HAFAS ReST API

Access to HAFAS journey planner systems

Version 2.12.0, 2020-10-01

Table of Contents

1. The interface 2
    1.1. Introduction 2
    1.2. General principles 3
2. Services 8
    2.1. Service overview 8
    2.2. Service description 8
    2.3. Location Search by Name (location.name) 8
    2.4. Location Search by Coordinate (location.nearbystops) 12
    2.5. Location Search in a bounding box (location.boundingbox) 14
    2.6. Location Search (location.search) 16
    2.7. Walking Links by Location (location.walkinglinks) 18
    2.8. Location Data (location.data) 22
    2.9. Location Details (location.details) 23
    2.10. Address Lookup (addresslookup) 24
    2.11. Trip Search (trip) 25
    2.12. Interval Trip Search (interval) 55
    2.13. Partial Trip Search (partialtripsearch) 56
    2.14. Partial Trip Search V2 (partialtripsearch.v2) 57
    2.15. M:N Search (manyToMany) 58
    2.16. Reconstruction (recon) 59
    2.17. Reconstruction Match (reconMatch) 63
    2.18. Search on Trip (sot) 68
    2.19. Trip Alternatives (trip.alternatives) 72
    2.20. GIS Route by Context (gisroute) 95
    2.21. Departure Board (departureBoard) 98
    2.22. Arrival Board (arrivalBoard) 103
    2.23. Journey Detail (journeyDetail) 107
    2.24. Journey Match (journeyMatch) 109
    2.25. Journey Position (journeyPos) 113
    2.26. Journey Validation (journeyValidation) 116
    2.27. Journey Track Match (journeyTrackMatch) 117
    2.28. Reachability search services (reachability) 122
    2.29. Train Search (trainSearch) 123
    2.30. Line Search (linesearch) 127
    2.31. Line Match (linematch) 128
    2.32. Line Info (lineinfo) 129
    2.33. Line Schedules (linesched) 132

2.34. HIM Search (himsearch) . . . 135
2.35. RSS Feed (feed) . . . 139
2.36. Real Time Archive Gateway (rtarchive) . . . 139
2.37. Data Information (datainfo) . . . 143
2.38. Time Table Information (tti) . . . 144
2.39. Walking Links (walkinglinks) . . . 144
2.40. Traffic Messages (Datex II) (trafficmessages/datex2) . . . 148
2.41. Download Service (downloads) . . . 148
2.42. XSD Service . . . 149
2.43. GraphQL Service . . . 149
3. Responses . . . 150
3.1. Location response . . . 150
3.2. Trip response . . . 150
3.3. Departure board response . . . 150
3.4. Arrival board response . . . 150
3.5. Journey detail response . . . 150
3.6. Polyline response structure . . . 150
4. Error codes and messages . . . 151
4.1. ReST Request Errors . . . 151
5. Document Version . . . 153

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.

HaCon Ingenieurgesellschaft mbH
Lister Straße 15
30163 Hannover
Germany

1. The interface

1.1. Introduction

1.1.1. Interface Overview

The public interface is implemented as a ReST [1: See http://rest.elkstein.org/ for a tutorial on ReST interfaces.] (**Re**presentational **S**tate **T**ransfer) interface which provides different methods for the different functionality of the journey planner, which are the following services:

- Location services
- Reachability search
- Station board services
- Trip Search
- Interval Trip Search
- Reconstruction
- Reconstruction Match
- Trip Alternatives
- GIS Route by Context
- Journey Detail
- Journey Match
- Journey Track Match
- Train Search
- Line service
- Print To Web Gateway
- Real Time Archive Gateway
- Time Table Information
- Data Information
- XSD
- GraphQL
- Status
- System Information

While Location, Trip Search, Interval Trip Search, Arrival Board and Departure Board services can be called directly, the Journey Detail service can only be called by a reference given in a result of the Trip Search, Departure Board or Arrival Board service. The Reconstruction service can only be called by a reference given

in a result from a Trip Search 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 behavior 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. Stop weight

In location services, each station or stop might have 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.

1.2.7. Versioning

Due to enhancements of the API input parameters as well as result structure will change over time. In case of API breaking changes, new versions will be provided as separate deployable packages.

To respect this, we suggest a URI layout including the version in the path info to access the API services:

`<baseUrl>/<version>/<servicename>`

Choosing this approach, different versions can exist in parallel without affecting each other.

1.2.8. Response Format

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

If XML is requested, the response will have the namespace `"http://hacon.de/hafas/proxy/hafas-proxy"`.

Preference for XML or JSON output can be signaled using the `Accept` Header:

`Accept: application/json` or `Accept: application/xml` with XML being the default.

Alternatively the `format` parameter overrides the requested type, even if the Accept Header is set.

In order to request a JSON response you have to set `Accept: application/json` or append the following parameter to each call of the interface: `format=json`.

JSONP is currently only supported by using `format=jsonp&jsonpCallback=<name_of_callback>`. The `jsonpCallback` parameter specifies the name of the function callback in which the json result is wrapped.

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 "value"
- The root element is directly converted into a JSON object (dropping the root elements name)

  <root><a>foo</root> becomes { "a": { "value" : "foo" } }
- Nested elements become nested properties

  <root><a>foo<c>foo</c></root>

  becomes

  { "a": { "b" : { "$": "foo" }, "c": { "value": "foo"} } }
- If there are multiple elements with the same name, the JSON code contains an array for these elements.

  <root><a>foo1foo2<root>

  becomes

  { "a": { "b" : [{"value": foo1" }, {"value": "foo2" }] } }
- Attribute names become object properties

  <root>foo2<root>

  becomes

  { "a": { "atb" : "foo1", "value" : "foo2" } }

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

XML:

```
<TripList serverVersion="..." dialectVersion="...">
    <Trip tripId="C-0">
        <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>
</TripList>
```

JSON:

```
{
    "serverVersion": "...",
    "dialectVersion": "...",
    "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" }
        }
    }]
}
```

1.2.9. Authentication

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

The authentication key can be passed either as parameter in the URL :

`accessId=<your_key_here>`

or by using the `Authorization` Header like this:

`Authorization: Bearer <your_key_here>`

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

1.2.10. 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

1.2.11. 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:

`<baseurl>/trip?…&requestId=123456789`

```
<?xml version="1.0" encoding="UTF-8"?>
<TripList serverVersion="1.5" dialectVersion="1.23" requestId="123456789"
xmlns="http://hacon.de/hafas/proxy/hafas-proxy">
...
</TripList>
```

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

1.2.12. OpenAPI Specification

If `enableOpenAPI: true` is set in the configuration file, the HAFAS Proxy will provide the following additional endpoints:

- `<baseurl>/api-doc` → JSON based Swagger 2.0 (OpenAPI) description of the available services
- `<baseurl>/swagger-ui` → Interactive API documentation generated from the OpenAPI description providing a `Swagger UI` based interface to the HAFAS Proxy.

2. Services

The list of services below describes all services provided by the HAFAS Proxy. 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 check 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 service: `<baseurl>/trip?wadl`

2.3. Location Search by Name (location.name)

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.

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).

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

2.3.1. 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	A, ALL, AP, P, S, SA, SP	ALL	Type filter for location types. ALL: search in all existing location pools S: Search for station/stops only A: Search for addresses only P: Search for POIs only SA: Search for station/stops and addresses SP: search for station/stops and POIs AP: search for addresses and POIs
locationSelectionMode	Optional	SLCT_A or SLCT_N	-	Selection mode for locations. SLCT_N: Not selectable SLCT_A: Selectable
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. Example: If 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".
coordLat	Optional	See Section 1.2.1	-	Latitude of centre coordinate.
coordLong	Optional	See Section 1.2.1	-	Longitude of centre coordinate.
r	Optional	-	1000	Search radius in meter around the given coordinate if any.
refineId	Optional	-	-	In case of an refinable location, this value takes the ID of the refinable one of a previous result.

meta	Optional	-	-	Filter by a predefined meta filter. If the rules of the predefined filter should not be negated, put ! in front of it.
stations	Optional	-	-	Filter for stations. Matches if the given value is prefix of any station in the itinerary. Multiple values are separated by comma.
sattributes	Optional	-	-	Filter locations by one or more attribute codes. Multiple attribute codes are separated by comma. If the attribute should not be part of the be location data, negate it by putting ! in front of it.
filterMode	Optional	DIST_PERI, EXCL_PERI, SLCT_PERI	DIST_PERI	Filter modes for nearby searches. DIST_PERI: accentuate matches EXCL_PERI: exclude not located SLCT_PERI: preselect

2.3.2. 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.

2.3.3. 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 2.4).

2.3.4. 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

2.3.5. 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>
```

2.4. Location Search by Coordinate (location.nearbystops)

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 center coordinate.

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

2.4.1. Request Parameters

Name	Use	Range	Default	Description
accessId	Mandatory	-	-	Access ID for identifying the requesting client.
originCoordLat	Mandatory	See Section 1.2.1	-	Latitude of centre coordinate.
originCoordLong	Mandatory	See Section 1.2.1	-	Longitude of centre coordinate.
r	Optional	-	1000	Search radius in meter around the given coordinate if any.
maxNo	Optional	1-1000	10	Maximum number of returned stops.

type	Optional	S, P, SP	S	Type filter for location types. Values are S: Search for station/stops only P: Search for POIs only SP: Search for stations/stops and POIs.
locationSelectionMode	Optional	SLCT_A or SLCT_N	-	Selection mode for locations. SLCT_N: Not selectable SLCT_A: Selectable
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. Example: If 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".
meta	Optional	-	-	Filter by a predefined meta filter. If the rules of the predefined filter should not be negated, put ! in front of it.
sattributes	Optional	-	-	Filter locations by one or more attribute codes. Multiple attribute codes are separated by comma. If the attribute should not be part of the be location data, negate it by putting ! in front of it.
sinfotexts	Optional	-	-	Filter locations by one or more station infotext codes and values. Multiple attribute codes are separated by comma the value by pipe |.

2.4.2. 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>
```

2.5. Location Search in a bounding box (location.boundingbox)

The `location.boundingbox` service returns all stops in a bounding box.

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

2.5.1. Request Parameters

Name	Use	Range	Default	Description
accessId	Mandatory	-	-	Access ID for identifying the requesting client.
llLat	Mandatory	See Section 1.2.1	-	Lower left latitude of bounding box.
llLon	Mandatory	See Section 1.2.1	-	Lower left longitude of bounding box.
urLat	Mandatory	See Section 1.2.1	-	Upper right latitude of bounding box.
urLon	Mandatory	See Section 1.2.1	-	Upper right longitude of bounding box.

type	Optional	S, P, SP	S	Type filter for location types. Values are S: Search for station/stops only P: Search for POIs only SP: Search for stations/stops and POIs.
locationSelectionMode	Optional	SLCT_A or SLCT_N	-	Selection mode for locations. SLCT_N: Not selectable SLCT_A: Selectable
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. Example: If 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".
meta	Optional	-	-	Filter by a predefined meta filter. If the rules of the predefined filter should not be negated, put ! in front of it.
sattributes	Optional	-	-	Filter locations by one or more attribute codes. Multiple attribute codes are separated by comma. If the attribute should not be part of the be location data, negate it by putting ! in front of it.
sinfotexts	Optional	-	-	Filter locations by one or more station infotext codes and values. Multiple attribute codes are separated by comma the value by pipe |.

2.5.2. Example

Request: Search for stations in the box [[1.500, 48.345], [3.301, 49.408]].

```
<baseurl>/location.boundingbox?llLon=1.500&llLat=48.345&urLon=3.301&urLat=49.408
```

Result:

```
<LocationList xmlns="http://hacon.de/hafas/proxy/hafas-proxy" serverVersion=
"2.9.3" dialectVersion="1.29" requestId="123456">
  <StopLocation id="A=1@O=Massy-TGV@X=2258854@Y=48725777@u=0@U=85@L=8739370@"
extId="8739370" name="Massy-TGV" lon="2.258854" lat="48.725777" weight="13"
dist="0" products="2">
    <productAtStop name="TGV" line="" catOut="TGV" cls="2" catOutS="TGV"
catOutL="Train à grande vit.">
      <icon>
        <foregroundColor r="255" g="255" b="255" hex="#FFFFFF"/>
        <backgroundColor r="224" g="0" b="0" hex="#E00000"/>
      </icon>
    </productAtStop>
  </StopLocation>
  <StopLocation id="A=1@O=Paris-
Montparnasse@X=2319577@Y=48840488@u=0@U=85@L=8739100@" extId="8739100" name=
"Paris-Montparnasse" lon="2.319577" lat="48.840488" dist="0" products="0"/>
  <StopLocation id="A=1@O=Paris-St-
Lazare@X=2325339@Y=48876328@u=0@U=85@L=8738400@" extId="8738400" name="Paris-
St-Lazare" lon="2.325339" lat="48.876328" dist="0" products="0"/>
  <StopLocation id="A=1@O=Paris-Nord@X=2354931@Y=48880886@u=0@U=85@L=8727100@"
extId="8727100" name="Paris-Nord" lon="2.354931" lat="48.880886" dist="0"
products="0"/>
...
</LocationList>
```

2.6. Location Search (location.search)

The `location.search` service returns filtered locations.

