,0,1000`. If the default distance should be used, just put no value, e.g `1,,1500` to have taxi enabled, default minimum and 1500 meters as maximum. **Other possible options** **Speed** < 100: faster = 100: normal (default) > 100: slower **Bee line calculation** 0 (default) or 1

originPark	Optional	0 or 1	1	Enables/disables using Park & Ride in the beginning of a trip when searching from an address. To fine-tune the minimum and/or maximum distance to the next public transport station, provide these values separted by comma. These values are expressed in meters. Samples To enable Park & Ride, minimum distance should be zero meters, maximum distance should be 1000 meters set the parameter `originPark=1,0,1000`. If the default distance should be used, just put no value, e.g `1,,1500` to have Park & Ride enabled, default minimum and 1500 meters as maximum. **Other possible options** **Speed** < 100: faster = 100: normal (default) > 100: slower **Bee line calculation** 0 (default) or 1
originMeta	Optional	-	-	Enables using a predefined individual transport meta profile at the beginning of a trip. The profiles are defined in the HAFAS installation.

destWalk	Optional	0 or 1	1	Enables/disables using footpaths at the end of a trip when searching to an address. To fine-tune the minimum and/or maximum distance from the next public transport station, provide these values separted by comma. These values are expressed in meters. Samples To enable walk, minimum distance should be zero meters, maximum distance should be 1000 meters set the parameter `destWalk=1,0,1000`. If the default distance should be used, just put no value, e.g `1,,1500` to have walk enabled, default minimum and 1500 meters as maximum. **Other possible settings are** **Speed** < 100: faster = 100: normal (default) > 100: slower **Bee line calculation** 0 (default) or 1

destBike	Optional	0 or 1	1	Enables/disables using bike routes at the end of a trip when searching to an address. To fine-tune the minimum and/or maximum distance from the next public transport station, provide these values separted by comma. These values are expressed in meters. Samples To enable bike, minimum distance should be zero meters, maximum distance should be 1000 meters set the parameter `destBike=1,0,1000`. If the default distance should be used, just put no value, e.g `1,,1500` to have bike enabled, default minimum and 1500 meters as maximum. **Other possible settings are** **Speed** < 100: faster = 100: normal (default) > 100: slower **Bee line calculation** 0 (default) or 1 **Vehicle mode** sharing, self (default) **Provider** enabled providers for vehicle mode, e.g. callabike, etc. Sample bikesharing using call-a-bike having max. of 2.5km default speed: `destBike=1,0,2500,100,0,sharing,callabike`

destCar	Optional	0 or 1	1	Enables/disables using car routes at the end of a trip when searching to an address. To fine-tune the minimum and/or maximum distance from the next public transport station, provide these values separted by comma. These values are expressed in meters. Samples To enable car, minimum distance should be 2000 meters, maximum distance should be 100 kilometers set the parameter `destCar=1,0,100000`. If the default distance should be used, just put no value, e.g `1,,100000` to have car enabled, default minimum and 100 kilometers as maximum. **Other possible options** **Speed** < 100: faster = 100: normal (default) > 100: slower **Bee line calculation** 0 (default) or 1 **Bee line calculation** 0 (default) or 1 **Vehicle mode** sharing, self (default) **Provider** enabled providers for vehicle mode, e.g. car2go, drivenow, etc. Sample Carsharing using car2go having max. of 15km default speed: `destCar=1,0,15000,100,0,sharing,car2go`

destTaxi	Optional	0 or 1	1	Enables/disables using taxi rides at the end of a trip when searching to an address. To fine-tune the minimum and/or maximum distance from the next public transport station, provide these values separted by comma. These values are expressed in meters. Samples To enable taxi, minimum distance should be zero meters, maximum distance should be 1000 meters set the parameter `destTaxi=1,0,1000`. If the default distance should be used, just put no value, e.g `1,,1500` to have taxi enabled, default minimum and 1500 meters as maximum. **Other possible options** **Speed** < 100: faster = 100: normal (default) > 100: slower **Bee line calculation** 0 (default) or 1

destPark	Optional	0 or 1	1	Enables/disables using Park & Ride at the end of a trip when searching to an address. To fine-tune the minimum and/or maximum distance from the next public transport station, provide these values separted by comma. These values are expressed in meters. Samples To enable Park & Ride, minimum distance should be zero meters, maximum distance should be 1000 meters set the parameter `destPark=1,0,1000`. If the default distance should be used, just put no value, e.g `1,,1500` to have Park & Ride enabled, default minimum and 1500 meters as maximum. **Other possible options** **Speed** < 100: faster = 100: normal (default) > 100: slower **Bee line calculation** 0 (default) or 1
destMeta	Optional	-	-	Enables using a predefined individual transport meta profile at the end of a trip. The profiles are defined in the HAFAS installation.

totalWalk	Optional	0 or 1	1	Enables/disables using footpaths for the whole trip. To fine-tune the minimum and/or maximum distance from the next public transport station, provide these values separted by comma. These values are expressed in meters. Samples To enable walk, minimum distance should be zero meters, maximum distance should be 1000 meters set the parameter `totalWalk=1,0,1000`. If the default distance should be used, just put no value, e.g `1,,1500` to have walk enabled, default minimum and 1500 meters as maximum. **Other possible settings are** **Speed** < 100: faster = 100: normal (default) > 100: slower **Bee line calculation** 0 (default) or 1

totalBike	Optional	0 or 1	1	Enables/disables using bike routes for the whole trip. To fine-tune the minimum and/or maximum distance from the next public transport station, provide these values separted by comma. These values are expressed in meters. Samples To enable bike, minimum distance should be zero meters, maximum distance should be 1000 meters set the parameter `totalBike=1,0,1000`. If the default distance should be used, just put no value, e.g `1,,1500` to have bike enabled, default minimum and 1500 meters as maximum. **Other possible settings are** **Speed** < 100: faster = 100: normal (default) > 100: slower **Bee line calculation** 0 (default) or 1 **Vehicle mode** sharing, self (default) **Provider** enabled providers for vehicle mode, e.g. callabike, etc. Sample bikesharing using call-a-bike having max. of 2.5km default speed: `totalBike=1,0,2500,100,0,sharing,callabike`

totalCar	Optional	0 or 1	1	Enables/disables using car routes for the whole trip. To fine-tune the minimum and/or maximum distance from the next public transport station, provide these values separted by comma. These values are expressed in meters. Samples To enable car, minimum distance should be 2000 meters, maximum distance should be 100 kilometers set the parameter `totalCar=1,0,100000`. If the default distance should be used, just put no value, e.g `1,,100000` to have car enabled, default minimum and 100 kilometers as maximum. **Other possible options** **Speed** < 100: faster = 100: normal (default) > 100: slower **Bee line calculation** 0 (default) or 1 **Vehicle mode** sharing, self (default) **Provider** enabled providers for vehicle mode, e.g. car2go, drivenow, etc. Sample Carsharing using car2go having max. of 15km default speed: `totalCar=1,0,15000,100,0,sharing,car2go`

totalTaxi	Optional	0 or 1	1	Enables/disables using taxi rides for the whole trip. To fine-tune the minimum and/or maximum distance from the next public transport station, provide these values separted by comma. These values are expressed in meters. Samples To enable taxi, minimum distance should be zero meters, maximum distance should be 1000 meters set the parameter `totalTaxi=1,0,1000`. If the default distance should be used, just put no value, e.g `1,,1500` to have taxi enabled, default minimum and 1500 meters as maximum. **Other possible options** **Speed** < 100: faster = 100: normal (default) > 100: slower **Bee line calculation** 0 (default) or 1
ivOnly	Optional	0 or 1	0	Enables/disables search for individual transport routes only. Default is 0.
mobilityProfile	Optional	-	-	Use a predefined filter by its name. The filters are defined in the HAFAS installation. You are able to negate the filter by adding ! in front of the profile name. BLOCK_BACKWARDS_TRAVEL or !BLOCK_BACKWARDS_TRAVEL If there are any predefined filters available, check your delivery package documentation.

bikeCarriage	Optional	0 or 1	0	Enables/disables search for trips explicit allowing bike carriage.
sleepingCar	Optional	0 or 1	0	Enables/disables search for trips having sleeping car. Default is 0. This will only work in combination with maxChange=0 as those trips are always ment to be direct connections.
couchetteCoach	Optional	0 or 1	0	Enables/disables search for trips having couchette coach. Default is 0. This will only work in combination with maxChange=0 as those trips are always ment to be direct connections.
showPassing-Points	Optional	0 or 1	0	Enables/disables the return of stops having no alighting and no boarding in its passlist for each leg of the trip. Needs passlist enbaled.
baim	Optional	0 or 1	0	Enables BAIM search and response.
eco	Optional	0 or 1	0	Enables/disables eco value calculation.
ecoCmp	Optional	0 or 1	0	Enables/disables eco comparison.
ecoParams	Optional	-	-	Provide additional eco parameters. For exact values, check your eco documentation if any.

rtMode	Optional	-	SERV-ER_DE FAULT	Set the realtime mode to be used. Values are OFF, INFOS, FULL, REALTIME, SERVER_DEFAULT if enabled. OFF – Search on planned data, ignore real-time information completely: Connections are computed on the basis of planned data. No real-time information is shown. INFOS – Search on planned data, use real-time information for display only: Connections are computed on the basis of planned data. Delays and feasibility of the connections are integrated into the result. Note that additional trains (supplied via realtime feed) will not be part of the resulting connections. FULL – Combined search on planned and real-time data This search consists of two steps: i. Search on scheduled data ii. If the result of step (i) contains a non-feasible connection, a search on real-time data is performed. REALTIME – Search on real-time data: Connections are computed on the basis of real-time data, using planned schedule only whenever no real-time data is available. All connections computed are feasible with respect to the currently known real-time situation. Additional trains (supplied via real-time feed) will be found if these are part of a fast, comfortable, or direct connection (or economic connection, if economic search is activated). SERVER_DEFAULT – one of the above configured in the HAFAS server back end.
unsharp	Optional	0 or 1	0	Enables/disables unsharp search mode. Default is 0. For details, see 2.4.2.1.

trainFilter	Optional	-	-	Set train number with or without category to shrink the result to one. First hit will be taken.
economic	Optional	0 or 1	0	Enables/disables economic search mode. Default is 0. For details, see 2.4.2.2.
groupFilter	Optional	-	-	Use a predefined group filter to query for certain modes.
blockingList	Optional	-	-	Defines a section of a route of a journey not to be used within the trip search. Each route section is defined by a tuple of the following style: <train name>|<departure id>|<arrival id>|<departure time>|<arrival time>|<departure date>|<arrival date> A set of tuples can be separated by semicolon.
includeEarlier	Optional	0 or 1	0	Disables search optimization in relation of duration.
withICTAlterna-tives	Optional	0 or 1	0	Enables/disables the search for alternatives with individualized change times (ICT).