2.6.1. Request Parameters

Name	Use	Range	Default	Description
accessId	Mandatory	-	-	Access ID for identifying the requesting client.
maxNo	Optional	1-1000	10	Maximum number of returned stops.

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. Example: If 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".
sattributes	Optional	-	-	Filter locations by one or more attribute codes. Multiple attribute codes are separated by comma. If the attribute should not be part of the be location data, negate it by putting ! in front of it.
sinfotexts	Optional	-	-	Filter locations by one or more station infotext codes and values. Multiple attribute codes are separated by comma the value by pipe |.

2.6.2. Example

Request: `<baseurl>/location.search?sattributes=ZO&sinfotexts=RI|1&maxNo=2`

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<LocationList serverVersion="2.5.1-SNAPSHOT"
    dialectVersion="1.28" requestId="yxwwurpkmgw4928w"
    xmlns="http://hacon.de/hafas/proxy/hafas-proxy">
    <StopLocation id="A=1@O=ASC004@X=0@Y=0@U=70@L=990002@"
        extId="990002" name="ASC004" lon="0.0" lat="0.0" products="0">
        <LocationNotes>
            <LocationNote key="ZO" type="A" txtS="Zone name">Zone
name</LocationNote>
            <LocationNote key="RI" type="I" txtS="1">1</LocationNote>
        </LocationNotes>
    </StopLocation>
    <StopLocation id="A=1@O=ASC005@X=0@Y=0@U=70@L=990003@"
        extId="990003" name="ASC005" lon="0.0" lat="0.0" products="0">
        <LocationNotes>
            <LocationNote key="ZO" type="A" txtS="Zone name">Zone
name</LocationNote>
            <LocationNote key="RI" type="I" txtS="1">1</LocationNote>
        </LocationNotes>
    </StopLocation>
</LocationList>
```

2.7. Walking Links by Location (location.walkinglinks)

This service provides easy access to all walking links starting at a certain location. It returns a list (`WalkingLink`) along with its attributes, polyline and HIM messages if any.

Request Parameters

Name	Use	Range	Default	Description
accessId	Mandatory	-	-	Access ID for identifying the requesting client.
id	Optional	See Section 2.3 or Section 2.4	-	Specifies the station/stop ID for which the walking links shall be retrieved.
extId	Optional	-	-	**Deprecated. Please use `id` as it supports external IDs.** Specifies the station/stop extID for which the walking links shall be retrieved.

fattributes	Optional	-	-	Filter walking links by one or more attribute codes. Multiple attribute codes are separated by comma. If the attribute should not be part of the walking link, negate it by putting ! in front of it.

2.7.2. Example

Request: Get all walking links starting at id=320001076 having the attribute AU (elevator).

```
<baseurl>/location.walkinglinks?accessId=abc&id=320001076&fattributes=AU
```

Result:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<WalkingLinks serverVersion="2.10.0" dialectVersion="1.29" requestId=
"kcw82p7mk8gg9kw8"
  xmlns="http://hacon.de/hafas/proxy/hafas-proxy">
  <WalkingLink extId="110021168">
    <from id="A=1@O=Frankfurt (Main)
Hauptbahnhof@X=8663318@Y=50108030@U=80@L=320001076@" extId="320001076" name=
"Frankfurt (Main) Hauptbahnhof" lon="8.663318" lat="50.10803"/>
    <to id="A=1@O=Frankfurt (Main)
Hauptbahnhof@X=8663318@Y=50108039@U=80@L=320001075@" extId="320001075" name=
"Frankfurt (Main) Hauptbahnhof" lon="8.663318" lat="50.108039"/>
    <Note key="AU" type="A" priority="920" txtN="Aufzug">Aufzug</Note>
    <Note key="TB" type="A" priority="357" txtN="Taktile Bedienelemente am
Aufzug">Taktile Bedienelemente am Aufzug</Note>
    <Note key="aU" type="A" priority="352" txtN="Aufzug aufwärts">Aufzug
aufwärts</Note>
    <Note key="fD" type="A" priority="355" txtN="Tiefe des Aufzuges 203 cm"
>Tiefe des Aufzuges 203 cm</Note>
    <Note key="fE" type="A" priority="356" txtN="Maximale Belastung des
Aufzuges 1000 kg">Maximale Belastung des Aufzuges 1000 kg</Note>
    <Note key="fF" type="A" priority="950" txtN="Spiegel auf Gegenseite"
>Spiegel auf Gegenseite</Note>
    <Note key="gW" type="A" priority="290" txtN="mit Aufzug vom Bahnsteig U4
Enkheim/U5 Preungesheim (C-Ebene) zur Passage ( B-Ebene)">mit Aufzug vom
Bahnsteig U4 Enkheim/U5 Preungesheim (C-Ebene) zur Passage ( B-Ebene)</Note>
    <Note key="xq" type="A" priority="354" txtN="Aufzugbreite 90 cm"
>Aufzugbreite 90 cm</Note>
    <Note key="xs" type="A" priority="353" txtN="Lichte Breite des Aufzugs
(Tür) 90 cm">Lichte Breite des Aufzugs (Tür) 90 cm</Note>
    <PolylineGroup>
      <polylineDesc delta="true" dim="2" crdEncYX="evypHw`{s@A?" crdEncS="NN
"/>
```

```
    </PolylineGroup>
  </WalkingLink>
  <WalkingLink extId="110021168">
    <from id="A=1@O=Frankfurt (Main)
Hauptbahnhof@X=8663318@Y=50108039@U=80@L=320001075@" extId="320001075" name=
"Frankfurt (Main) Hauptbahnhof" lon="8.663318" lat="50.108039"/>
    <to id="A=1@O=Frankfurt (Main)
Hauptbahnhof@X=8663318@Y=50108030@U=80@L=320001076@" extId="320001076" name=
"Frankfurt (Main) Hauptbahnhof" lon="8.663318" lat="50.10803"/>
    <Note key="AU" type="A" priority="920" txtN="Aufzug">Aufzug</Note>
    <Note key="Au" type="A" priority="351" txtN="Aufzug abwärts">Aufzug
abwärts</Note>
    <Note key="TB" type="A" priority="357" txtN="Taktile Bedienelemente am
Aufzug">Taktile Bedienelemente am Aufzug</Note>
    <Note key="fD" type="A" priority="355" txtN="Tiefe des Aufzuges 203 cm"
>Tiefe des Aufzuges 203 cm</Note>
    <Note key="fE" type="A" priority="356" txtN="Maximale Belastung des
Aufzuges 1000 kg">Maximale Belastung des Aufzuges 1000 kg</Note>
    <Note key="fF" type="A" priority="950" txtN="Spiegel auf Gegenseite"
>Spiegel auf Gegenseite</Note>
    <Note key="gV" type="A" priority="290" txtN="mit Aufzug von Passage (B-
Ebene) zum  Bahnsteig U4 Enkheim/U5 Preungesheim (C-Ebene)">mit Aufzug von
Passage (B-Ebene) zum  Bahnsteig U4 Enkheim/U5 Preungesheim (C-Ebene)</Note>
    <Note key="xq" type="A" priority="354" txtN="Aufzugbreite 90 cm"
>Aufzugbreite 90 cm</Note>
    <Note key="xs" type="A" priority="353" txtN="Lichte Breite des Aufzugs
(Tür) 90 cm">Lichte Breite des Aufzugs (Tür) 90 cm</Note>
    <PolylineGroup>
      <polylineDesc delta="true" dim="2" crdEncYX="gvypHw`{s@@?" crdEncS="NN
"/>
    </PolylineGroup>
  </WalkingLink>
  <WalkingLink extId="110021169">
    <from id="A=1@O=Frankfurt (Main)
Hauptbahnhof@X=8663489@Y=50108138@U=80@L=320001078@" extId="320001078" name=
"Frankfurt (Main) Hauptbahnhof" lon="8.663489" lat="50.108138"/>
    <to id="A=1@O=Frankfurt (Main)
Hauptbahnhof@X=8663489@Y=50108156@U=80@L=320001077@" extId="320001077" name=
"Frankfurt (Main) Hauptbahnhof" lon="8.663489" lat="50.108156"/>
    <Note key="AU" type="A" priority="920" txtN="Aufzug">Aufzug</Note>
    <Note key="TB" type="A" priority="357" txtN="Taktile Bedienelemente am
Aufzug">Taktile Bedienelemente am Aufzug</Note>
    <Note key="aU" type="A" priority="352" txtN="Aufzug aufwärts">Aufzug
aufwärts</Note>
    <Note key="fD" type="A" priority="355" txtN="Tiefe des Aufzuges 203 cm"
>Tiefe des Aufzuges 203 cm</Note>
    <Note key="fE" type="A" priority="356" txtN="Maximale Belastung des
Aufzuges 1000 kg">Maximale Belastung des Aufzuges 1000 kg</Note>
```

```
    <Note key="fF" type="A" priority="950" txtN="Spiegel auf Gegenseite"
>Spiegel auf Gegenseite</Note>
    <Note key="gY" type="A" priority="290" txtN="mit Aufzug vom Bahnsteig U4
Bockenheimer Warte (C-Ebene) zur Passage (B-Ebene)">mit Aufzug vom Bahnsteig
U4 Bockenheimer Warte (C-Ebene) zur Passage (B-Ebene)</Note>
    <Note key="xq" type="A" priority="354" txtN="Aufzugbreite 90 cm"
>Aufzugbreite 90 cm</Note>
    <Note key="xs" type="A" priority="353" txtN="Lichte Breite des Aufzugs
(Tür) 90 cm">Lichte Breite des Aufzugs (Tür) 90 cm</Note>
    <PolylineGroup>
      <polylineDesc delta="true" dim="2" crdEncYX="{vypHya{s@C?" crdEncS="NN
"/>
    </PolylineGroup>
  </WalkingLink>
  <WalkingLink extId="110021169">
    <from id="A=1@O=Frankfurt (Main)
Hauptbahnhof@X=8663489@Y=50108156@U=80@L=320001077@" extId="320001077" name=
"Frankfurt (Main) Hauptbahnhof" lon="8.663489" lat="50.108156"/>
    <to id="A=1@O=Frankfurt (Main)
Hauptbahnhof@X=8663489@Y=50108138@U=80@L=320001078@" extId="320001078" name=
"Frankfurt (Main) Hauptbahnhof" lon="8.663489" lat="50.108138"/>
    <Note key="AU" type="A" priority="920" txtN="Aufzug">Aufzug</Note>
    <Note key="Au" type="A" priority="351" txtN="Aufzug abwärts">Aufzug
abwärts</Note>
    <Note key="TB" type="A" priority="357" txtN="Taktile Bedienelemente am
Aufzug">Taktile Bedienelemente am Aufzug</Note>
    <Note key="fD" type="A" priority="355" txtN="Tiefe des Aufzuges 203 cm"
>Tiefe des Aufzuges 203 cm</Note>
    <Note key="fE" type="A" priority="356" txtN="Maximale Belastung des
Aufzuges 1000 kg">Maximale Belastung des Aufzuges 1000 kg</Note>
    <Note key="fF" type="A" priority="950" txtN="Spiegel auf Gegenseite"
>Spiegel auf Gegenseite</Note>
    <Note key="gX" type="A" priority="290" txtN="mit Aufzug von Passage (B-
Ebene) zum Bahnsteig U4  Bockenheimer Warte (C-Ebene)">mit Aufzug von Passage
(B-Ebene) zum Bahnsteig U4  Bockenheimer Warte (C-Ebene)</Note>
    <Note key="xq" type="A" priority="354" txtN="Aufzugbreite 90 cm"
>Aufzugbreite 90 cm</Note>
    <Note key="xs" type="A" priority="353" txtN="Lichte Breite des Aufzugs
(Tür) 90 cm">Lichte Breite des Aufzugs (Tür) 90 cm</Note>
    <PolylineGroup>
      <polylineDesc delta="true" dim="2" crdEncYX="_wypHya{s@B?" crdEncS="NN
"/>
    </PolylineGroup>
  </WalkingLink>
</WalkingLinks>
```

2.8. Location Data (location.data)

The `location.data` service returns locations in a certain ID range. You can specify, if the service should return attributes, infotexts or product information.