2.4.2 Search algorithm

The numB and numF parameters indicate the minimum number of search results returned by the service.

The HAFAS search algorithm is tuned towards finding not only the fastest connection but also convenient connections. For the given departure time, always the fastest connection is calculated. But if it turns out that the fastest connection isn't a direct connection but includes changes, also so called convenient connections are calculated. Convenient connections are connections which include a lesser number of changes than the fastest connection but don't take much longer.

When searching forward in time, HAFAS starts out searching for the fastest connection. If the fastest connection contains changes, also all convenient connections are calculated. Then the number of calculated connection is compared to the value of the numF parameter. If more connections than required are calculated, all calculated connections are returned. In the case that not enough connections are found, the start time is increased by one minute and again the fastest connection possibly along with all associated convenient connections are calculated. Then the same comparison against the minimum required number of connec-

tions in numF is performed. The last two steps are repeated until enough connections are found.

Searching backwards in time is a bit more complicated to ensure continuity with the forward connection search. The start time for the backward search is derived from the arrival time of the fastest connection of the forward search. From that time, fastest connections are calculated until the first connection is found which has a departure time earlier than the fastest connection of the forward search. Then again, the matching convenient connections are calculated. And again, the procedure is repeated until the minimum number of backward connections is exceeded.

This makes a backward search relatively costly in terms of computation time. Instead, it is recommended to shift the intended departure time backwards and make it for example 10 minutes earlier and use the first package of the fastest and matching convenient connections as the backward results.

To keep the response time of the HAFAS server at a minimum, the limitation of 6 connections combined as the maximum for connections searched backwards and forward was introduced.

2.4.2.1 Unsharp search

If the unsharp search mode is requested, the algorithm will take additional stations nearby the given start and destination station into account. These additional stops are reachable by walk. Duration and destination are not based on planning data but of GIS routers.

2.4.2.2 Economic search

Default search mode of HAFAS returns fastest and convenient trips. Using the economic search mode, more search operations with different evaluation methods are performed. This may return other trips having more or equal count of changes being faster.

To do this, HAFAS makes use of different options like considering journey attributes or exclude certain products. Also searching for outperformed direct connections is possible.

Note: The use of economic search is slower than using the normal search. Options to be used have to be configured by HAFAS.

2.4.3 Scrolling

Based on a previous result, earlier or later connections for the same trip can be easily retrieved. This way scrolling back and forth in time can be implemented. It is achieved by keeping the same request parameters as the original trip and specifiying a starting point for the scroll operation with the additional context parameter.

Each trip result contains two attributes `scrB` and `scrF` in the `TripList` element which specify starting points for scrolling back and forth. Add one of these values as the context parameter in a new trip request and the server will return earlier or later connections for the same trip.

2.4.4 Response

As a result, the service returns the calculated trips with base information for every leg of the found trips. This will include arrival and departure stop/station, arrival and departure time (incl. real-time if available).

2.4.5 Direct Train search

To get information on a train running at a specific date between two specific stations without any change, set the following parameters on a trip search:

- originId
- destinationId
- trainFilter
- date
- maxChange = 0

2.4.6 Mobility profiles

The following tables provide you a standard set of mobility profiles to be used with the parameter `mobilityProfile`. Not all of them might be available in your installationyou're your response, capacity information and barrier free information will be added as attributes at the departure and arrival stops. See chapter 1.2.11 and 1.2.12.

2.4.6.1 Capacity utilization filtes

Code	Description	Possible results
ctg_1st_0	First class capactity: Do not filter	cap1st_11, cap1st_12, cap1st_13
ctg_1st_1	First class capactity: Filter for low to medium capacity utilization.	cap1st_11
ctg_1st_2	First class capactity: Filter for high capacity utilization.	cap1st_11, cap1st_12
ctg_1st_3	First class capactity: Filter for very high capacity utilization.	cap1st_11, cap1st_12, cap1st_13
ctg_2nd_0	Second class capactity: Do not filter	cap2nd_11, cap2nd_11, cap2nd_13
ctg_2nd _1	Second class capactity: Filter for low to medium capacity utilization.	cap2nd_11

ctg_2nd _2	Second class capactity: Filter for high capacity utilization.	cap2nd_11, cap2nd_11
ctg_2nd _3	Second class capactity: Filter for very high capacity utilization.	cap2nd_11, cap2nd_11, cap2nd_13

2.4.6.2 Barrier free filters (baim)

Code	Description	Possible results
hcp_0	Do not filter.	baim_1, baim_2, baim_3, baim_4
hcp_1	Boarding and alighting possible.	baim_1
hcp_2	Boarding and alighting possible with the help of the crew.	baim_1, baim_2
hcp_3	Boarding and alighting requires advance notification.	baim_1, baim_2, baim_3

Result values baim_info and baim_rsv are not filterable and will be returned if present.

2.5 Interval trip search service

The `interval trip search` service calculates trips from a specified origin to a specified destination in a time interval starting at a given date and time.

2.5.1 Request Parameters

Request parameters are very much the same as used in the Trip search service with two additions:

Name	Use	Range	Default	Description
date	Mandatory	See 1.2.2	Current server date	Sets the departure date for the search.
time	Mandatory	See 1.2.2	Current server time	Sets the departure time for the search.

Name	Use	Range	Default	Description
duration	Mandatory	1 to 1439	-	Time interval to search for trips in minutes.
max	Optional	-	-	Maximum number of trips returned.

Following parameters have no effect:

Name	Use	Range	Default	Description
numF	Not used			
numB	Not used			

2.5.2 Response

As a result, the service returns the calculated trips with base information for every leg of the found trips. This will include arrival and departure stop/station, arrival and departure time (incl. real-time if available).

2.6 Reconstruction service

Reconstructing a trip can be achieved using the resconstruction context provided by any trip result in the `ctxRecon` attribute of `Trip` element. The result will be a true copy of the original trip search result given that the underlying data did not change.

2.6.1 Request Parameters

Name	Use	Range	Default	Description
accessId	Mandatory	-	-	Access ID for identifying the requesting client.
ctx	Mandatory	-	-	Specifies the reconstruction context.
poly	Optional	0 or 1	0	Enables/disables the calculation of the polyline for each leg of the trip.

date	Optional			This parameter will force the service to reconstruct the trip on that specific date. If the trip is not available on that date, because it does not operate, the error code SVC_NO_RESULT will be returned.
passlist	Optional	0 or 1	0	Enables/disables the return of the passlist for each leg of the trip.
showPassingPoints	Optional	0 or 1	0	Enables/disables the return of stops having no alighting and no boarding in its passlist for each leg of the trip. Needs passlist parameter set to 1.
eco	Optional	0 or 1	0	Enables/disables eco value calculation.
ecoCmp	Optional	0 or 1	0	Enables/disables eco comparison.
ecoParams	Optional	-	-	Provide additional eco parameters. For exact values, check your eco documentation if any.

2.6.2 Example

Request:

Reconstruct the trip from Varhaug, Stasjonsvegen 29 to Holmestrand, Langgaten 2 on 18th September 2014 at 14:43

```
<baseurl>/ recon?ctx=G@F$A=2@O=Varhaug, Stasjonsvegen
29@X=5646115@Y=58618325@u=36@a=128@$A=1@O=Varhaug@L=7602220@a=128@$2
01409181430$201409181443$$$T$A=1@O=Varhaug@L=7602220@a=128@$A=1@O=Eg
ersund@L=7602212@a=128@$201409181443$201409181510$
3040$$T$A=1@O=Egersund@L=7602212@a=128@$A=1@O=Drammen@L=7601421@a=12
8@$201409181525$201409182152$
728$$T$A=1@O=Drammen@L=7601421@a=128@$A=1@O=Holmestrand@L=7601505@a=
128@$201409182215$201409182238$
R10$$G@F$A=1@O=Holmestrand@L=7601505@a=128@$A=2@O=Holmestrand, Lang-
gaten
2@X=10312173@Y=59492256@u=60@a=128@$201409182238$201409182244$$
```

Response will follow the structure of trip service but containing one trip only if any.

2.7 Search on trip

The search on trip service performs a trip search starting at a location of a journey. This journey is identified by its name first and last stop information. The next routable location is then identified by the date and time given.