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

2.8.1. Request Parameters

Name	Use	Range	Default	Description
accessId	Mandatory	-	-	Access ID for identifying the requesting client.
startId	Mandatory	-	-	Start ID of location id range.
endId	Mandatory	-	-	End ID of location id range.
attributes	Optional	0 or 1	0	Return locations with attributes.
infotexts	Optional	0 or 1	0	Return locations with infotexts.
products	Optional	0 or 1	0	Return locations with products.

2.8.2. Example

Request:
`<baseurl>/location.data?startId=800000001&endId=800000100&attributes=1&infotexts=1`

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<LocationList serverVersion="2.5.1-SNAPSHOT"
    dialectVersion="1.28" requestId="yxwwurpkmgw4928w"
    xmlns="http://hacon.de/hafas/proxy/hafas-proxy">
    <StopLocation id="A=1@O=ASC004@X=0@Y=0@U=70@L=800000001@"
        extId="800000001" name="ASC004" lon="0.0" lat="0.0" products="0">
        <LocationNotes>
            <LocationNote key="ZO" type="A" txtS="Zone name">Zone
name</LocationNote>
            <LocationNote key="RI" type="I" txtS="1">1</LocationNote>
        </LocationNotes>
    </StopLocation>
    <StopLocation id="A=1@O=ASC005@X=0@Y=0@U=70@L=800000002@"
        extId="800000002" name="ASC005" lon="0.0" lat="0.0" products="0">
        <LocationNotes>
            <LocationNote key="ZO" type="A" txtS="Zone name">Zone
name</LocationNote>
            <LocationNote key="RI" type="I" txtS="1">1</LocationNote>
        </LocationNotes>
    </StopLocation>
    ...
</LocationList>
```

2.9. Location Details (location.details)

The `location.details` service returns details of a specific location.

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

2.9.1. Request Parameters

Name	Use	Range	Default	Description
accessId	Mandatory	-	-	Access ID for identifying the requesting client.
id	Mandatory	-	-	Specifies the station/stop ID for which the details shall be retrieved.
date	Optional	-	-	Sets the start date for which the departures shall be retrieved. Represented in the format YYYY-MM-DD.

time	Optional	-	-	Sets the start time for which the departures shall be retrieved. Represented in the format hh:mm[:ss] in 24h nomenclature. Seconds will be ignored for requests.
attributes	Optional	0 or 1	1	Enables/disables return of location attribtues.
infotexts	Optional	0 or 1	1	Enables/disables return of location infotexts.
products	Optional	0 or 1	1	Enables/disables return of products served by this location.
messages	Optional	0 or 1	0	Enables/disables return of location messages.
tariffs	Optional	0 or 1	0	Enables/disables return of tariff information.

2.9.2. Example

Request:
<baseurl>/location.details?id=A=4@O=Bascharage@X=5924673@Y=49557989@U=105@L=000980017@

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<LocationList serverVersion="2.6.4-SNAPSHOT" dialectVersion="1.29" requestId=
"debug">
  <CoordLocation id="A=4@O=Bascharage@X=5924673@Y=49557989@U=105@L=000980017@"
extId="000980017" name="Bascharage" type="POI" lon="5.924673" lat="49.557989">
    <LocationNotes>
      <LocationNote key="Pb" type="A" txtS="Park&Ride">Park&
Ride</LocationNote>
      <LocationNote key="XM" type="A" txtS="X-Mode">X-Mode</LocationNote>
      <LocationNote key="ID" type="I" txtS="mmil-pr:pr.25">mmil-
pr:pr.25</LocationNote>
    </LocationNotes>
    <icon res="mmil_pr"/>
  </CoordLocation>
</LocationList>
```

2.10. Address Lookup (addresslookup)

The addresslookup service returns a list of possible addresses around a given center coordinate.

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

2.10.1. Request Parameters

Name	Use	Range	Default	Description
accessId	Mandatory	-	-	Access ID for identifying the requesting client.
originCoordLat	Mandatory	See Section 1.2.1	-	Latitude of coordinate.
originCoordLong	Mandatory	See Section 1.2.1	-	Longitude of coordinate.
maxNo	Optional	1-1000	10	Maximum number of returned stops.

2.10.2. Example

Request:
<baseurl>/addresslookup?originCoordLat=52.538289657517964&originCoordLong=13.450675105390859&maxNo=2