2.7.1 Request Parameters

Name	Use	Range	Default	Description
accessId	Mandatory	-	-	Access ID for identifying the requesting client.
match	Mandatory	-	-	Matching criteria like train name, number or both. We recommend both always. Sample: - ICE 827 - RE 48 - S1
firstStopId	Mandatory	See 2.3.2 or 2.3.3	-	Specifies the first station/stop ID (location reconstruction context) of the journey this search is based on. Such ID can be retrieved from the location.name or location.nearbystops services.
firstStopDate	Mandatory	See 1.2.2	Current server date	Departure date for the first stop of the journey this search is based on.
firstStopTime	Mandatory	See 1.2.2	Current server time	Departure time for the first stop of the journey this search is based on.
lastStopId	Mandatory	See 2.3.2 or 2.3.3	-	Specifies the last station/stop ID (location reconstruction context) of the journey this search is based on. Such ID can be retrieved from the location.name or location.nearbystops services.

lastStopDate	Mandatory	See 1.2.2	Current server date	Arrival date for the last stop of the journey this search is based on.
lastStopTime	Mandatory	See 1.2.2	Current server time	Arrival time for the last stop of the journey this search is based on.
currentDate	Mandatory	See 1.2.2	Current server date	Sets the departure date for the search.
currentTime	Mandatory	See 1.2.2	Current server time	Sets the departure time for the search.
destId	Mandatory	See 2.3.2 or 2.3.3	-	Specifies the station/stop ID (location reconstruction context) of the destination for the trip. Such ID can be retrieved from the location.name or location.nearbystops services.
viaId	Optional	See 2.3.2 or 2.3.3		ID of a station/stop used as a via for the trip. Specifying a via station forces the trip search to look for trips which must pass through this station. Such IDs can be retrieved from the location.name or location.nearbystops services. If via is used, viaId and viaWaitTime are having no effect.
operators	Optional	All operator codes or names from HAFAS raw data file *betrieb*.	-	Only trips provided by the given operators are part of the result. To filter multiple operators, separate the codes by comma. If the operator should not be part of the trip, negate it by putting ! in front of it. E.g. filter for A and B operator: operators=A,B.

products	Optional	-	-	Decimal value defining the product classes to be included in the search. It represents a bitmask combining bit number of a product as defined in the HAFAS raw data file zugart. If, for example, you would like to search for local and regional trains only, and your HAFAS raw data file *zugart* states regional trains are product class 2 and local trains are class 3, you need a bitmask where bits 2 and 3 are set. Calculation is then: 2^2 + 2^3 = 12 which would be the parameter value for "products".
blockingList	Optional	-	-	Defines a section of a route of a journey not to be used within the search on trip search. Each route section is defined by a tuple of the following style: <train name>|<departure id>|<arrival id>|<departure time>|<arrival time>|<departure date>|<arrival date> A set of tuples can be separated by semicolon.
poly	Optional	0 or 1	0	Enables/disables the calculation of the polyline for each leg of the trip.

2.8 GIS route service

The `gisroute` service takes a GIS reference as provided by a Trip result like this and delivers a routing graph, routing instructions as well as distance information.

2.8.1 Request Parameters

Name	Use	Range	Default	Description
accessId	Mandatory	-	-	Access ID for identifying the requesting client.

ctx	Mandatory	-	-	Specifies the GIS route context.
poly	Optional	0 or 1	0	Enables/disables the calculation of the polyline for each leg of the trip.
eco	Optional	0 or 1	0	Enables/disables eco value calculation.

2.8.2 Example

Request: Indoor routing between two tracks

```
<baseurl>/gisroute?accessId=nsr&ctx=G|1|G@F|A=2@O=1062HD Amsterdam, Cornelis Lelylaan
35@l=@X=4834021@Y=52357940@u=9@|A=1@O=Amsterdam Le-
lylaan@X=4833913@Y=52357887@U=684@L=8400079@|25082016|123500|123600|bf|ft@0@2000@120@-
1@100@1@1000@0@@@@@false@0@$f@$f@$f@$f@$f@$$bt@0@2000@120@-
1@100@1@1000@0@@@@@false@0@$f@$f@$f@$f@$f@$$tt@0@5000@120@-
1@100@1@2500@0@@@@@false@0@$t@0@25000@120@-1@100@1@3000@0@@@@@false@0@$f@$f@$f@$f@$$|
```

Response will follow the structure of trip service but containing one trip only if any.

```
<?xml version="1.0" encoding="UTF-8"?>
<TripList serverVersion="1.5-SNAPSHOT" dialectVersion="1.23"
  requestId="1483016905755" xmlns="hafas_rest">
  <Trip idx="0"
    ctxRecon="G@F$A=2@O=1062HD Amsterdam, Cornelis Lelylaan
35@X=4834021@Y=52357940@u=9@a=128@$A=1@O=Amsterdam Le-
lylaan@L=8400079@a=0@$201608251236$201608251236$$"
    checksum="523B5244_4" tripId="C-0" duration="PT0S"

ecoUrl="demo.hafas.de/bin/pub/bene/eco/query.exe/dn?&amp;L=ns_hispeed_eco&amp;application=ECOL
OGYIN-
FO&amp;requestConnection=1&amp;conReconstruction=1&amp;newEcologyValues=1&amp;ecoFrom=1062HD
Amsterdam, Cornelis Lelylaan
35&amp;ecoFromId=205996696&amp;ecoFromX=4834021&amp;ecoFromY=52357940&amp;ecoTo=Amsterdam Le-
lylaan&amp;ecoToId=008400079&amp;ecoToX=4833913&amp;ecoToY=52357887&amp;VH=G@F$A=2@O=1062HD
Amsterdam, Cornelis Lelylaan 35@X=4834021@Y=52357940@u=9@a=128@$A=1@O=Amsterdam Le-
lylaan@L=8400079@a=0@$201608251236$201608251236$$&amp;">
    <LegList>
      <Leg type="WALK" idx="0" dist="8" duration="PT0S" name="">
        <Origin
          id="A=2@O=1062HD Amsterdam, Cornelis Lelylaan 35@X=4834021@Y=52357940@u=9@"
          name="1062HD Amsterdam, Cornelis Lelylaan 35" type="ADR" lon="4.834021"
          lat="52.35794" date="2016-08-25" time="12:36:00" />
        <Destination
          id="A=1@O=Amsterdam Lelylaan@X=4833913@Y=52357887@U=684@L=8400079@"
          extId="8400079" name="Amsterdam Lelylaan" type="ST" lon="4.833913"
          lat="52.357887" date="2016-08-25" time="12:36:00" />
        <GisRef
          ref="G|1|G@F|A=2@O=1062HD Amsterdam, Cornelis Lelylaan
35@X=4834021@Y=52357940@u=9@|A=1@O=Amsterdam Le-
lylaan@X=4833913@Y=52357887@U=684@L=8400079@|25082016|123600|123600|ft|ft@0@2000@120@-
1@100@1@1000@0@@@@@false@0@$f@$f@$f@$f@$f@$$bt@0@2000@120@-
1@100@1@1000@0@@@@@false@0@$f@$f@$f@$f@$f@$$tt@0@5000@120@-
1@100@1@2500@0@@@@@false@0@$t@0@25000@120@-1@100@1@3000@0@@@@@false@0@$f@$f@$f@$f@$$|" />
        <GisRoute dist="8" durS="PT0S">
          <seg dist="8" manTx="Nehmen Sie die Treppe" name="Treppe" ori="E"
            rType="CT" />
          <seg man="TO" manTx="Nehmen Sie die Treppe" />
        </GisRoute>
      </Leg>
    </LegList>
```

```
  </Trip>
</TripList>
```

2.9 Stationboard services

The station board can be retrieved by a call to the `departureBoard` or `arrivalBoard` services. This method will return the next departures (or less if not existing) or arrivals from a given point in time within a duration covered time span. The default duration size is 60 minutes.

Note: The result list always contains all departures/arrivals running the the last minute found even if the requested maximum was overrun.

2.9.1 Request Parameters

Name	Use	Range	Default	Description
accessId	Mandatory	-	-	Access ID for identifying the requesting client.
id	Mandatory if extId is not specified	See 2.3.2 or 2.3.3	-	Specifies the station/stop ID for which the departures or arrivals shall be retrieved. Such ID can be retrieved from the location.name or location.nearbystops services.
extId	Mandatory if id is not specified			Specifies the external station/stop ID. Such ID can be retrieved from the location.name or location.nearbystops services
direction	Optional	See 2.3.2 or 2.3.3		If only vehicles departing or arriving from a certain direction shall be returned, specify the direction by giving the station/stop ID of the last stop on the journey.
date	Optional	See 1.2.2	Current server date	Sets the start date for which the departures or arrivals shall be retrieved.
time	Optional	See 1.2.2	Current server time	Sets the start time for which the departures or arrivals shall be retrieved.

duration	Optional	0 - 1439	60	Set the interval size in minutes.
dur	Optional	0 - 1439	60	Deprecated. Please use **duration**.
products	Optional	-	-	Decimal value defining the product classes to be included in the search. It represents a bitmask combining bit number of a product as defined in the HAFAS raw data file *zugart*. For example, regional trains are product class 2 and local trains are class 3, while busses are 4. If you would like to search for local and regional trains only, you would need a bitmask where bits 2 and 3 are set. Calculation is 2^2 + 2^3 = 12 which would be the parameter value for "products". When searching for busses only, "products" need to be set to 16 = 2^4.
operators	Optional	All operator codes or names from HAFAS raw data file *betrieb*.	-	Only journeys provided by the given operators are part of the result. To filter multiple operators, separate the codes by comma. If the operator should not be part of the result, negated it by putting ! in front of it. E.g. filter for A and B operator: operators=A,B.
lines	Optional	-	-	Only journeys running the given line are part of the result. To filter multiple lines, separate the codes by comma. If the line should not be included, negated it by putting ! in front of it. E.g. filter for lines 120 and 140: lines=120,140

maxJourneys	Optional		-	Maximum number of journeys to be returned. If no value is defined, all departing/arriving services within the duration sized period are returned. Please note: maxJourneys is not a hard limit. If the limit of maxJourneys is reached and there are additional journeys that have the same departure/arrival time as the last journey within the limit (e.g. 14:57), those additional journeys are also returned. This ensures that scrolling forward works by executing another departure/arrival board request where the time is equal to the departure/arrival time of the last journey increased by one minute (14:58 in our example).
filterEquiv	Optional	0 or 1	1	Enables/disables the filtering of equivalent marked stops.
attributes	Optional	All attribute codes from HAFAS raw data.	-	Filter arriving or departing journeys by one or more attribute codes. Multiple attribute codes are separated by comma. If the attribute should not be part of the journey, negate it by putting ! in front of it.
rtMode	Optional	-	SERVER_DEFAULT	Set the realtime mode to be used. Values are OFF, INFOS, FULL, REALTIME, SERVER_DEFAULT if enabled.