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<LocationList serverVersion="2.3.0" dialectVersion="1.25" requestId=
"vpw8zph2iww2w2ww"
    xmlns="http://hacon.de/hafas/proxy/hafas-proxy">
    <CoordLocation id="A=2@O=10409 Berlin-Prenzlauer Berg, Einsteinstr.
15D@X=13450640@Y=52538588@u=33@U=103@L=770017133@p=1524056779@" name="10409
Berlin-Prenzlauer Berg, Einsteinstr. 15D" type="ADR" lon="13.45064" lat=
"52.538588" dist="33">
        <icon res="ADR"/>
    </CoordLocation>
    <CoordLocation id="A=2@O=10407 Berlin-Prenzlauer Berg, Kniprodestr.
93@X=13450209@Y=52537707@u=72@U=103@L=770013502@p=1524056779@" name="10407
Berlin-Prenzlauer Berg, Kniprodestr. 93" type="ADR" lon="13.450209" lat=
"52.537707" dist="72">
        <icon res="ADR"/>
    </CoordLocation>
</LocationList>
```

2.11. Trip Search (trip)

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.

2.11.1. Request Parameters

Name	Use	Range	Default	Description
accessId	Mandatory	-	-	Access ID for identifying the requesting client.
originId	Optional	See Section 2.3 or Section 2.4	-	Specifies the station/stop ID of the origin for the trip. Such ID can be retrieved from the location.name or location.nearbystops services.
originExtId	Optional	-	-	**Deprecated. Please use `originId` as it supports external IDs.** 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	Optional	See Section 1.2.1 and Section 2.3 or Section 2.4	-	Latitude of station/stop coordinate of the trip's origin. The coordinate can be retrieved from the location.name or location.nearbystops services.
originCoordLong	Optional	See Section 1.2.1 and Section 2.3 or Section 2.4	-	Longitude of station/stop coordinate of the trip's origin. The coordinate can be retrieved from the location.name or location.nearbystops services.
originCoordName	Optional	-	-	Name of the trip's origin.
destId	Optional	See Section 2.3 or Section 2.4	-	Specifies the station/stop ID of the destination for the trip. Such ID can be retrieved from the location.name or location.nearbystops services.

destExtId	Optional	-	-	**Deprecated. Please use `destId` as it supports external IDs.** 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	Optional	See Section 1.2.1 and Section 2.3 or Section 2.4	-	Latitude of station/stop coordinate of the trip's destination. The coordinate can be retrieved from the location.name or location.nearbystops services.
destCoordLong	Optional	See Section 1.2.1 and Section 2.3 or Section 2.4	-	Longitude of station/stop coordinate of the trip's destination. The coordinate can be retrieved from the location.name or location.nearbystops services.
destCoordName	Optional	-	-	Name of the trip's destination.

via	Optional	-	-	Complex structure to provide multiple via points separated by semicolon.
				
				This structure is build like this:
				viaId|waittime|viastatus|products|direct|sleepingCar|couchetteCoach
				viaId: id, extId or altId of the via, mandatory. Routing via coordinates in idividual routing mode is possible as well: geo:x,y
				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
				direct: via section is direct (1) or not (0), optional, default 0
				sleepingCar: via section should include sleeping car (1), optional, default 0
				couchetteCoach: via section should include couchette coach (1), optional, default 0
				attributes: Filter via section by one or more attribute codes. Multiple attribute codes are separated by comma. If the attribute should not be part of the via section, negate it by putting ! in front of it.
				
				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 Section 2.3 or Section 2.4	-	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 Section 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 provide multiple points to be avoided separated by semicolon. This structure is build like this: avoidId|avoidstatus avoidId: id, extId or altId 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 Section 2.3 or Section 2.4	-	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	-	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 response, change time is presented in full minutes. If the calculation based on changeTime-Percent does not result in a full minute, it is rounded using "round half up" method.
minChangeTime	Optional	See Section 1.2.2	-	Minimum change time at stop in minutes.
maxChangeTime	Optional	See Section 1.2.2	-	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	-	Maximum number of changes.
date	Optional	See Section 1.2.2	-	Sets the start date for which the departures shall be retrieved. Represented in the format YYYY-MM-DD. By default the current server date is used
time	Optional	See Section 1.2.2	-	Sets the start time for which the departures shall be retrieved. Represented in the format hh:mm[:ss] in 24h nomenclature. Seconds will be ignored for requests. By default the current server time is used
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	0-6	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	0-6	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.
context	Optional	See Section 2.11.3	-	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 except any GIS route.
polyEnc	Optional	DLT, GPA, N	N	Defines encoding of the returned polyline. Possible values are N (no encoding / compression), DLT (delta to the previous coordinate), GPA (Google encoded polyline format) defaults to N. Not all option might be available in your installation.
passlist	Optional	0 or 1	0	Enables/disables the return of the passlist for each leg of the trip.
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. 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 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 be trip, negate it by putting ! in front of it. Example: Filter for operator A and B: `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 be 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 be trip, negate it by putting ! in front of it.
fattributes	Optional	All footway attribute codes from HAFAS raw data.	-	Filter trips by one or more footway attribute codes of a journey. Multiple attribute codes are separated by comma. If the attribute should not be part of the be 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 be trip, negate it by putting ! in front of it. This filter needs extended line data of HAFAS 5.40 in the back end.
lineids	Optional	-	-	Only journeys running the given line (identified by its line ID) are part of the result. To filter multiple lines, separate the line IDs by comma. If the line should not be part of the be 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 SW: Stairway EA: Elevator ES: Escalator RA: Ramp CB: Convey Belt Please note: Attribute codes may vary in your installation. Example: Use paths without ramp and stairway: `avoidPaths=SW,RA`.

originWalk	Optional	-	-	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 separated 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
				
				Be faster than normal: `1,0,1000,50`
				Be slower than normal: `1,0,1000,150`
				
				Bee line calculation
				0 (default) or 1
				
				Customer specific options can be set by appending a 'pipe' symbol followed by the comma-separated key-value pairs:
				`…|key1=value1,key2=value2`
				
				Example: footpaths with custom parameter cust1 having a value of 123:
				`originWalk=1|cust1=123`

originBike	Optional	-	-	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 separated 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 Be faster than normal: `1,0,1000,50` Be slower than normal: `1,0,1000,150` Bee line calculation 0 (default) or 1 Vehicle mode sharing, self (default) Provider enabled providers for vehicle mode, e.g. callabike, etc. Customer specific options can be set by appending a 'pipe' symbol followed by the comma-separated key-value pairs: `…|key1=value1,key2=value2` Example: bikesharing using call-a-bike having

originCar	Optional	-	-	Enables/disables using car in the beginning of a trip when searching from an address.
				`originBike=1,0,2500,100,0,sharing,callabike`
				To fine-tune the minimum and/or maximum distance to the next public transport station, provide these values separated 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=1,2000,100000`.
				
				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
				
				Be faster than normal: `1,0,1000,50`
				Be slower than normal: `1,0,1000,150`
				
				Bee line calculation
				0 (default) or 1
				
				Vehicle mode
				sharing, self (default)
				
				Provider
				enabled providers for vehicle mode, e.g. car2go, drivenow, etc.
				
				Customer specific options can be set by appending a 'pipe' symbol followed by the comma-separated key-value pairs:
				`…|key1=value1,key2=value2`
				
				Example: Carsharing using car2go having max. of 15km default speed:
				`originCar=1,0,15000,100,0,sharing,car2go`

originTaxi	Optional	-	-	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 separated 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 Be faster than normal: `1,0,1000,50` Be slower than normal: `1,0,1000,150` Bee line calculation 0 (default) or 1 Customer specific options can be set by appending a 'pipe' symbol followed by the comma-separated key-value pairs: `…|key1=value1,key2=value2`

originPark	Optional	-	-	Enables/disables using Park and 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 separated 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 Be faster than normal: `1,0,1000,50` Be slower than normal: `1,0,1000,150` Bee line calculation 0 (default) or 1 Customer specific options can be set by appending a 'pipe' symbol followed by the comma-separated key-value pairs: `…|key1=value1,key2=value2`
originMeta	Optional	-	-	Enables using one or more predefined individual transport meta profile at the beginning of a trip. The profiles are defined in the HAFAS installation.

destWalk	Optional	-	-	Enables/disables using footpaths at the end of a trip when searching to an address. To fine-tune the minimum and/or maximum distance to the next public transport station, provide these values separated 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 options Speed < 100: faster = 100: normal (default) > 100: slower Be faster than normal: `1,0,1000,50` Be slower than normal: `1,0,1000,150` Bee line calculation 0 (default) or 1 Customer specific options can be set by appending a 'pipe' symbol followed by the comma-separated key-value pairs: `…|key1=value1,key2=value2`

destBike	Optional	-	-	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 to the next public transport station, provide these values separated 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 options Speed < 100: faster = 100: normal (default) > 100: slower Be faster than normal: `1,0,1000,50` Be slower than normal: `1,0,1000,150` Bee line calculation 0 (default) or 1 Vehicle mode sharing, self (default) Provider enabled providers for vehicle mode, e.g. callabike, etc. Customer specific options can be set by appending a 'pipe' symbol followed by the comma-separated key-value pairs: `…|key1=value1,key2=value2` Example: bikesharing using call-a-bike having max. of 2.5km default speed: `destBike=1,0,2500,100,0,sharing,callabike`

destCar	Optional	-	-	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 to the next public transport station, provide these values separated 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=1,2000,100000`. 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 Be faster than normal: `1,0,1000,50` Be slower than normal: `1,0,1000,150` Bee line calculation 0 (default) or 1 Vehicle mode sharing, self (default) Provider enabled providers for vehicle mode, e.g. car2go, drivenow, etc. Customer specific options can be set by appending a 'pipe' symbol followed by the comma-separated key-value pairs: `…|key1=value1,key2=value2` Example: Carsharing using car2go having max. of 15km default speed: `destCar=1,0,15000,100,0,sharing,car2go`

destTaxi	Optional	-	-	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 to the next public transport station, provide these values separated 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 Be faster than normal: `1,0,1000,50` Be slower than normal: `1,0,1000,150` Bee line calculation 0 (default) or 1 Customer specific options can be set by appending a 'pipe' symbol followed by the comma-separated key-value pairs: `…|key1=value1,key2=value2`

destPark	Optional	-	-	Enables/disables using Park and Ride at the end of a trip when searching to an address. To fine-tune the minimum and/or maximum distance to the next public transport station, provide these values separated 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 Be faster than normal: `1,0,1000,50` Be slower than normal: `1,0,1000,150` Bee line calculation 0 (default) or 1 Customer specific options can be set by appending a 'pipe' symbol followed by the comma-separated key-value pairs: `…|key1=value1,key2=value2`
destMeta	Optional	-	-	Enables using one or more predefined individual transport meta profile at the end of a trip. The profiles are defined in the HAFAS installation.

totalWalk	Optional	-	-	Enables/disables using footpaths for the whole trip. To fine-tune the minimum and/or maximum distance to the next public transport station, provide these values separated 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 options Speed < 100: faster = 100: normal (default) > 100: slower Be faster than normal: `1,0,1000,50` Be slower than normal: `1,0,1000,150` Bee line calculation 0 (default) or 1 Customer specific options can be set by appending a 'pipe' symbol followed by the comma-separated key-value pairs: `…|key1=value1,key2=value2`

totalBike	Optional	-	-	Enables/disables using bike routes for the whole trip. To fine-tune the minimum and/or maximum distance to the next public transport station, provide these values separated 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 options Speed < 100: faster = 100: normal (default) > 100: slower Be faster than normal: `1,0,1000,50` Be slower than normal: `1,0,1000,150` Bee line calculation 0 (default) or 1 Vehicle mode sharing, self (default) Provider enabled providers for vehicle mode, e.g. callabike, etc. Customer specific options can be set by appending a 'pipe' symbol followed by the comma-separated key-value pairs: `…|key1=value1,key2=value2` Example: bikesharing using call-a-bike having max. of 2.5km default speed: `totalBike=1,0,2500,100,0,sharing,callabike`

totalCar	Optional	-	-	Enables/disables using car routes for the whole trip. To fine-tune the minimum and/or maximum distance to the next public transport station, provide these values separated 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=1,2000,100000`. 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 Be faster than normal: `1,0,1000,50` Be slower than normal: `1,0,1000,150` Bee line calculation 0 (default) or 1 Vehicle mode sharing, self (default) Provider enabled providers for vehicle mode, e.g. car2go, drivenow, etc. Customer specific options can be set by appending a 'pipe' symbol followed by the comma-separated key-value pairs: `…|key1=value1,key2=value2` Example: Carsharing using car2go having max. of 15km default speed: `totalCar=1,0,15000,100,0,sharing,car2go`

totalTaxi	Optional	-	-	Enables/disables using taxi rides for the whole trip. To fine-tune the minimum and/or maximum distance to the next public transport station, provide these values separated 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 Be faster than normal: `1,0,1000,50` Be slower than normal: `1,0,1000,150` Bee line calculation 0 (default) or 1 Customer specific options can be set by appending a 'pipe' symbol followed by the comma-separated key-value pairs: `…|key1=value1,key2=value2`
totalMeta	Optional	-	-	Enables using one or more predefined individual transport meta profile for a trip. The profiles are defined in the HAFAS installation.
includeIv	Optional	0 or 1	0	Enables/disables search for individual transport routes.

ivOnly	Optional	0 or 1	0	Enables/disables search for individual transport routes only.
mobilityProfile	Optional	-	-	Use a predefined filter by its name. The filters are defined in the HAFAS installation. If the filter should be negated, put a ! in front of its 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. This will only work in combination with `maxChange=0` as those trips are always meant to be direct connections.
sleepingCar	Optional	0 or 1	0	Enables/disables search for trips having sleeping car. This will only work in combination with `maxChange=0` as those trips are always meant to be direct connections.
couchetteCoach	Optional	0 or 1	0	Enables/disables search for trips having couchette coach. This will only work in combination with `maxChange=0` as those trips are always meant to be direct connections.
showPassingPoints	Optional	0 or 1	0	Enables/disables the return of stops having no alighting and boarding in its passlist for each leg of the trip. Needs passlist enabled.
baim	Optional	0 or 1	0	Enables/disables 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	FULL, INFOS, OFF, REALTIME, SERVER_DEFAULT	-	Set the realtime mode to be used. 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. For details, see Section 2.11.2.1

trainFilter	Optional	-	-	Filters a trip search for a certain train. First hit will be taken
economic	Optional	0 or 1	0	Enables/disables economic search mode. For details, see Section 2.11.2.2
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.
trainComposition	Optional	0 or 1	0	Enables/disables train composition data.
includeEarlier	Optional	0 or 1	0	Disables search optimization in relation of duration.
withICTAlternatives	Optional	0 or 1	0	Enables/disables the search for alternatives with individualized change times (ICT).
tariff	Optional	0 or 1	-	Enables/disables the output of tariff data. The default is configurable via provisioning.
trafficMessages	Optional	0 or 1	-	Enables/disables the output of traffic messages. The default is configurable via provisioning.

2.11.2. Search algorithm

The numB and numF parameters indicate the minimum number of search results returned by the service. Please note, that the effect of the parameters numF and numB depends on the search direction. If the search direction is in the past (backward search: ReST parameter: searchForArrival=1), the parameter numF acts on this search direction. That means the server return not less than the latest n (numF=n) connections before the requested arrival time. If the search direction is in the future (forward search: ReST parameter:

searchForArrival=0), then the server returns with the same parameter (numF=n) not less than the next n possible connections in the future. The corresponding applies to the numB parameter.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 connections 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.

Limitations:

The parameter numF must be > 0 for both forward and backward search. If you omit this parameter the default numF=3 is used. If you set numF=0, the proxy returns an error message. If you omit the parameter numB, the default numB=0 is used as well.

Depending on whether a special server configuration for the scrolling functionality ("scrolling stack") is active or not, the usage of the combination of the parameters numB and numF isn't allowed.

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.

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 ex-clude 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.

2.11.3. 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.

2.11.4. 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).

2.11.5. 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

2.11.6. Via segments

To have best control on via segments, the via parameter supports the following options:
viaId|waittime|viastatus|products|direct|sleepingCar|couchetteCoach

- viaId: id or extId of the via, mandatory
- waittime: waiting time spent at via station in minutes, optional
- viastatus, optional but defaults to EXR:

    - EXR (boarding and alighting necessary)
    - NER (boarding not necessary)
    - NXR (alighting not necessary)
    - NEXR (boarding and alighting not necessary)
- products: products used at the via, optional
- direct: via section is direct (1) or not (0), optional, default 0
- sleepingCar: via section should include sleeping car (1), optional, default 0
- couchetteCoach: via section should include couchette coach (1), optional, default 0
- attributes: Filter via section by one or more attribute codes. Multiple attribute codes are separated by comma. If the attribute should not be part of the via section, negate it by putting ! in front of it.

Direct connection on via segment

In combination with parameter maxChange there are the following options:

- Nothing set: no direct connection forced
- maxChange=0: direct connection
- maxChange=0 and via=1234: direct connection fort he first segment. After the via stop, interchanges are possible
- maxChange=0 and via=1234|||1: direct connection on the first and the second segment
- via=1234|||1: interchanges possible to reach the via stop. After the via, it is a direct connection

2.11.7. HAFAS Time Machine integration

You are able to use the Trip Search service to find connections with respect of their reliability within the HAFAS Time Machine setup.

General approach

A reconstructed connection should be checked regarding its reliability. The goal is to determine between three main categories:

- Guaranteed working connection
- Guaranteed not working connection
- Connection to be checked further
    - Connection possibly working
    - Connection possibly not working, but alternative connection available
    - Connection possibly not working

Response structure

The TripType structure is extended by the element reliability of type ConnectionReliabilityType defined like here

```
<xs:complexType name="ConnectionReliabilityType">
    <xs:attribute name="original" type="ConnectionReliabilityValueType">
        <xs:annotation>
            <xs:documentation>Reliability of the connection itself regarding
            its realtime status including cancellations, delays etc. to the
            get to the destination in time. Used in time machine feature.
            </xs:documentation>
        </xs:annotation>
    </xs:attribute>
    <xs:attribute name="alternative" type="ConnectionReliabilityValueType">
        <xs:annotation>
            <xs:documentation>Reliability of an alternative connection to the
            original connection regarding its realtime status including
            cancellations, delays etc. Used in time machine feature.
            </xs:documentation>
        </xs:annotation>
    </xs:attribute>
</xs:complexType>
```

It provides information about the reliability of the original trip and the alternative one. Each can have one of the following values fitting the general approach mentioned above.

GUARANTEED

Guaranteed to get the user from A to B in time within the scope

HIGH

Likely to get the user from A to B in time within the scope

LOW

Unlikely to get the user from A to B in time within the scope

ABORTIVE

Definitely not going to get the user from A to B in time within the scope

UNDEF

No information

Defined in XSD as follows:

```
<xs:simpleType name="ConnectionReliabilityValueType">
    <xs:restriction base="xs:string">
        <xs:enumeration value="GUARANTEED">
            <xs:annotation>
                <xs:documentation>Guaranteed to get the user from A to B in
                time within the scope</xs:documentation>
            </xs:annotation>
        </xs:enumeration>
        <xs:enumeration value="HIGH">
            <xs:annotation>
                <xs:documentation>Likely to get the user from A to B in time
                within the scope</xs:documentation>
            </xs:annotation>
        </xs:enumeration>
        <xs:enumeration value="LOW">
            <xs:annotation>
                <xs:documentation>Unlikely to get the user from A to B in time
                within the scope</xs:documentation>
            </xs:annotation>
        </xs:enumeration>
        <xs:enumeration value="ABORTIVE">
            <xs:annotation>
                <xs:documentation>Definitely not going to get the user from
                A to B in time within the scope</xs:documentation>
            </xs:annotation>
        </xs:enumeration>
        <xs:enumeration value="UNDEF">
            <xs:annotation>
                <xs:documentation>No information</xs:documentation>
            </xs:annotation>
        </xs:enumeration>
    </xs:restriction>
</xs:simpleType>
```

2.12. Interval Trip Search (interval)

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.

2.12.1. Request Parameters

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

Name	Use	Range	Default	Description
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			

2.12.2. 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).

2.12.3. Request path

The service listens at `<baseurl>/interval`

2.13. Partial Trip Search (partialtripsearch)

2.13.1. Request Parameters

Name	Use	Range	Default	Description
accessId	Mandatory	-	-	Access ID for identifying the requesting client.
ctx	Mandatory	-	-	Specifies the reconstruction context.
psCtx	Mandatory	-	-	Specifies the partial search context.
psSupplChangeTime	Optional	-	-	Supplementary changetime at station.
poly	Optional	0 or 1	0	Enables/disables the calculation of the polyline for each leg of the trip except any GIS route.

polyEnc	Optional	DLT, GPA, N	N	Defines encoding of the returned polyline. Possible values are N (no encoding / compression), DLT (delta to the previous coordinate), GPA (Google encoded polyline format) defaults to N. Not all option might be available in your installation.
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 boarding in its passlist for each leg of the trip. Needs passlist enabled.
baim	Optional	0 or 1	0	Enables/disables 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.
tariff	Optional	0 or 1	-	Enables/disables the output of tariff data. The default is configurable via provisioning.

2.13.2. Example

Request:

`<baseurl>/partialtripsearch?`

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

2.14. Partial Trip Search V2 (partialtripsearch.v2)

The `Partial Trip Search` service, version 2. The request has to include a body with either XML or JSON encoded data according to the `PartialTripSearchRequest` element of the XSD.

This service is able to search for pre- and post-carriage of a trip. Input are a reconstruction context and a steady main run. The steady main run needs to be a journey, not an walk.

2.14.1. Request Parameters

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

Request Body

2.14.2. Example

Request: Search a connection with a not reconstructible post-carriage. Input is the original reconstruction context and the begin and end location of the steady part from Dreieich-Sprendlingen to Hauptbahnhof, Frankfurt a.M.