A response will return `DepartureBoard` or `ArrivalBoard` as defined in the HAFAS Proxy XSD (see chapter 3.3). This will contain a list of departures/arrivals with train/line number, type of transport, departure or arrival times (incl. real-time), departure or arrival stop/stations (might be different from requested stop), direction text and a track information if available. Every departure or arrival will also contain a reference to the Journey Detail Service.

2.9.2 Example

Request: Departure board for Oslo S on 1st June 2014 at 18:00

```
<baseurl>/departureBoard?id=A=1@L=007600100&date=2014-06-01
&time=18:00
```

Result: (abbreviated)

```
<DepartureBoard xmlns="hafas_rest_v1">
    <Departure direction="Gardermoen" name="F2" trainNumber="3781"
        trainCategory="5" stopid="A=1@O=Oslo S@X=10755332@Y=59910200
        @U=70@L=7600100@" stop="Oslo S" date="2014-06-01"
        time="18:00:00" track="13">
    <JourneyDetailRef ref="1|25|0|70|1062014"/>
    </Departure>
    <Departure direction="Göteborg C" name="R20" trainNumber="127"
        trainCategory="2" stopid="A=1@O=Oslo S@X=10755332@Y=59910200
        @U=70@L=7600100@" stop="Oslo S" date="2014-06-01"
        time="18:02:00" track="18">
        <JourneyDetailRef ref="1|1977|0|70|1062014"/>
    </Departure>
    <Departure direction="Skøyen" name="L22" trainNumber="1928"
        trainCategory="5" stopid="A=1@O=Oslo S@X=10755332@Y=59910200
        @U=70@L=7600100@" stop="Oslo S" date="2014-06-01"
        time="18:03:00" track="7">
        <JourneyDetailRef ref="1|329|0|70|1062014"/>
    </Departure>
   …
<DepartureBoard>
```

2.9.3 Scrolling

To scroll station boards, following action is to be performed: Take the departure/arrival time of the last departure/arrival of your result. Add one minute and do the same request with the new time value again. If the response passes midnight, the date has to be incremented too.

Note: This is possible, because the result list always completes the departures of the last minute found even if a maxJourneys value needs to be overrun.

2.10 Journey Detail Service

The `journeyDetail` service will deliver information about the complete route of a vehicle. The journey identifier is part of a `trip` or `departureBoard` response. It contains a list of all stops/stations of this journey including all departure and arrival times (with real-time data if available) and additional information like specific attributes about facilities and other texts.

2.10.1 Request Parameters

Name	Use	Range	Default	Description
accessId	Mandatory	-	-	Access ID for identifying the requesting client.

id	Mandatory	See 2.3.2 or 2.3.3	-	Specifies the internal journey id of the journey that shall be retrieved. It may be necessary to escape the | character by its URL encoding %7C.
date	Optional	See 1.2.2	Current server date	Day of operation
poly	Optional	0 or 1	0	Enables/disables the calculation of the polyline for this journey.
showPassing-Points	Optional	0 or 1	0	Enables/disables the return of stops having no alighting and no boarding in its passlist for each leg of the trip.
rtMode	Optional	-	SERV-ER_DE FAULT	Set the realtime mode to be used. Values are OFF, INFOS, FULL, REALTIME, SERVER_DEFAULT if enabled.

2.10.2 Example

Request: Get the journey details of the first journey returned by the example for the DepartureBoard service

```
<baseurl>/journeyDetail?id=1|25|0|70|1062014
```

Result: (abbreviated)

```
<JourneyDetail xmlns="hafas_rest_v1">
    <Stops>
        <Stop id="A=1@O=Drammen@X=10204842@Y=59740160@U=70
            @L=7601421@" name="Drammen" routeIdx="0" extId="7601421"
            lon="10.204842" lat="59.74016" depDate="2014-06-01"
            depTime="17:22:00" track="3"/>
        <Stop id="A=1@O=Asker@X=10434552@Y=59833747@U=70@L=7601413@"
            name="Asker" routeIdx="1" extId="7601413"
            lon="10.434552" lat="59.833747" depDate="2014-06-01"
            depTime="17:35:00" track="3"/>
        <Stop id="A=1@O=Sandvika@X=10526017@Y=59893022@U=70
            @L=7601408@" name="Sandvika" routeIdx="2"
            extId="7601408" lon="10.526017" lat="59.893022"
            depDate="2014-06-01" depTime="17:41:00" track="4"/>    …
    </Stops>
    <Names>
        <Name name="F2" routeIdxFrom="0" routeIdxTo="8"
            number="3781" category="5"/>
    </Names>
```

```
    <Directions>
        <Direction routeIdxFrom="0" routeIdxTo="8">
            Gardermoen
        </Direction>
    </Directions>
</JourneyDetail>
```

2.11 Journey match

The `journeyMatch` service will deliver information about the complete route of a journey if it is matched by any of the given criteria. It only returns details about the first matched journey. If you need all matching journeys with fewer details, please refer to the Train Search service. The `journeyMatch` service returns the same structure as Journey Detail Service.

2.11.1 Request parameters

Name	Use	Range	Default	Description
accessId	Mandatory	-	-	Access ID for identifying the requesting client.
match	Optional	-	-	Matching criteria like train name, number or both. We recommend both always. Sample: - ICE 827 - RE 48 - S1 **Even if optional, this field should be filled.**
filters	Optional	-	-	Customer specific filters. Read additional document if available for your installation.
stations	Optional	-	-	Filter for stations. Matches if the given value is prefix of any station in the itinerary. Multiple values are separated by comma.

operators	Optional	All operator codes or names from HAFAS raw data file *betrieb*.	-	Filter for operators. To filter multiple operators, separate the codes by comma. E.g. filter for A and B operator: operators=A,B.
date	Optional	See 1.2.2	-	Day of operation. **If not provided, all matching trains of the current timetable are taken into account. This will slow the operation at all.**
time	Optional	See 1.2.2	-	Time the service operates according to scheduled data. If not provided, the whole day is taken into account.
showPassingPoints	Optional	0 or 1	0	Enables/disables the return of stops having no alighting and no boarding in its passlist for each leg of the trip. Needs passlist enbaled.

2.11.2 Example

Request: Get the journey details of the first matched journey returned

```
<baseurl>/journeyMatch?accessId=abc&match=IR2169&date=2016-11-14
```