```
<baseurl>/partialtripsearch.v2?accessId=abc
```

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<PartialTripSearchRequest xmlns="http://hacon.de/hafas/proxy/hafas-proxy">
    <ctx>¶HKI¶T$A=1@O=Dreieich-
Sprendlingen@L=8005636@a=128@$A=1@O=Frankfurt(Main)Hbf@L=8000105@a=128@$202009
081047$202009081101$RB
15630$$1$$$$§W$A=1@O=Frankfurt(Main)Hbf@L=8000105@a=128@$A=1@O=Hauptbahnhof,
Frankfurt
a.M.@L=100010@a=128@$202009081101$202009081111$$$1$$$$§T$A=1@O=Hauptbahnhof,
Frankfurt a.M.@L=100010@a=128@$A=1@O=Griesheim Jägerallee, Frankfurt
a.M.@L=100112@a=128@$202009081115$202009081150$STR   21$$1$$$$</ctx>
    <psSettings>
        <partialSearchSegment>
            <beginLocation extId="8005636" time="10:47:00" />
            <endLocation extId="8000105" time="11:01:00" />
        </partialSearchSegment>
    </psSettings>
</PartialTripSearchRequest>
```

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

2.15. M:N Search (manyToMany)

The `M:N Search` service. The request has to include a body with either XML or JSON encoded data according to the `ManyToManyConnectionRequest` element of the XSD.

2.15.1. Request Parameters

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

Request Body

2.15.2. Example

Request: Reconstruct a connection based travel sections

`<baseurl>/manyToMany?accessId=abc`

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ManyToManyConnectionRequest xmlns="http://hacon.de/hafas/proxy/hafas-proxy"
datetime="2020-04-01T11:59:14">
    <origin id="A=2@O=Hannover - Südstadt, Heidornstraße
6@X=9759238@Y=52363100@U=99@L=980524093@B=1@p=1453900975@"/>
    <destination id="A=2@O=Hannover - List, Lister Kirchweg
16@X=9743444@Y=52394913@U=99@L=980522871@B=1@p=1453900975@"/>
    <gisProfile type="F" appliesTo="F" minDist="0" maxDist="2000" />
    <searchOptions rtMode="OFF" />
    <backPreselection>
        <GisProfile type="F" />
        <PreselectionNode id="BACK:FOOT:0:StationPreselected">
            <location id="A=1@O=Hannover Isernhagener
Straße@X=9739156@Y=52391911@U=80@L=000992091@B=1@p=1583385840@"/>
            <PreselectionEdge id=
"BACK:FOOT:0:ContainsDistanceAndDurationThatWillBeMirrored" dist="-100"/>
        </PreselectionNode>
    </backPreselection>
</ManyToManyConnectionRequest>
```

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

2.16. Reconstruction (recon)

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.

2.16.1. 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 except any GIS route.
polyEnc	Optional	DLT, GPA, N	N	Defines encoding of the returned polyline. Possible values are N (no encoding / compression), DLT (delta to the previous coordinate), GPA (Google encoded polyline format) defaults to N. Not all option might be available in your installation.
date	Optional	-	-	Sets the start date for which the departures shall be retrieved. Represented in the format YYYY-MM-DD. This parameter will force the ser-vice 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.
useCombinedComparison	Optional	0 or 1	-	Compare based on combined output name - false: Compare parameters (category, line, train number) individually
acceptGaps	Optional	0 or 1	-	Accept an incomplete description of the connection (with gaps) i.e. missing walks/transfers
allowDisabledStops	Optional	0 or 1	-	Should stops be allowed as start or destination if boarding or alighting is disabled?
allowDummySections	Optional	0 or 1	-	Allow a partial reconstruction that will not lead to a reconstruction failure if sections are not reconstructable. Instead, for theses inconstructable sections, dummy sections will be created in the result.

flagAllNonReachable	Optional	0 or 1	-	Should all non-reachable journeys be flagged (true), or only the first one encountered?
matchCatStrict	Optional	0 or 1	-	Should the category (Gattung) match exactly? Only applicable if useCombinedComparison is false
matchIdNonBlank	Optional	0 or 1	-	Should the train identifier (Zugbezeichner) without whitespace match?
matchIdStrict	Optional	0 or 1	-	Should the train identifier (Zugbezeichner) match exactly?
matchNumStrict	Optional	0 or 1	-	Should the train number (Zugnummer) match exactly? Only applicable if useCombinedComparison is false
matchRtType	Optional	0 or 1	-	Should the realtime type that journeys are based on (e.g. SOLL, IST, additional, deviation, ...) be considered?
arrL	Optional	-	-	Lower deviation in minutes within interval [0, 720] indicating "how much earlier than original arrival"
arrU	Optional	-	-	Upper deviation in minutes within interval [0, 720] indicating "how much later than original arrival"
depL	Optional	-	-	Lower deviation in minutes within interval [0, 720] indicating "how much earlier than original departure"
depU	Optional	-	-	Upper deviation in minutes within interval [0, 720] indicating "how much later than original departure"
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 boarding in its passlist for each leg of the trip. Needs passlist parameter enabled.

rtMode	Optional	FULL, INFOS, OFF, REALTIME, SERVER_DEFAULT	-	Set the realtime mode to be used. 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.
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. Values vary.
tariff	Optional	0 or 1	-	Enables/disables the output of tariff data. The default is configurable via provisioning.

trafficMessages	Optional	0 or 1	-	Enables/disables the output of traffic messages. The default is configurable via provisioning.

2.16.2. 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@$201409181430$20140
9181443§T$A=1@O=Varhaug@L=7602220@a=128@$A=1@O=Egersund@L=7602212@a=128@$201409181443
$201409181510$
3040$§T$A=1@O=Egersund@L=7602212@a=128@$A=1@O=Drammen@L=7601421@a=128@$201409181525$2
01409182152$
728$§T$A=1@O=Drammen@L=7601421@a=128@$A=1@O=Holmestrand@L=7601505@a=128@$201409182215
$201409182238$ R1O$§G@F$A=1@O=Holmestrand@L=7601505@a=128@$A=2@O=Holmestrand,
Langgaten 2@X=10312173@Y=59492256@u=60@a=128@$201409182238$201409182244
```

2.16.3. Response

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

2.16.4. Partial reconstruction

The parameter `allowDummySections` enables the reconstruction service to mark not reconstructible sections as dummy sections in the response but reconstruct any others.

2.16.5. HAFAS Time Machine integration

You are able to use the Reconstruction service to check a connection regarding its reliability within the HAFAS Time Machine setup.

For HAFAS Time Machine specific data structures in the response, see Section 2.11.7.

2.17. Reconstruction Match (reconMatch)

The `reconstructionMatch` service tries to reconstruct a journey based on a series of stops and additional information like departure Time, line number etc. The `reconstructionMatch` service is an exception to other services of the HAFAS Proxy since it can only be called using a HTTP Post Request. The request has to include a body with either XML or JSON encoded tracking data according to the `ReconstructionMatchRequest` element of the XSD.

2.17.1. Request Parameters

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

Request Body

ReconstructionMatchRequest

Field	Use	Range	Default	Description
Elements				
section	Mandatory	List of ReconstructionSectionDataTypes	-	The individual sections from which a reconstruction should be attempted
Attributes				
useCombinedComparison	Optional	xs:boolean	-	Compare based on combined output name - false: Compare parameters (category, line, train number) individually
acceptGaps	Optional	xs:boolean	-	Accept an incomplete description of the connection (with gaps) i.e. missing walks/transfers
allowDisabledStops	Optional	xs:boolean	-	Should stops be allowed as start or destination if boarding or alighting is disabled?
flagAllNonReachable	Optional	xs:boolean	-	Should all non-reachable journeys be flagged (true), or only the first one encountered?
matchCatStrict	Optional	xs:boolean	-	Should the category (Gattung) match exactly? Only applicable if useCombinedComparison is false
matchIdNonBlank	Optional	xs:boolean	-	Should the train identifier (Zugbezeichner) without whitespace match?
matchIdStrict	Optional	xs:boolean	-	Should the train identifier (Zugbezeichner) match exactly?
matchNumStrict	Optional	xs:boolean	-	Should the train number (Zugnummer) match exactly? Only applicable if useCombinedComparison is false
matchRtType	Optional	xs:boolean	-	Should the realtime type that journeys are based on (e.g. SOLL, IST, additional, deviation, ...) be considered?
arrL	Optional	xs:int	0-720	Lower deviation in minutes indicating "how much earlier than original arrival"
arrU	Optional	xs:int	0-720	Upper deviation in minutes indicating "how much later than original arrival"

Field	Use	Range	Default	Description
depL	Optional	xs:int	0-720	Lower deviation in minutes indicating "how much earlier than original departure"
depU	Optional	xs:int	0-720	Upper deviation in minutes indicating "how much later than original departure"

ReconstructionSectionDataType

Field	Use	Range	Default	Description
Elements				
Departure	Mandatory	Location	-	Beginning of this section
Arrival	Mandatory	Location	-	End of this section
Attributes				
type	Mandatory	JNY, WALK, TRSF, DEVI, GIS_FOOT, GIS_BIKE, GIS_PARK, GIS_KISS, GIS_TAXI	-	If set to true, result contains additional data about the match quality.
departureTime	Optional	xs:dateTime	-	Date and time of the departure.
departureTimeRt	Optional	xs:dateTime	-	Date and time of the actual departure.
arrivalTime	Optional	xs:dateTime	-	Date and time of the arrival.
arrivalTimeRt	Optional	xs:dateTime	-	Date and time of the actual arrival.
trainName	Optional	-	-	If known, the train name.
trainNumber	Optional	-	-	If known, the train number.
trainCategory	Optional	-	-	Product category.

Location

A location can be either a specific station or stop identified by an Id or a generic GeoLocation based on longitude and latitude.

Field	Use	Range	Default	Description
Attributes				
id	Optional	-	-	Internal Id of the stop/station.
extId	Optional	-	-	External Id of the stop/station.
name	Optional	-	-	Name of the stop/station/location.
lon	Optional	See Section 1.2.1	-	Latitude of the geographical position.
lat	Optional	See Section 1.2.1	-	Latitude of the geographical position.

2.17.2. Example

Request: Reconstruct a connection based travel sections

<baseurl>/reconMatch?accessId=abc&

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ReconstructionMatchRequest xmlns="http://hacon.de/hafas/proxy/hafas-proxy">
    <section trainName="7" type="JNY" departureTime="2019-09-09T16:29:00.000"
arrivalTime="2019-09-09T16:33:00.000">
        <Departure id="at:45:50819:0:1" />
        <Arrival id="at:45:50817:0:1" />
    </section>
</ReconstructionMatchRequest>
```

Result: (abbreviated)

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<TripList [...] xmlns="http://hacon.de/hafas/proxy/hafas-proxy">
    <Trip idx="0" tripId="C-0" ctxRecon="¶HKI¶T$A=1@O=Salzburg
Strubergasse@L=455081901@a=128@$A=1@O=Salzburg
Fasaneriestraße@L=455081701@a=128@$201909091629$201909091633$       7$$1$$$"
duration="PT4M" checksum="f4e6072f_3">
        <ServiceDays planningPeriodBegin="2019-06-01" planningPeriodEnd="2020-
05-31" sDaysR="Mo - Sa" sDaysI="nicht 26. Okt, 1. Nov, 24. bis 26. Dez 2019,
31. Dez, 1., 6. Jan, 13. Apr, 1., 21. Mai" sDaysB=
"BF3EEDFBF7EFDFBF7EEDFBF7EFDFBF7EFDFBE7AFDFBF7EFDFBF469CFBF7EFDFBF7EFDFBF7EFDF
BF3EFDEBF7EEDF8"/>
        <Freq waitMinimum="10"/>
        <LegList>
            <Leg idx="0" name="Obus 7" number="7" category="S01" type="JNY"
reachable="true" direction="Salzburg Salzachsee" duration="PT4M">
                <Origin name="Salzburg Strubergasse" type="ST" id=
"A=1@O=Salzburg Strubergasse@X=13033154@Y=47808913@U=81@L=455081901@" extId=
"455081901" lon="13.033154" lat="47.808913" routeIdx="19" prognosisType=
"PROGNOSED" time="16:29:00" date="2019-09-09" hasMainMast="true" mainMastId=
"A=1@O=Salzburg Strubergasse@X=13033100@Y=47808823@U=81@L=455081900@"
mainMastExtId="455081900">
                    <altId>at:45:50819:0:1</altId>
                    <mainMastAltId>at:45:50819</mainMastAltId>
                </Origin>
                <Destination name="Salzburg Fasaneriestraße" type="ST" id=
"A=1@O=Salzburg Fasaneriestraße@X=13029370@Y=47816617@U=81@L=455081701@"
extId="455081701" lon="13.02937" lat="47.816617" routeIdx="22" prognosisType=
"PROGNOSED" time="16:33:00" date="2019-09-09" hasMainMast="true" mainMastId=
"A=1@O=Salzburg Fasaneriestraße@X=13029325@Y=47816617@U=81@L=455081700@"
mainMastExtId="455081700">
                    <altId>at:45:50817:0:1</altId>
                    <mainMastAltId>at:45:50817</mainMastAltId>
                </Destination>
                <Notes>
```

```
                    <Note key="LF" type="A" priority="50" routeIdxFrom="19"
routeIdxTo="22" txtS="Niederflurfahrzeug">Niederflurfahrzeug</Note>
                </Notes>
                <JourneyDetailRef ref=
"2|#VN#0#ST#1573033536#PI#0#ZI#88851#TA#14#DA#90919#1S#455045802#1T#1604#LS#45
5081001#LT#1641#PU#81#RT#1#CA#S01#ZE#7#ZB#       7#"/>
                <Freq waitMinimum="10" waitMaximum="10" alternativeCount="13
"/>
                <JourneyStatus>P</JourneyStatus>
                <Product name="Obus 7" num="14854" line="7" lineId="svv-1-7-
j19-21" catIn="S01" catCode="7" cls="128" catOutS="S01" catOutL="Obus"
operator="Salzburg AG - OBus" admin="S10000" routeIdxFrom="19" routeIdxTo="22
">
                    <icon res="prod_bus" txtS="7">
                        <foregroundColor r="255" g="255" b="255" hex="#FFFFFF
"/>
                        <backgroundColor r="0" g="121" b="58" hex="#00793A"/>
                    </icon>
                </Product>
                <Stops>
                    <Stop name="Salzburg Strubergasse" id="A=1@O=Salzburg
Strubergasse@X=13033154@Y=47808913@U=81@L=455081901@" extId="455081901"
routeIdx="19" lon="13.033154" lat="47.808913" depTime="16:29:00" depDate=
"2019-09-09" hasMainMast="true" mainMastId="A=1@O=Salzburg
Strubergasse@X=13033100@Y=47808823@U=81@L=455081900@" mainMastExtId="
455081900">
                        <altId>at:45:50819:0:1</altId>
                        <mainMastAltId>at:45:50819</mainMastAltId>
                    </Stop>
                    <Stop name="Salzburg Gaswerkgasse" id="A=1@O=Salzburg
Gaswerkgasse@X=13033208@Y=47812042@U=81@L=455001301@" extId="455001301"
routeIdx="20" lon="13.033208" lat="47.812042" depTime="16:31:00" depDate=
"2019-09-09" arrTime="16:31:00" arrDate="2019-09-09" hasMainMast="true"
mainMastId="A=1@O=Salzburg
Gaswerkgasse@X=13033118@Y=47811242@U=81@L=455001300@" mainMastExtId="
455001300">
                        <altId>at:45:50013:0:1</altId>
                        <mainMastAltId>at:45:50013</mainMastAltId>
                    </Stop>
                    <Stop name="Salzburg Schule Lehen" id="A=1@O=Salzburg
Schule Lehen@X=13030727@Y=47814783@U=81@L=455081801@" extId="455081801"
routeIdx="21" lon="13.030727" lat="47.814783" depTime="16:32:00" depDate=
"2019-09-09" arrTime="16:32:00" arrDate="2019-09-09" hasMainMast="true"
mainMastId="A=1@O=Salzburg Schule
Lehen@X=13030871@Y=47814532@U=81@L=455081800@" mainMastExtId="455081800">
                        <altId>at:45:50818:0:1</altId>
                        <mainMastAltId>at:45:50818</mainMastAltId>
                    </Stop>
```

```
                    <Stop name="Salzburg Fasaneriestraße" id="A=1@O=Salzburg
Fasaneriestraße@X=13029370@Y=47816617@U=81@L=455081701@" extId="455081701"
routeIdx="22" lon="13.02937" lat="47.816617" arrTime="16:33:00" arrDate="2019-
09-09" hasMainMast="true" mainMastId="A=1@O=Salzburg
Fasaneriestraße@X=13029325@Y=47816617@U=81@L=455081700@" mainMastExtId=
"455081700">
                        <altId>at:45:50817:0:1</altId>
                        <mainMastAltId>at:45:50817</mainMastAltId>
                    </Stop>
                </Stops>
            </Leg>
        </LegList>
        <TariffResult>
            [...]
        </TariffResult>
    </Trip>
</TripList>
```

2.18. Search on Trip (sot)

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.

2.18.1. 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. Example: • ICE 827 • RE 48 • S1
firstStopId	Mandatory	See Section 2.3 or Section 2.4	-	First station/stop ID of itinerary. Such ID can be retrieved from the location.name or location.nearbystops services.

firstStopDate	Mandatory	See Section 1.2.2	-	Departure date at first station/stop. Represented in the format YYYY-MM-DD.
firstStopTime	Mandatory	See Section 1.2.2	-	Departure time at first station/stop. Represented in the format hh:mm[:ss] in 24h nomenclature. Seconds will be ignored for requests.
currentDate	Mandatory	See Section 1.2.2	-	Current date to be used in trip search. Represented in the format YYYY-MM-DD.
currentTime	Mandatory	See Section 1.2.2	-	Current time to be used in trip search. Represented in the format hh:mm[:ss] in 24h nomenclature. Seconds will be ignored for requests.
lastStopId	Mandatory	See Section 2.3 or Section 2.4	-	First station/stop ID of itinerary. Such ID can be retrieved from the location.name or location.nearbystops services.
lastStopDate	Mandatory	See Section 1.2.2	-	Arrival date at last station/stop. Represented in the format YYYY-MM-DD.
lastStopTime	Mandatory	See Section 1.2.2	-	Arrival time at last station/stop. Represented in the format hh:mm[:ss] in 24h nomenclature. Seconds will be ignored for requests.
destId	Mandatory	See Section 2.3 or Section 2.4	-	Station/stop ID of destination. Such ID can be retrieved from the location.name or location.nearbystops services.

via	Optional	-	-	Complex structure to provide multiple via points separated by semicolon.
				
				This structure is build like this:
				viaId|waittime|viastatus|products|direct|sleepingCar|couchetteCoach
				viaId: id, extId or altId of the via, mandatory. Routing via coordinates in idividual routing mode is possible as well: geo:x,y
				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
				direct: via section is direct (1) or not (0), optional, default 0
				sleepingCar: via section should include sleeping car (1), optional, default 0
				couchetteCoach: via section should include couchette coach (1), optional, default 0
				attributes: Filter via section by one or more attribute codes. Multiple attribute codes are separated by comma. If the attribute should not be part of the via section, negate it by putting ! in front of it.
				
				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 Section 2.3 or Section 2.4	-	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 ID can be retrieved from the location.name or location.nearbystops services. If via is used, viaId and viaWaitTime will have 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 be trip, negate it by putting ! in front of it. Example: Filter for operator A and B: `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. 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.
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 except any GIS route.
polyEnc	Optional	DLT, GPA, N	N	Defines encoding of the returned polyline. Possible values are N (no encoding / compression), DLT (delta to the previous coordinate), GPA (Google encoded polyline format) defaults to N. Not all option might be available in your installation.

2.19. Trip Alternatives (trip.alternatives)

Search for an alternative trips based on an existing trip represented by the reconstruction context of it.

Aside the reconstruction context, the service requires first and last stop as well.

2.19.1. Request Parameters

Name	Use	Range	Default	Description
accessId	Mandatory	-	-	Access ID for identifying the requesting client.
ctx	Mandatory	-	-	Specifies the reconstruction context.
originId	Mandatory	See Section 2.3 or Section 2.4	-	Specifies the station/stop ID of the origin for the trip. Such ID can be retrieved from the location.name or location.nearbystops services.
originExtId	Optional	-	-	**Deprecated. Please use `originId` as it supports external IDs.** 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.
destId	Mandatory	See Section 2.3 or Section 2.4	-	Specifies the station/stop ID of the destination for the trip. Such ID can be retrieved from the location.name or location.nearbystops services.

destExtId	Optional	-	-	**Deprecated. Please use `destId` as it supports external IDs.** 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.

via	Optional	-	-	Complex structure to provide multiple via points separated by semicolon.
				
				This structure is build like this:
				viaId|waittime|viastatus|products|direct|sleepingCar|couchetteCoach
				viaId: id, extId or altId of the via, mandatory. Routing via coordinates in idividual routing mode is possible as well: geo:x,y
				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
				direct: via section is direct (1) or not (0), optional, default 0
				sleepingCar: via section should include sleeping car (1), optional, default 0
				couchetteCoach: via section should include couchette coach (1), optional, default 0
				attributes: Filter via section by one or more attribute codes. Multiple attribute codes are separated by comma. If the attribute should not be part of the via section, negate it by putting ! in front of it.
				
				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 Section 2.3 or Section 2.4	-	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 Section 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 provide multiple points to be avoided separated by semicolon. 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 Section 2.3 or Section 2.4	-	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	-	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 response, change time is presented in full minutes. If the calculation based on changeTime-Percent does not result in a full minute, it is rounded using "round half up" method.
minChangeTime	Optional	See Section 1.2.2	-	Minimum change time at stop in minutes.
maxChangeTime	Optional	See Section 1.2.2	-	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	-	Maximum number of changes.
poly	Optional	0 or 1	0	Enables/disables the calculation of the polyline for each leg of the trip except any GIS route.
polyEnc	Optional	DLT, GPA, N	N	Defines encoding of the returned polyline. Possible values are N (no encoding / compression), DLT (delta to the previous coordinate), GPA (Google encoded polyline format) defaults to N. Not all option might be available in your installation.
passlist	Optional	0 or 1	0	Enables/disables the return of the passlist for each leg of the trip.

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. 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 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 be trip, negate it by putting ! in front of it. Example: Filter for operator A and B: `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 be 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 be trip, negate it by putting ! in front of it.
fattributes	Optional	All footway attribute codes from HAFAS raw data.	-	Filter trips by one or more footway attribute codes of a journey. Multiple attribute codes are separated by comma. If the attribute should not be part of the be 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 be 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 SW: Stairway EA: Elevator ES: Escalator RA: Ramp CB: Convey Belt Please note: Attribute codes may vary in your installation. Example: Use paths without ramp and stairway: `avoidPaths=SW,RA`.

originWalk	Optional	-	-	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 separated 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 Customer specific options can be set by appending a 'pipe' symbol followed by the comma-separated key-value pairs: `…|key1=value1,key2=value2` Example: footpaths with custom parameter cust1 having a value of 123: `originWalk=1|cust1=123`

originBike	Optional	-	-	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 separated 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.
				