Result: (abbreviated)

```
<?xml version="1.0" encoding="UTF-8"?>
<JourneyDetail serverVersion="1.1"
  dialectVersion="1.23" ref="1|26883|4|85|14112016" xmlns="hafas_rest">
  <Stops>
    <Stop id="A=1@O=Bern@X=7439122@Y=46948825@U=85@L=8507000@" name="Bern"
      routeIdx="0" extId="8507000" depPrognosisType="PROGNOSED" lon="7.439122"
      lat="46.948825" depDate="2016-11-14" depTime="10:34:00" depTrack="9">
      <Notes>
        <Note key="RA" priority="999" type="A">RT_BHF</Note>
      </Notes>
    </Stop>
    <Stop id="A=1@O=Olten@X=7907685@Y=47351929@U=85@L=8500218@" name="Olten"
      routeIdx="2" extId="8500218" lon="7.907685" lat="47.351929" arrDate="2016-11-14"
      arrTime="11:00:00" depDate="2016-11-14" depTime="11:02:00" depTrack="4">
      <Notes>
        <Note key="RA" priority="999" type="A">RT_BHF</Note>
      </Notes>
    </Stop>
    <Stop id="A=1@O=Aarau@X=8051251@Y=47391355@U=85@L=8502113@" name="Aarau"
      routeIdx="3" extId="8502113" lon="8.051251" lat="47.391355" arrDate="2016-11-14"
```

```
      arrTime="11:13:00" depDate="2016-11-14" depTime="11:15:00" depTrack="2">
      <Notes>
        <Note key="RA" priority="999" type="A">RT_BHF</Note>
      </Notes>
    </Stop>
    <Stop id="A=1@O=Brugg AG@X=8208823@Y=47480852@U=85@L=8500309@"
      name="Brugg AG" routeIdx="4" extId="8500309" lon="8.208823" lat="47.480852"
      arrDate="2016-11-14" arrTime="11:28:00" depDate="2016-11-14" depTime="11:30:00"
      depTrack="3">
      <Notes>
        <Note key="RA" priority="999" type="A">RT_BHF</Note>
      </Notes>
    </Stop>
    <Stop id="A=1@O=Baden@X=8307696@Y=47476420@U=85@L=8503504@" name="Baden"
      routeIdx="5" extId="8503504" lon="8.307696" lat="47.47642" arrDate="2016-11-14"
      arrTime="11:36:00" depDate="2016-11-14" depTime="11:38:00" depTrack="1">
      <Notes>
        <Note key="RA" priority="999" type="A">RT_BHF</Note>
      </Notes>
    </Stop>
    <Stop id="A=1@O=Zürich HB@X=8540193@Y=47378177@U=85@L=8503000@"
      name="Zürich HB" routeIdx="6" extId="8503000" arrPrognosisType="PROGNOSED"
      lon="8.540193" lat="47.378177" arrDate="2016-11-14" arrTime="11:54:00"
      arrTrack="18">
      <Notes>
        <Note key="RA" priority="999" type="A">RT_BHF</Note>
      </Notes>
    </Stop>
  </Stops>
  <Names>
    <Name routeIdxFrom="0" routeIdxTo="6" name="IR 2169 " number="2169"
      category="IR">
      <Product catCode="2" catIn="IR" catOut="IR" catOutL="InterRegio"
        catOutS="IR" line="" name="IR 2169 " num="2169"
        operator="Schweizerische Bundesbahnen SBB" operatorCode="SBB" admin="000011" />
    </Name>
  </Names>
  <Directions>
    <Direction routeIdxFrom="0" routeIdxTo="6">Zürich HB</Direction>
  </Directions>
  <JourneyStatus>P</JourneyStatus>
  <ServiceDays sDaysR="nicht täglich" sDaysI="14. Nov bis 10. Dez 2016 täglich"

sDaysB="FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFE7FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF0000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000" />
  <ServiceDays sDaysR="nicht täglich" sDaysI="14. Nov bis 10. Dez 2016 täglich"

sDaysB="FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF0000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000" />
  <lastPassRouteIdx>0</lastPassRouteIdx>
  <lastPassStopRef>0</lastPassStopRef>

</JourneyDetail>
```


2.12 Journey Pos Service

The journeyPos service delivers real time position information for given journeys inside a map region. The region is required and is defined via a bounding box. Results can be filtered by operators, products and lines as well as further criteria.

2.12.1 Request parameters

Name	Use	Range	Default	Description
accessId	Mandatory	-	-	Access ID for identifying the requesting client.
llLat	Mandatory	See 1.2.1	-	Latitude of the lower left corner of the bounding box.
llLon	Mandatory	See 1.2.1	-	Longitude of the lower left corner of the bounding box.
urLat	Mandatory	See 1.2.1	-	Latitude of the upper right corner of the bounding box.
urLon	Mandatory	See 1.2.1	-	Longitude of the upper right corner of the bounding box.
operators	Optional	All operator codes or names from HAFAS raw data file *betrieb*.	-	Filter for operators. To filter multiple operators, separate the codes by comma. E.g. filter for A and B operator: operators=A,B.
products	Optional	-	-	Decimal value defining the product classes to be included in the search. It represents a bitmask combining bit number of a product as defined in the HAFAS raw data file *zugart*.
attributes	Optional	-	-	Filter for attributes. Inklusive list of comma separated attributes.
lines	Optional	-	-	Filter for lines. Inklusive list of comma separated lines
jid	Optional	-	-	Filter for journey ids. To include multiple journey ids, separate by comma
infotexts	Optional	-	-	Filter by custom infotexts. Multiple filters separated by comma.

maxJny	Optional	-	1000	Maximum number of journies in response.
date	Optional	See 1.2.2	-	Day of operation. **If not provided, all matching trains of the current timetable are taken into account. This will slow down the operation.**
time	Optional	See 1.2.2	-	Time the service operates according to scheduled data. If not provided, the whole day is taken into account.

2.12.2 Example

Get details to all journeys inside a bounding box.

```
<baseseurl>/journeyPos?accessId=a&
llLon=1.500&llLat=48.345&urLon=3.301&urLat=49.408&infotexts=CT|TGV
```

Result: (abbreviated)

```
<?xml version="1.0" encoding="UTF-8"?>
<JourneyList serverVersion="1.14-SNAPSHOT"
        dialectVersion="1.23" requestId="1528465447790" xmlns="hafas_rest">
        <Journey name="" trainNumber="" trainCategory="" lon="2.57078"
                lat="49.005701">
                <JourneyDetailRef ref="1|217176|0|87|8062018" />
                <Product catCode="1" lineId="C" name="OUIGO 7839" />
                <Notes>
                        <Note routeIdxFrom="0" routeIdxTo="6" key="ID">OUIGO 7839</Note>
                        <Note routeIdxFrom="0" routeIdxTo="6" key="rt">7839</Note>
                </Notes>
        </Journey>
        <Journey name="" trainNumber="" trainCategory="" lon="2.344737"
                lat="48.938291">
                <JourneyDetailRef ref="1|40217|0|87|8062018" />
                <Product catCode="1" lineId="C" name="Eurostar 9024" />
                <Notes>
                        <Note routeIdxFrom="0" routeIdxTo="2" key="ID">Eurostar 9024</Note>
                        <Note routeIdxFrom="0" routeIdxTo="2" key="rt">9024</Note>
                </Notes>
        </Journey>
        <Journey name="" trainNumber="" trainCategory="" lon="2.87466"
                lat="48.470142">
                <JourneyDetailRef ref="1|31533|0|87|8062018" />
                <Product catCode="1" lineId="C" name="TGV 5324" />
                <Notes>
                        <Note routeIdxFrom="0" routeIdxTo="7" key="ID">TGV 5324</Note>
                        <Note routeIdxFrom="0" routeIdxTo="7" key="rt">5324</Note>
                </Notes>
        </Journey>
</JourneyList>
```

2.13 Journey Validation Service

The `journeyValidation` service allows testing if position information is available for provided train numbers. This is done by filtering via infotext. It returns a list (`JourneyValidation`). Only queries using the RT infotexts filter are allowed, see 2.13.2 for a valid example.

2.13.1 Request parameters

Name	Use	Range	Default	Description
accessId	Mandatory	-	-	Access ID for identifying the requesting client.
infotexts	Mandatory	-	-	Filter by custom infotexts. Multiple filters separated by comma.
llLat	Optional	See 1.2.1	-	Latitude of the lower left corner of the bounding box.
llLon	Optional	See 1.2.1	-	Longitude of the lower left corner of the bounding box.
urLat	Optional	See 1.2.1	-	Latitude of the upper right corner of the bounding box.
urLon	Optional	See 1.2.1	-	Longitude of the upper right corner of the bounding box.
date	Optional	See 1.2.2	-	Day of operation.
time	Optional	See 1.2.2	-	Time the service operates according to scheduled data.

2.13.2 Example

Get details to all journeys inside a bounding box.

```
<baseurl>/journeyValidation?accessId=a&
infotexts=RT|862412,RT|884416
```

Result:

```
<?xml version="1.0" encoding="UTF-8"?>
<JourneyValidation serverVersion="1.14-SNAPSHOT"
        dialectVersion="1.23" requestId="1528797948147">
        <item name="862412" value="false" />
        <item name="884416" value="true" />
</JourneyValidation>
```

2.14 Train search

The `trainSearch` service will deliver information about the route of a journey if it is matched by any of the given criteria. It returns a list (`JourneyDetailList`) of matched journeys with as much details as possible. The stop list will only contain first and last stop. If you need more details or the complete stop list, take the `JourneyDetail@ref` and put it to Journey Detail Service.

This service tries to find all matching trains by the different match criterias. Even if **match** and **date** are optional parameters, they should always be filled. Otherwise, the whole planning period in behind will be searched which will slow the service as well and will deliver more results than useful.

To not break the system, Train search has a configurable server side result limit which defaults to 25000. If this limit is reached, results are not weighted and sorted. But if there is such a high amount of results, the query needs to be refined.

2.14.1 Request parameters

Name	Use	Range	Default	Description
accessId	Mandatory	-	-	Access ID for identifying the requesting client.
match	Optional	-	-	Matching criteria like train name, number or both. We recommend both always. Sample: - ICE 827 - RE 48 - S1 **Even if optional, this field should be filled always.**
filters	Optional	-	-	Customer specific filters. Read additional document if available for your installation.
stations	Optional	-	-	Filter for stations. Matches if the given value is prefix of any station in the itinerary. Multiple values are separated by comma.

operators	Optional	All operator codes or names from HAFAS raw data file *betrieb*.	-	Filter for operators. To filter multiple operators, separate the codes by comma. E.g. filter for A and B operator: operators=A,B.
date	Optional	See 1.2.2	-	Day of operation. **If not provided, all matching trains of the current timetable are presented. This will slow the operation at all.**
time	Optional	See 1.2.2	-	Time the service operates according to scheduled data. If not provided, the whole day is taken into account.

2.14.2 Example

Request: Get the journey details of the first matched journey returned

```
<baseurl>/trainSearch?accessId=abc&match=S4&date=2016-11-14
```

Result: (abbreviated)

```
<?xml version="1.0" encoding="UTF-8"?>
<JourneyDetailList serverVersion="1.1"
 dialectVersion="1.23" xmlns="hafas_rest">
 <JourneyDetail ref="1|287783|0|85|14112016">
  <Stops>
   <Stop id="A=1@O=Milano Centrale@X=9204711@Y=45486388@U=85@L=8301700@"
    name="Milano Centrale" routeIdx="0" extId="8301700" lon="45.486388"
    lat="9.204711" depDate="2016-11-14" depTime="08:58:00" />
   <Stop id="A=1@O=Bellinzona@X=9029512@Y=46195439@U=85@L=8505213@"
    name="Bellinzona" routeIdx="0" extId="8505213" lon="9.029512" lat="46.195439"
    arrDate="2016-11-14" arrTime="09:56:00">
    <Notes>
     <Note key="RA" priority="999" type="A">RT_BHF</Note>
    </Notes>
   </Stop>
  </Stops>
  <Names>
   <Name routeIdxFrom="0" routeIdxTo="0" name="" number=""
    category="">
    <Product catCode="" catIn="" catOut="" catOutL="" catOutS=""
     line="" lineId="5_000011_10" name="" num="" operator="SBB"
     operatorCode="SBB" />
   </Name>
  </Names>
  <JourneyStatus>P</JourneyStatus>
  <ServiceDays
   sDaysR="13. Dez 2015 bis 10. Dez 2016 täglich; nicht 26. Jun bis 28. Aug 2016, 29., 30. Okt 2016" />
  <lastPassRouteIdx>0</lastPassRouteIdx>
  <lastPassStopRef>1</lastPassStopRef>
```

```
  </JourneyDetail>
  <JourneyDetail ref="1|287789|0|85|14112016">
    <Stops>
      <Stop id="A=1@O=Milano Centrale@X=9204711@Y=45486388@U=85@L=8301700@"
        name="Milano Centrale" routeIdx="0" extId="8301700" lon="45.486388"
        lat="9.204711" depDate="2016-11-14" depTime="13:58:00" />
      <Stop id="A=1@O=Bellinzona@X=9029512@Y=46195439@U=85@L=8505213@"
        name="Bellinzona" routeIdx="0" extId="8505213" lon="9.029512" lat="46.195439"
        arrDate="2016-11-14" arrTime="14:56:00">
        <Notes>
          <Note key="RA" priority="999" type="A">RT_BHF</Note>
        </Notes>
      </Stop>
    </Stops>
    <Names>
      <Name routeIdxFrom="0" routeIdxTo="0" name="" number=""
        category="">
        <Product catCode="" catIn="" catOut="" catOutL="" catOutS=""
          line="" lineId="5_000011_10" name="" num="" operator="SBB"
          operatorCode="SBB" />
      </Name>
    </Names>
    <JourneyStatus>P</JourneyStatus>
    <ServiceDays
      sDaysR="13. Dez 2015 bis 10. Dez 2016 täglich; nicht 26. Jun bis 28. Aug 2016, 29., 30. Okt 2016" />
  </JourneyDetail>
...
</JourneyDetailList>
```

2.15 HIM search

The `himSearch` service will deliver a list of HIM messages if matched by the given criteria as well as affected products if any.

2.15.1 Request parameters

Name	Use	Range	Default	Description
accessId	Mandatory	-	-	Access ID for identifying the requesting client.
dateB	Optional	See 1.2.2	-	Sets the event period start date.
dateE	Optional	See 1.2.2	-	Sets the event period end date.
timeB	Optional	See 1.2.2	-	Sets the event period start time.
timeE	Optional	See 1.2.2	-	Sets the event period end time.
himIds	Optional	-	-	List of HIM message IDs seperated by comma.

operators	Optional	-	-	List of operators seperated by comma.
categories	Optional	-	-	List of train categories seperated by comma. Value depends on your HAFAS server data.
channels	Optional	-	-	List of channels seperated by comma.
companies	Optional	-	-	List of companies seperated by comma.
metas	Optional	-	-	List of predefined filters seperated by comma.
himcategory	Optional	-	-	Filter by HIM category, e.g. Works and/or Disturbance. Value depends on your HAFAS server data.
poly	Optional	0 or 1	0	Enables/disables returning of geo information for affected edges and regions if available and enabled in the backend.
searchmode	Optional	-	-	HIM search mode. Possible options: NOMATCH iterate over all HIM messages available. MATCH iterate over all trips to find HIM messages. TFMATCH uses filters defined at **metas** parameter.
minprio	Optional	-	-	Filter for HIM messages having at least this priority.
maxprio	Optional	-	-	Filter for HIM messages having this priority as maximum.

2.15.2 Example

Request: Get the HIM messages for

```
<baseurl>/ himsearch?accessId=123&dateB=2015-06-01&dateE=2016-07-29
```

Result: (abbreviated)

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<HimMessages serverVersion="1.1" dialectVersion="1.0"
  xmlns="hafas_rest">
  <Message id="3419" act="true" head="Bauarbeiten."
    text="Der Zug fällt zwischen Dessau Hbf und Bitterfeld aus. Ein Ersatzverkehr von Dessau
Hbf nach Bitterfeld ist eingerichtet. Bitte überprüfen Sie Ihre Verbindung noch einmal kurz
vor der Reise."
    priority="5" category="3" products="65535" sTime="03:47:00" sDate="2017-02-17"
    eTime="04:38:00" eDate="2017-02-17">
    <affectedProduct name="RB" operator="DB Regio AG" />
  </Message>
  <Message id="3459" act="true" head="Bauarbeiten."
    text="Der Zug fällt zwischen Delitzsch unt Bf und Halle(Saale)Hbf aus. Ein Ersatzverkehr
von Delitzsch unt Bf nach Halle(Saale)Hbf ist eingerichtet. Bitte überprüfen Sie Ihre Verbin-
dung noch einmal kurz vor der Reise."
    priority="5" category="3" products="65535" sTime="05:57:00" sDate="2017-03-19"
    eTime="06:25:00" eDate="2017-03-19">
    <affectedProduct name="RB" operator="DB Regio AG" />
  </Message>
  <Message id="3460" act="true" head="Bauarbeiten."
    text="Der Zug fällt zwischen Dessau Hbf und Weißandt-Gölzau aus. Ein Ersatzverkehr von
Dessau Hbf nach Weißandt-Gölzau ist eingerichtet. Bitte überprüfen Sie Ihre Verbindung noch
einmal kurz vor der Reise."
    priority="5" category="3" products="65535" sTime="18:03:00" sDate="2017-03-20"
    eTime="19:27:00" eDate="2017-03-20">
    <affectedProduct name="RB" operator="DB Regio AG" />
  </Message>
…
</HimMessages>
```

2.15.3 Rendering RSS feed

If the RSS feature is enabled, HIM messages can be accessed in RSS 2.0 format by specifying rss as the output format. All other parameters are supported.

`<baseurl>/ himsearch?accessId=123&format=rss`

Result: (abbreviated)

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<rss version="2.0" xmlns:dc="http://purl.org/dc/elements/1.1/">
    <channel>
        <title>Verkehrsinformation</title>
        <language>de</language>
        <pubDate>Thu, 7 Jun 2018 15:14:49 +0200</pubDate>
        <item>
            <title>Bauarbeiten</title>
            <description>07.06.2018 03:47 - 04:38&lt;br/&gt;&lt;br/&gt;Der Zug fällt zwischen
Dessau Hbf und Bitterfeld aus. Ein Ersatzverkehr von Dessau Hbf nach Bitterfeld ist eingerich-
tet. Bitte überprüfen Sie Ihre Verbindung noch einmal kurz vor der Reise.</description>
            <pubDate>Thu, 7 Jun 2018  03:17:00 +0200</pubDate>
            <category domain="validityBegin">2018-06-07 03:47:00</category>
            <category domain="validityEnd">2018-06-07 04:38:00</category>
        </item>
    </channel>
</rss>
```

2.16 Real time archive gateway

This service provides a gateway to your real time archive. It identifies a trip in you current schedule data and ask the real time archive for the archived real time states. The result is presented as a Journey Detail Service result.

2.16.1 Request parameters

Name	Use	Range	Default	Description
accessId	Mandatory	-	-	Access ID for identifying the requesting client.
id	Mandatory	See 2.3.2 or 2.3.3	-	Specifies the internal journey id of the journey shall be retrieved. It may be necessary to escape the `|` character by its URL encoding `%7C`.
date	Mandatory	See 1.2.2	-	Day of operation. Represented in the format YYYY-MM-DD.