				Customer specific options can be set by appending a 'pipe' symbol followed by the comma-separated key-value pairs:
				`…|key1=value1,key2=value2`
				
				Example: bikesharing using call-a-bike having max. of 2.5km default speed:
				`originBike=1,0,2500,100,0,sharing,callabike`

originCar	Optional	-	-	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 separated 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=1,2000,100000`. 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. Customer specific options can be set by appending a 'pipe' symbol followed by the comma-separated key-value pairs: `…|key1=value1,key2=value2` Example: Carsharing using car2go having max. of 15km default speed: `originCar=1,0,15000,100,0,sharing,car2go`

originTaxi	Optional	-	-	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 separated 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 Customer specific options can be set by appending a 'pipe' symbol followed by the comma-separated key-value pairs: `…|key1=value1,key2=value2`

originPark	Optional	-	-	Enables/disables using Park and 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 separated 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 Customer specific options can be set by appending a 'pipe' symbol followed by the comma-separated key-value pairs: `…|key1=value1,key2=value2`
originMeta	Optional	-	-	Enables using one or more predefined individual transport meta profile at the beginning of a trip. The profiles are defined in the HAFAS installation.

destWalk	Optional	-	-	Enables/disables using footpaths at the end of a trip when searching to an address. To fine-tune the minimum and/or maximum distance to the next public transport station, provide these values separated 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 options Speed < 100: faster = 100: normal (default) > 100: slower Bee line calculation 0 (default) or 1 Customer specific options can be set by appending a 'pipe' symbol followed by the comma-separated key-value pairs: `…|key1=value1,key2=value2`

destBike	Optional	-	-	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 to the next public transport station, provide these values separated 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 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. callabike, etc. Customer specific options can be set by appending a 'pipe' symbol followed by the comma-separated key-value pairs: `…|key1=value1,key2=value2` Example: bikesharing using call-a-bike having max. of 2.5km default speed: `destBike=1,0,2500,100,0,sharing,callabike`

destCar	Optional	-	-	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 to the next public transport station, provide these values separated 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=1,2000,100000`. 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. Customer specific options can be set by appending a 'pipe' symbol followed by the comma-separated key-value pairs: `…|key1=value1,key2=value2` Example: Carsharing using car2go having max. of 15km default speed: `destCar=1,0,15000,100,0,sharing,car2go`

destTaxi	Optional	-	-	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 to the next public transport station, provide these values separated 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 Customer specific options can be set by appending a 'pipe' symbol followed by the comma-separated key-value pairs: `…|key1=value1,key2=value2`

destPark	Optional	-	-	Enables/disables using Park and Ride at the end of a trip when searching to an address. To fine-tune the minimum and/or maximum distance to the next public transport station, provide these values separated 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 Customer specific options can be set by appending a 'pipe' symbol followed by the comma-separated key-value pairs: `...|key1=value1,key2=value2`
destMeta	Optional	-	-	Enables using one or more predefined individual transport meta profile at the end of a trip. The profiles are defined in the HAFAS installation.

totalWalk	Optional	-	-	Enables/disables using footpaths for the whole trip. To fine-tune the minimum and/or maximum distance to the next public transport station, provide these values separated 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 options Speed < 100: faster = 100: normal (default) > 100: slower Bee line calculation 0 (default) or 1 Customer specific options can be set by appending a 'pipe' symbol followed by the comma-separated key-value pairs: `…|key1=value1,key2=value2`

totalBike	Optional	-	-	Enables/disables using bike routes for the whole trip. To fine-tune the minimum and/or maximum distance to the next public transport station, provide these values separated 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 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. callabike, etc. Customer specific options can be set by appending a 'pipe' symbol followed by the comma-separated key-value pairs: `…|key1=value1,key2=value2` Example: bikesharing using call-a-bike having max. of 2.5km default speed: `totalBike=1,0,2500,100,0,sharing,callabike`

totalCar	Optional	-	-	Enables/disables using car routes for the whole trip. To fine-tune the minimum and/or maximum distance to the next public transport station, provide these values separated 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=1,2000,100000`. 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. Customer specific options can be set by appending a 'pipe' symbol followed by the comma-separated key-value pairs: `…|key1=value1,key2=value2` Example: Carsharing using car2go having max. of 15km default speed: `totalCar=1,0,15000,100,0,sharing,car2go`

totalTaxi	Optional	-	-	Enables/disables using taxi rides for the whole trip. To fine-tune the minimum and/or maximum distance to the next public transport station, provide these values separated 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 Customer specific options can be set by appending a 'pipe' symbol followed by the comma-separated key-value pairs: `...|key1=value1,key2=value2`
totalMeta	Optional	-	-	Enables using one or more predefined individual transport meta profile for a trip. The profiles are defined in the HAFAS installation.
includeIv	Optional	0 or 1	0	Enables/disables search for individual transport routes.
ivOnly	Optional	0 or 1	0	Enables/disables search for individual transport routes only.

mobilityProfile	Optional	-	-	Use a predefined filter by its name. The filters are defined in the HAFAS installation. If the filter should be negated, put a ! in front of its 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. This will only work in combination with `maxChange=0` as those trips are always meant to be direct connections.
sleepingCar	Optional	0 or 1	0	Enables/disables search for trips having sleeping car. This will only work in combination with `maxChange=0` as those trips are always meant to be direct connections.
couchetteCoach	Optional	0 or 1	0	Enables/disables search for trips having couchette coach. This will only work in combination with `maxChange=0` as those trips are always meant to be direct connections.
showPassingPoints	Optional	0 or 1	0	Enables/disables the return of stops having no alighting and boarding in its passlist for each leg of the trip. Needs passlist enabled.
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 available.

rtMode	Optional	FULL, INFOS, OFF, REALTIME, SERVER_DEFAULT	-	Set the realtime mode to be used. 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. For details, see Section 2.11.2.1
trainFilter	Optional	-	-	Filters a trip search for a certain train. First hit will be taken
economic	Optional	0 or 1	0	Enables/disables economic search mode. For details, see Section 2.11.2.2

groupFilter	Optional	-	-	Use a predefined group filter to query for certain modes.
trainComposition	Optional	0 or 1	0	Enables/disables train composition data.

2.19.2. Response

As a result, the service returns a list of alternative trips.

2.19.3. Request path

The service listens at `<baseurl>/trip.alternatives`

2.20. GIS Route by Context (gisroute)

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.

2.20.1. 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.
polyEnc	Optional	DLT, GPA, N	N	Defines encoding of the returned polyline. Possible values are N (no encoding / compression), DLT (delta to the previous coordinate), GPA (Google encoded polyline format) defaults to N. Not all option might be available in your installation.
eco	Optional	0 or 1	0	Enables/disables eco value calculation.

2.20.2. 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" 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$$"
    checksum="523B5244_4" tripId="C-0" duration="PT0S"
    ecoUrl="demo.hafas.de/bin/pub/bene/eco/query.exe/dn?&amp;
L=ns_hispeed_eco&amp;application=ECOLOGYIN-FO&amp;requestConnection=1
&amp;conReconstruction=1&amp;newEcologyValues=1&amp;ecoFrom=1062HD Amsterdam,
Cornelis Lelylaan 35&amp;ecoFromId=205996696&amp;ecoFromX=4834021
&amp;ecoFromY=52357940&amp;ecoTo=Amsterdam Le-lylaan&amp;
ecoToId=008400079&amp;ecoToX=4833913&amp;ecoToY=52357887&amp;VH=G@F$A=2@O=1062
HD Amsterdam, Cornelis Lelylaan
35@X=4834021@Y=52357940@u=9@a=128@$A=1@O=Amsterdam
Lelylaan@L=8400079@a=0@$201608251236$201608251236$$&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@20
00@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>
```

2.21. Departure Board (departureBoard)

The separture board can be retrieved by a call to the `departureBoard` service. This method will return the next departures (or less if not existing) 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 running the the last minute found even if the requested maximum was overrun.

2.21.1. Request Parameters

Name	Use	Range	Default	Description
accessId	Mandatory	-	-	Access ID for identifying the requesting client.
id	Optional	See Section 2.3 or Section 2.4	-	Specifies the station/stop ID for which the departures shall be retrieved. Required if extId is not present. Such ID can be retrieved from the location.name or location.nearbystops services.
extId	Optional	-	-	**Deprecated. Please use `id` as it supports external IDs.** Specifies the external station/stop ID for which the departures shall be retrieved. Required if id is not present. Such ID can be retrieved from the location.name or location.nearbystops services.
direction	Optional	See Section 2.3 or Section 2.4	-	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 Section 1.2.2	-	Sets the start date for which the departures shall be retrieved. Represented in the format YYYY-MM-DD. By default the current server date is used

time	Optional	See Section 1.2.2	-	Sets the start time for which the departures shall be retrieved. Represented in the format hh:mm[:ss] in 24h nomenclature. Seconds will be ignored for requests. By default the current server time is used
duration	Optional	0-1439	60	Set the interval size in minutes.
maxJourneys	Optional	-	-1	Maximum number of journeys to be returned. If no value is defined or -1, 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).
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. 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 be trip, negate it by putting ! in front of it. Example: Filter for operator A and B: `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 part of the be trip, negate it by putting ! in front of it. This filter needs extended line data of HAFAS 5.40 in the back end.
filterEquiv	Optional	0 or 1	1	**Use type instead** Enables/disables the filtering of equivalent marked stops.
attributes	Optional	-	-	Filter boards 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 result, negate it by putting ! in front of it.
platforms	Optional	-	-	Filter boards by platform. Multiple platforms are separated by comma.
rtMode	Optional	FULL, INFOS, OFF, REALTIME, SERVER_DEFAULT	-	Set the realtime mode to be used if enabled.
passlist	Optional	0 or 1	0	Include a list of all passed waystops?

type	Mandatory	DEP, DEP_EQUIVS, DEP_MAST, DEP_STATION	DEP	Set the station departure board type to be used. DEP: Departure board as configured in HAFAS DEP_EQUIVS: Departure board with all journeys at any masts and equivalent stops DEP_MAST: Departure board at mast DEP_STATION: Departure board with all journeys at any masts of the requested station

A response will return `DepartureBoard`. This will contain a list of departures with train/line number, type of transport, departure times (incl. real-time), departure stop/stations (might be different from requested stop), direction text and a track information if available. Every departure will also contain a reference to the Journey Detail Service.

2.21.2. Example

Request: Departure board for Hannover, Lister Platz on June 1st, 2020, at 6 pm

```
<baseurl>/departureBoard?id=A=1@O=Hannover Lister Platz (U)@&date=2020-06-
01&time=18:00
```

Result: (abbreviated)