2.16.2 Example

Request: Get the real time history for journey with id `1|26391|7|80|29122016`

```
<baseurl>/rtarchive?id=1|26391|7|80|29122016&date=2016-12-29&accessId=123
```

Result: (abbreviated)

```
<?xml version="1.0" encoding="UTF-8"?>
<JourneyDetail serverVersion="1.1" dialectVersion="1.23" xmlns="hafas_rest">
  <Stops>
    <Stop routeIdx="0" extId="3006907" name="Wiesbaden Hauptbahnhof"
      depDate="2016-12-29" depTime="11:49:00" rtDepDate="2016-12-29"
      rtDepTime="11:49:00" lon="8.244636" lat="50.069485" />
    <Stop routeIdx="1" extId="3006906"
      name="Wiesbaden-Biebrich Bahnhof Wiesbaden Ost" arrDate="2016-12-29"
      arrTime="11:53:00" rtArrDate="2016-12-29" rtArrTime="11:55:00"
      depDate="2016-12-29" depTime="11:53:00" rtDepDate="2016-12-29"
      rtDepTime="11:55:00" lon="8.256448" lat="50.041726" />
    <Stop routeIdx="2" extId="3006905" name="Mainz Nordbahnhof"
      arrDate="2016-12-29" arrTime="11:57:00" rtArrDate="2016-12-29"
      rtArrTime="11:57:00" depDate="2016-12-29" depTime="11:57:00"
      rtDepDate="2016-12-29" rtDepTime="11:57:00" lon="8.245607" lat="50.020017" />
    <Stop routeIdx="3" extId="3006904" name="Mainz Hauptbahnhof"
      arrDate="2016-12-29" arrTime="12:01:00" rtArrDate="2016-12-29"
      rtArrTime="12:01:00" depDate="2016-12-29" depTime="12:03:00"
      rtDepDate="2016-12-29" rtDepTime="12:03:00" lon="8.258453" lat="50.001436" />
    <Stop routeIdx="4" extId="3006902" name="Mainz Römisches Theater Bahnhof"
      arrDate="2016-12-29" arrTime="12:05:00" rtArrDate="2016-12-29"
      rtArrTime="12:06:00" depDate="2016-12-29" depTime="12:06:00"
      rtDepDate="2016-12-29" rtDepTime="12:07:00" lon="8.278283" lat="49.993355" />
    <Stop routeIdx="5" extId="3006901"
      name="Ginsheim-Gustavsburg Mainz-Gustavsburg Bf" arrDate="2016-12-29"
      arrTime="12:09:00" rtArrDate="2016-12-29" rtArrTime="12:10:00"
      depDate="2016-12-29" depTime="12:09:00" rtDepDate="2016-12-29"
      rtDepTime="12:10:00" lon="8.314483" lat="49.994353" />
    <Stop routeIdx="6" extId="3006999" name="Mainz-Bischofsheim Bahnhof"
      arrDate="2016-12-29" arrTime="12:11:00" rtArrDate="2016-12-29"
      rtArrTime="12:13:00" depDate="2016-12-29" depTime="12:12:00"
      rtDepDate="2016-12-29" rtDepTime="12:14:00" lon="8.358053" lat="49.993427" />
    <Stop routeIdx="7" extId="3004913" name="Rüsselsheim Opelwerk Bahnhof"
      arrDate="2016-12-29" arrTime="12:15:00" rtArrDate="2016-12-29"
      rtArrTime="12:17:00" depDate="2016-12-29" depTime="12:15:00"
      rtDepDate="2016-12-29" rtDepTime="12:17:00" lon="8.400590" lat="49.988213" />
```

```
    <Stop routeIdx="8" extId="3004912" name="Rüsselsheim Bahnhof"
      arrDate="2016-12-29" arrTime="12:17:00" rtArrDate="2016-12-29"
      rtArrTime="12:20:00" depDate="2016-12-29" depTime="12:18:00"
      rtDepDate="2016-12-29" rtDepTime="12:21:00" lon="8.414146" lat="49.992016" />
    <Stop routeIdx="9" extId="3004910" name="Raunheim Bahnhof"
      arrDate="2016-12-29" arrTime="12:20:00" rtArrDate="2016-12-29"
      rtArrTime="12:23:00" depDate="2016-12-29" depTime="12:21:00"
      rtDepDate="2016-12-29" rtDepTime="12:24:00" lon="8.454139" lat="50.009455" />
    <Stop routeIdx="10" extId="3002901" name="Kelsterbach Bahnhof"
      arrDate="2016-12-29" arrTime="12:26:00" rtArrDate="2016-12-29"
      rtArrTime="12:29:00" depDate="2016-12-29" depTime="12:26:00"
      rtDepDate="2016-12-29" rtDepTime="12:29:00" lon="8.529100" lat="50.063408" />
    <Stop routeIdx="11" extId="3002930"
      name="Frankfurt (Main) Flughafen Regionalbahnhof" arrDate="2016-12-29"
      arrTime="12:29:00" rtArrDate="2016-12-29" rtArrTime="12:33:00"
      depDate="2016-12-29" depTime="12:32:00" rtDepDate="2016-12-29"
      rtDepTime="12:35:00" lon="8.571430" lat="50.051210" />
    <Stop routeIdx="12" extId="3002899" name="Frankfurt (Main) Stadion"
      arrDate="2016-12-29" arrTime="12:36:00" rtArrDate="2016-12-29"
      rtArrTime="12:38:00" depDate="2016-12-29" depTime="12:36:00"
      rtDepDate="2016-12-29" rtDepTime="12:38:00" lon="8.632970" lat="50.068082" />
    <Stop routeIdx="13" extId="3001830" name="Frankfurt (Main) Niederrad Bahnhof"
      arrDate="2016-12-29" arrTime="12:39:00" rtArrDate="2016-12-29"
      rtArrTime="12:41:00" depDate="2016-12-29" depTime="12:39:00"
      rtDepDate="2016-12-29" rtDepTime="12:41:00" lon="8.637096" lat="50.080784" />
    <Stop routeIdx="14" extId="3007010" name="Frankfurt (Main) Hauptbahnhof tief"
      arrDate="2016-12-29" arrTime="12:43:00" rtArrDate="2016-12-29"
      rtArrTime="12:45:00" depDate="2016-12-29" depTime="12:44:00"
      rtDepDate="2016-12-29" rtDepTime="12:46:00" lon="8.662509" lat="50.107167" />
    <Stop routeIdx="15" extId="3000011" name="Frankfurt (Main) Taunusanlage"
      arrDate="2016-12-29" arrTime="12:46:00" rtArrDate="2016-12-29"
      rtArrTime="12:49:00" depDate="2016-12-29" depTime="12:46:00"
      rtDepDate="2016-12-29" rtDepTime="12:49:00" lon="8.668765" lat="50.113370" />
    <Stop routeIdx="16" extId="3000001" name="Frankfurt (Main) Hauptwache"
      arrDate="2016-12-29" arrTime="12:47:00" rtArrDate="2016-12-29"
      rtArrTime="12:50:00" depDate="2016-12-29" depTime="12:48:00"
      rtDepDate="2016-12-29" rtDepTime="12:51:00" lon="8.679292" lat="50.113963" />
    <Stop routeIdx="17" extId="3000510" name="Frankfurt (Main) Konstablerwache"
      arrDate="2016-12-29" arrTime="12:49:00" rtArrDate="2016-12-29"
      rtArrTime="12:51:00" depDate="2016-12-29" depTime="12:50:00"
      rtDepDate="2016-12-29" rtDepTime="12:52:00" lon="8.687859" lat="50.114691" />
    <Stop routeIdx="18" extId="3000525" name="Frankfurt (Main) Ostendstraße"
      arrDate="2016-12-29" arrTime="12:51:00" rtArrDate="2016-12-29"
      rtArrTime="12:53:00" depDate="2016-12-29" depTime="12:51:00"
      rtDepDate="2016-12-29" rtDepTime="12:53:00" lon="8.696605" lat="50.113370" />
    <Stop routeIdx="19" extId="3000903" name="Frankfurt (Main) Mühlberg"
      arrDate="2016-12-29" arrTime="12:53:00" rtArrDate="2016-12-29"
      rtArrTime="12:55:00" depDate="2016-12-29" depTime="12:53:00"
      rtDepDate="2016-12-29" rtDepTime="12:55:00" lon="8.699706" lat="50.101900" />
    <Stop routeIdx="20" extId="3011263" name="Offenbach (Main) Kaiserlei S-Bahn"
      arrDate="2016-12-29" arrTime="12:56:00" rtArrDate="2016-12-29"
      rtArrTime="12:59:00" depDate="2016-12-29" depTime="12:56:00"
      rtDepDate="2016-12-29" rtDepTime="12:59:00" lon="8.737317" lat="50.105181" />
    <Stop routeIdx="21" extId="3011264"
      name="Offenbach (Main) S-Bahn-Station Ledermuseum" arrDate="2016-12-29"
      arrTime="12:58:00" rtArrDate="2016-12-29" rtArrTime="13:01:00"
      depDate="2016-12-29" depTime="12:58:00" rtDepDate="2016-12-29"
      rtDepTime="13:01:00" lon="8.749471" lat="50.105909" />
    <Stop routeIdx="22" extId="3011265" name="Offenbach (Main) Marktplatz S-Bahn"
      arrDate="2016-12-29" arrTime="12:59:00" rtArrDate="2016-12-29"
      rtArrTime="13:02:00" depDate="2016-12-29" depTime="13:00:00"
      rtDepDate="2016-12-29" rtDepTime="13:03:00" lon="8.765669" lat="50.105271" />
    <Stop routeIdx="23" extId="3002601" name="Offenbach (Main) Ost"
      arrDate="2016-12-29" arrTime="13:02:00" rtArrDate="2016-12-29"
      rtArrTime="13:05:00" lon="8.784843" lat="50.102817" />
  </Stops>

</JourneyDetail>
```

2.17 Time table info service

This service provides detailed information about all data sets (pools) loaded by the underlying HAFAS server.

Each pool carries an identification filed as well as the date and time of its physical creation and its type. The type can be `ST` for station, `ADR` for addresses and `POI` for point of interesst. If a type can not be determined, the value will be `U`.

2.17.1 Request parameters

Name	Use	Range	Default	Description
accessId	Mandatory	-	-	Access ID for identifying the requesting client.

2.17.2 Example

Request: `<baseurl>/tti?accessId=123`

Result:

```
<TimetableInfoList serverVersion="1.7.6" dialectVersion="1.23" requestId="1507536412387"
begin="2016-12-11" end="2018-12-08">
  <TimetableInfo ident="7mtpu" date="2017-10-05" time="10:50:08" type="ST"/>
  <TimetableInfo ident="65wha" date="2017-09-26" time="07:30:36" type="ST"/>
  <TimetableInfo ident="6jpdq" date="2017-09-14" time="07:50:04" type="ADR"/>
  <TimetableInfo ident="2x6uq" date="2017-07-06" time="09:19:28" type="POI"/>
</TimetableInfoList>
```

2.18 XSD Service

The `XSD` service will return a certain XML Schema Definition of a certain version. The URL parameter `name` specifies the requested XSD file. The version has to be specified in the ReST call as usual. Calling the `XSD` service with the single parameter list will return a list of all available XSD files in HTML format.

2.18.1 Example

Request: List all available XSD files

`<baseurl>/xsd?list`

Response:

- `rest-1.23.xsd`

Request: Return XSD

`<baseurl>/xsd?rest-1.23.xsd`

Result:

<Content of rest-1.23.xsd>

3 Responses

All services return their responses either in XML or JSON format (see 1.2.7). All possible response elements are defined in rest-1.23.xsd which could be retrieved via <baseURL>/xsd?**rest-1.23.xsd**.

The formats might be enhanced in the future so the implementation of the parsing should be implemented in view of future possible changes.

3.1 Location response

This is the response of the `location.name` and `location.nearbystops` services. The location consists of a list of entries, which are either stops/stations or named coordinates. The root element of the response is `LocationList`.

3.2 Trip Response

The trip response consists of a list of trips. Every trip has one to many legs with an origin and destination. The root element of the response is `TripList`. Trip services responds using that structure.

3.3 Departure board response

The departure board response contains a list of departures incl. all information concerning times, tracks, realtime data and journey. It also contains reference to get more details for the different journeys. The root element is `DepartureBoard`.

3.4 Arrival board response

The arrival board response contains a list of arrivals incl. all information concerning times, tracks, realtime data and journey. It also contains reference to get more details for the different journeys. The root element is `ArrivalBoard`.

3.5 Journey detail response

The journey detail response delivers all information about a single journey (vehicle route). It contains a list of stops including their indexes on the route and their coordinates. It contains also all times, tracks and real-time information if available for the whole route. It also contains the journeys name and type (there might be different names and types on parts of the journey). Finally it contains notes including information about their validity on segments of the total route.

3.6 Polyline response structure

Trip service responses may contain geometry parts in form of coded polyline structure.

4 Error codes and messages

If a request failed an error code and textual description will be returned. The error can be classified into several categories. The Rest API and Backend Server related errors are independent from the called service. Other errors depend on the called service.

If the service hits a time out, HTTP status code **504** is returned without any further description.

4.1 ReST Request Errors

Code	HTTP status code	Text
API_AUTH	403	access denied for 'key' on 'service'
API_QUOTA	400	quota exceeded for 'key' on 'service'
API_PARAM	400	required parameter <<name>> is missing
API_PARAM	400	numB wrong, only number in range [0,6] allowed
API_PARAM	400	numF wrong, only number in range [0,6] allowed
API_PARAM	400	numF + numB not greater than [6] allowed
API_FORMAT	400	response format not supported
SVC_PARAM	400	request parameter missing or invalid
SVC_LOC	400	location missing or invalid
SVC_LOC_ARR	400	arrival location missing or invalid
SVC_LOC_DEP	400	departure location missing or invalid
SVC_LOC_VIA	400	unknown change stop
SVC_LOC_EQUAL	400	start/destination or vias are equal
SVC_LOC_NEAR	400	start and destination to close
SVC_DATATIME	400	date/time missing or invalid
SVC_DATATIME_PERIOD	400	date/time not in timetable or allowed period
SVC_PROD	400	product field missing or invalid
SVC_CTX	400	context invalid
SVC_MAIL_ADR	400	sender/receiver mail address invalid or missing
SVC_MAIL	500	fail to send mail
SVC_SMS_NUM	400	receiver sms phone number invalid or missing
SVC_SMS	500	fail to send sms

SVC_FAILED_SEARCH	500	unsuccessful search
SVC_NO_RESULT	500	no result found
SVC_NO_MATCH	500	no match found
INT_ERR	500	internal error

5 Document Version

Date	Version	Author	Remarks
21.01.2014	1.2	mfr	initial version
12.03.2014	1.5	mfr	Extensions
30.04.2014	1.7	mfr	Extensions, e.g. period service
20.05.2014	1.9	mschu	Added and updated examples, added new response values for train type, train number and station UIC code, general overhaul
24.06.2014	1.10	mfr	Added Status service details. Added numF and numB parameters to Trip service. Added namespace to XML-response format.
25.08.2014	1.11	mfr	Added parameters to Trip Service. Added poly parameters. Response structure documentation completely replaced.
04.09.2014	1.12	mschu	Improved formatting to reduce page count.
22.09.2014	1.12	mfr	Add pre and post route parameters to trip and ifp service. Add operator filter to trip, ifp and station board services. Add avoid path to trip and ifp service. Add reconstruction service. Add alternative product filter to trip, ifp and station board services.
30.09.2014	1.12	mschu	Proofreading, Added more details to response parameters.
17.10.2014	1.13	mschu	Added information about train composition.
24.10.2014	1.14	mschu	Added new response values “weight” and “products” for the location response. Added an explanation how the station weight is calculated.
28.11.2014	1.15	mschu	Added products parameter to location.name and location.stopsnearby requests.
06.02.2015	1.16	mschu	Added description of additional parameters.
09.02.2015	1.17	mschu	Added new fields in Trip response description.

15.04.2015	1.20	rhu/mfr	Removing xsd from this document Location.name Service.Request: Filter “type” added Trip search service/Interval trip search service.Request: - parameter passlist added - parameters originExtId/destExtId added - parameter maxChange added - parameters originName/destName removed Trip search service/Interval trip search service.Response: - arrival and departure time added to the passlist - prognosis data for arrival and departure time added to the passlist - track data added to the passlist
17.04.2015	1.21	mfr	Stationboard service - Removing use* query parameters from station board service. - Add extId parameter Trip service - Add originExtId parameter - Add destExtId parameter Service overview - Add the description of this service General - Add error code R5000, access denied
13.08.2015	1.21	mfr	Error code consolidation
09.10.2015	1.22	mfr	New parameter on Location.nearbystops New parameters on Trip search New structure for Message result
08.03.2016	1.22.2	mfr	New parameter avoidId on Trip search
12.07.2016	1.23	mfr	Restructuring 2.1 Location services Restructuring 2.3 Interval trip search service Adding new parameter description to Trip search service, Reconstruction service, Station board services and Journey detail service.
12.09.2016	1.23.1	mfr	Description of XSD service changed.
23.09.2016	1.23.2	mfr	Fixed description of numF, numB. Add detailed description for location.name input parameter.

29.12.2016	1.23.3	mfr	Add service description for • Journey match • Train search • HIM search • Print to web gateway • Real time archive gateway • GIS route. Add description for request tracking. Consolidate trip search, interval trip search and station board parameters.
03.01.2017	1.23.4	mfr	Additional error code description.
10.01.2017	1.23.5	mfr	Corrected document structure. Corrected station board services introduction.
17.01.2017	1.23.6	mfr	Removed unused time parameter from Journey match and Train search services. Corrected description of date parameter of those two services. Additional description of rtMode options in Trip search
15.02.2017	1.23.7	fgel	Add sattributes filter description to trip search. Adjust operator and line filter description on trip and stationboard services.
07.04.2017	1.23.8	mfr	Add Time table info service. Add baim parameter to Trip search service.
12.05.2017	1.23.9	mfr	Add scroll description to station board service. Add eco, ecoCmp and ecoParams parapeters to reconstruction service. Add chapter for barrier free information. Add trainFilter to Trip Search service. Add direct train seach description to Trip Search. Add himcategory to HIM Search service.
08.06.2017	1.23.10	mfr	Add chapter capacity information. Add chapter mobility profiles.
14.06.2017	1.23.11	mfr	Add detailed description to originWalk, originBike, originCar, originTaxi, originPark, destWalk, destBike, destCar, destTaxi, destPark in Trip service Add more samples to via in Trip service
07.07.2017	1.23.12	mfr	Add detailed description for maxJourneys parameter of station board service. Add poly parameter to HIM search service. Time out error results in HTTP 504. Spelling in general.
18.08.2017	1.23.13	mfr	Add refinement description to Location.name service. Add unsharp parameter to Trip service. Add chapter “Unsharp search” for detailing.

01.09.2017	1.23.14	mfr	Remove parameter tripId from Journey match and Train search. Refine description of Journey match and Train search. Refine description of Station board services. Add economic and groupFilter parameter to Trip service. Add detailed description of result values for barrier free information and capacity utilization.
10.10.2017	1.23.15	mfr	Add description of coordinate request parameters to location.name service. Add type parameter description to location.nearbystops service. Add showPassingPoints description to JourneyMatch service.
22.12.2017	1.23.16	mfr	Add options to HIM search service.
15.02.2018	1.23.17	mfr	Changed documented time pattern to hh:mm[:ss].
27.03.2018	1.23.18	mfr	Add Search on trip service description Add blockingList, includeEarlier description to trip search service. Extend time table info service result sample.
08.05.2018	1.23.19	mfr	Add operators filter to Journey Match and Train Search service. Add viaId parameter to Search on trip service. Add withICTAlternatives to Trip service.
14.05.2018	1.23.20	mfr	Add ivOnly, totalWalk, totalBike, totalCar, totalTaxi to Trip service.
31.05.2018	1.23.21	mfr	Changed range of changeTimePercent in Trip service.
11.06.2018	1.23.22	mfr	Add rounding method to changeTimePercent.
22.06.2018	1.23.23	kha	Add Journey Validation service. Correct codes of 1.2.12 Capacity information.