```
<DepartureBoard xmlns="http://hacon.de/hafas/proxy/hafas-proxy">
    <Departure name="Stb 9" type="ST" stop="Hannover Lister Platz (U)"
        stopid="A=1@O=Hannover Lister Platz
(U)@X=9750545@Y=52388809@U=80@L=902021@"
        stopExtId="902021" time="18:00:00" date="2020-06-01" track="UG 2"
        reachable="true" direction="Empelde" trainNumber="9" trainCategory=
"Stb">
        <JourneyStatus>P</JourneyStatus>
        ...
    </Departure>
    <Departure name="Stb 9" type="ST" stop="Hannover Lister Platz (U)"
        stopid="A=1@O=Hannover Lister Platz
(U)@X=9750285@Y=52388738@U=80@L=972021@"
        stopExtId="972021" time="18:00:00" date="2020-06-01" track="UG 1"
        reachable="true" direction="Fasanenkrug" trainNumber="9"
trainCategory="Stb">
        <JourneyStatus>P</JourneyStatus>
        ...
    </Departure>
    <Departure name="Stb 3" type="ST" stop="Hannover Lister Platz (U)"
        stopid="A=1@O=Hannover Lister Platz
(U)@X=9750545@Y=52388809@U=80@L=902021@"
        stopExtId="902021" time="18:04:00" date="2020-06-01" track="UG 2"
        reachable="true" direction="Wettbergen" trainNumber="3" trainCategory
="Stb">
        <JourneyStatus>P</JourneyStatus>
        ...
    </Departure>
</DepartureBoard>
```

2.21.3. Scrolling

To scroll departure boards, following action is to be performed: Take the departure time of the last departure 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.

2.21.4. Type

The attribute `type` specifies the type of the departure location. Valid values are ST (stop/station), ADR (address), POI (point of interest), CRD (coordinate), MCP (mode change point) or HL (hailing point).

2.22. Arrival Board (arrivalBoard)

The arrival board can be retrieved by a call to the `arrivalBoard` service. This method will return the next 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 arrivals running the the last minute found even if the requested maximum was overrun.

2.22.1. Request Parameters

Name	Use	Range	Default	Description
accessId	Mandatory	-	-	Access ID for identifying the requesting client.
id	Optional	See Section 2.3 or Section 2.4	-	Specifies the station/stop ID for which the arrivals shall be retrieved. Required if extId is not present. Such ID can be retrieved from the location.name or location.nearbystops services.
extId	Optional	-	-	**Deprecated. Please use `id` as it supports external IDs.** Specifies the external station/stop ID for which the arrivals shall be retrieved. Required if id is not present. Such ID can be retrieved from the location.name or location.nearbystops services.
direction	Optional	See Section 2.3 or Section 2.4	-	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 Section 1.2.2	-	Sets the start date for which the departures shall be retrieved. Represented in the format YYYY-MM-DD. By default the current server date is used

time	Optional	See Section 1.2.2	-	Sets the start time for which the departures shall be retrieved. Represented in the format hh:mm[:ss] in 24h nomenclature. Seconds will be ignored for requests. By default the current server time is used
duration	Optional	0-1439	60	Set the interval size in minutes.
maxJourneys	Optional	-	-1	Maximum number of journeys to be returned. If no value is defined or -1, 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).
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. 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 be trip, negate it by putting ! in front of it. Example: Filter for operator A and B: `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 part of the be trip, negate it by putting ! in front of it. This filter needs extended line data of HAFAS 5.40 in the back end.
filterEquiv	Optional	0 or 1	1	**Use type instead** Enables/disables the filtering of equivalent marked stops.
attributes	Optional	-	-	Filter boards 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 result, negate it by putting ! in front of it.
platforms	Optional	-	-	Filter boards by platform. Multiple platforms are separated by comma.
rtMode	Optional	FULL, INFOS, OFF, REALTIME, SERVER_DEFAULT	-	Set the realtime mode to be used if enabled.
passlist	Optional	0 or 1	0	Include a list of all passed waystops?
type	Mandatory	ARR, ARR_EQUIVS, ARR_MAST, ARR_STATION	ARR	Set the station arrival board type to be used. ARR: Arrival board as configured in HAFAS ARR_EQUIVS: Arrival board with all journeys at any masts and equivalent stops ARR_MAST: Arrival board at mast ARR_STATION: Arrival board with all journeys at any masts of the requested station

A response will return `ArrivalBoard`. This will contain a list of arrivals with train/line number, type of transport, arrival times (incl. real-time), arrival stop/stations (might be different from requested stop), direction text and a track information if available. Every arrival will also contain a reference to the Journey Detail Service.

2.22.2. Example

Request: Arrival board for Hannover, Lister Platz on June 1st, 2020, at 6 pm

```
<baseurl>/arrivalBoard?id=A=1@O=Hannover Lister Platz (U)@&date=2020-06-01&time=18:00
```

Result: (abbreviated)

```
<ArrivalBoard xmlns="http://hacon.de/hafas/proxy/hafas-proxy">
    <Arrival name="Stb 9" type="ST" stop="Hannover Lister Platz (U)"
        stopid="A=1@O=Hannover Lister Platz
(U)@X=9750545@Y=52388809@U=80@L=902021@"
        stopExtId="902021" time="18:00:00" date="2020-06-01" track="UG 2"
        reachable="true" direction="Empelde" trainNumber="9" trainCategory=
"Stb">
        <JourneyStatus>P</JourneyStatus>
        ...
    </Arrival>
    <Arrival name="Stb 9" type="ST" stop="Hannover Lister Platz (U)"
        stopid="A=1@O=Hannover Lister Platz
(U)@X=9750285@Y=52388738@U=80@L=972021@"
        stopExtId="972021" time="18:00:00" date="2020-06-01" track="UG 1"
        reachable="true" direction="Fasanenkrug" trainNumber="9"
trainCategory="Stb">
        <JourneyStatus>P</JourneyStatus>
        ...
    </Arrival>
    <Arrival name="Stb 3" type="ST" stop="Hannover Lister Platz (U)"
        stopid="A=1@O=Hannover Lister Platz
(U)@X=9750545@Y=52388809@U=80@L=902021@"
        stopExtId="902021" time="18:04:00" date="2020-06-01" track="UG 2"
        reachable="true" direction="Wettbergen" trainNumber="3" trainCategory
="Stb">
        <JourneyStatus>P</JourneyStatus>
        ...
    </Arrival>
</ArrivalBoard>
```

2.22.3. Scrolling

To scroll arrival boards, following action is to be performed: Take the arrival time of the last 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 arrivals of the last minute found even if a maxJourneys value needs to be overrun.

2.22.4. Type

The attribute `type` specifies the type of the arrival location. Valid values are ST (stop/station), ADR (address), POI (point of interest), CRD (coordinate), MCP (mode change point) or HL (hailing point).

2.23. Journey Detail (journeyDetail)

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.

2.23.1. Request Parameters

Name	Use	Range	Default	Description
accessId	Mandatory	-	-	Access ID for identifying the requesting client.
id	Mandatory	See Section 2.3 or Section 2.4	-	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	Optional	See Section 1.2.2	-	Day of operation By default the current server date is used
poly	Optional	0 or 1	0	Enables/disables the calculation of the polyline for each leg of the trip except any GIS route.
polyEnc	Optional	DLT, GPA, N	N	Defines encoding of the returned polyline. Possible values are N (no encoding / compression), DLT (delta to the previous coordinate), GPA (Google encoded polyline format) defaults to N. Not all option might be available in your installation.

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.
rtMode	Optional	FULL, INFOS, OFF, REALTIME, SERVER_DEFAULT	-	Set the realtime mode to be used if enabled
fromId	Optional	-	-	Specifies the station/stop ID the partial itinerary shall start from.
fromIdx	Optional	-	-	Specifies the station/stop index the partial itinerary shall start from.
toId	Optional	-	-	Specifies the station/stop ID the partial itinerary shall end at.
toIdx	Optional	-	-	Specifies the station/stop index the partial itinerary shall end at.
baim	Optional	0 or 1	0	Enables/disables BAIM search and response.

2.23.2. 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>
```

2.24. Journey Match (journeyMatch)

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.

2.24.1. 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. **Even if optional, this field should be filled.** Example: • ICE 827 • RE 48 • S1
filters	Optional	-	-	Filter definitions. Read additional document if available for your installation.
stations	Optional	-	-	**Deprecated. Please use `station` or `uic`.** Filter for stations. Matches if the given value is prefix of any station in the itinerary. Multiple values are separated by comma.
station	Optional	-	-	Filter for station by extId. Matches if the given value part of at least one station in the itinerary.
uic	Optional	-	-	Filter for UIC prefix of stations. Matches if the given value part of at least one station id 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. If the operator should not be part of the be trip, negate it by putting ! in front of it. Example: Filter for operator A and B: `operators=A,B.`
lines	Optional	All line numbers or names from HAFAS raw data.	-	Filter for lines. To filter multiple lines, separate the names/numbers by comma. If the line should not be part of the be trip, negate it by putting ! in front of it. Example: Filter for line 2 and 5: `lines=2,5.`

infotexts	Optional	-	-	Filter journeys by one or more custom infotext filters. Multiple infotexts are separated by comma. Use this to filter for trains: RT|112233,RT|445566
date	Optional	See Section 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 considerably.**
time	Optional	See Section 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 boarding.

2.24.2. 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=
"FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFE7FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFF000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000" />
  <ServiceDays sDaysR="nicht täglich" sDaysI="14. Nov bis 10. Dez 2016
täglich"
    sDaysB=
"FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFF000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000" />
  <lastPassRouteIdx>0</lastPassRouteIdx>
  <lastPassStopRef>0</lastPassStopRef>
</JourneyDetail>
```

2.25. Journey Position (journeyPos)

The journey position 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.

2.25.1. Request Parameters

Name	Use	Range	Default	Description
accessId	Mandatory	-	-	Access ID for identifying the requesting client.
llLat	Mandatory	See Section 1.2.1	-	Lower left latitude of bounding box.
llLon	Mandatory	See Section 1.2.1	-	Lower left longitude of bounding box.
urLat	Mandatory	See Section 1.2.1	-	Upper right latitude of bounding box.
urLon	Mandatory	See Section 1.2.1	-	Upper right longitude of 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. Example: Filter for operator A and B: `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. 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.
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 be trip, negate it by putting ! in front of it.
jid	Optional	-	-	Filter journeys by one or more journey id. Multiple journey ids are separated by comma. If the journey id should not be part of the be 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 be trip, negate it by putting ! in front of it.
infotexts	Optional	-	-	Filter journeys by one or more custom infotext filters. Multiple infotexts are separated by comma. If the infotext should not be part of the be trip, negate it by putting ! in front of it.
maxJny	Optional	1-1000	1000	Maximum number of journeys in response.
date	Optional	See Section 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 considerably.**

time	Optional	See Section 1.2.2	-	Time the service operates according to scheduled data. If not provided, the whole day is taken into account.
positionMode	Optional	CALC, CALC_REPORT, REPORT_ONLY	REPORT_ONLY	Mode the used for position calculation REPORT_ONLY: Only get back reported positions. CALC_REPORT: Use reported position if available, calculate if not. CALC: Calculate all positions

2.25.2. Example

Request: Get details to all journeys inside a bounding box.

```
<baseurl>/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"
    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>
```

2.26. Journey Validation (journeyValidation)

The `journeyValidation` service allows testing if position information is available for pro-vided train numbers. This is done by filtering via infotext. It returns a list (`JourneyValidation`). Only queries using the RT infotexts filter are allowed, see Section 2.26.2 for a valid example.

2.26.1. Request Parameters

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

infotexts	Mandatory	-	-	Filter journeys by one or more custom infotext filters. Multiple infotexts are separated by comma. Use this to validate trains: RT|112233,RT|445566
llLat	Optional	See Section 1.2.1	-	Lower left latitude of bounding box.
llLon	Optional	See Section 1.2.1	-	Lower left longitude of bounding box.
urLat	Optional	See Section 1.2.1	-	Upper right latitude of bounding box.
urLon	Optional	See Section 1.2.1	-	Upper right longitude of bounding box.
date	Optional	See Section 1.2.2	-	Day of operation.
time	Optional	See Section 1.2.2	-	Time the service operates according to scheduled data.

2.26.2. Example

Request: Get details to all journeys inside a bounding box.

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

Result: (abbreviated)

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

2.27. Journey Track Match (journeyTrackMatch)

The `journeyTrackMatch` service tries to reconstruct a journey based on tracked geo-positions and additional provided information. If enabled, information regarding the quality of the found matches is added to the response. The `journeyTrackMatch` service is an exception to other services of the HAFAS Proxy since it can only be called using a HTTP Post Request. The request has to include a body with either XML or JSON encoded tracking data according to the `TrackData` element of the XSD.

2.27.1. Request Parameters

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

Request Body

TrackData

Field	Use	Range	Default	Description
Elements				
TrackPoint	Mandatory	List of TrackPoints	-	The individual Data points from which a reconstruction should be attempted
TrackSection	Optional	List of TrackSectionData	-	Additional Data relating to individual trip segments - this is used, if a given trip should be validated.
Attributes				
algorithm	Optional	-	AUTO	Allows selecting the backing algorithm, if multiple methods are configured.
calcMatchQuality	Optional	xs:boolean	false	If set to true, result contains additional data about the match quality.

TrackPoint

Field	Use	Range	Default	Description
Elements				
Location	Mandatory	List of Locations	-	The individual Data points from which a reconstruction should be attempted
Probability	Optional	List of Key-Value-Pairs	-	Additional information about the quality of the match. See Probabilities for more information.
Attributes				
timestamp	Optional	xs:dateTime	-	Date and time of the tracked coordinate.
seId	Optional	-	-	Tracking specific id (specific to the used algorithm).
source	Optional	BEACON, GPS or MOBILE	-	If set to true, result contains additional data about the match quality.
accuracy	Optional	xs:int	-	Accuracy of the tracked coordinate. Smaller values mean higher accuracy.
speed	Optional	xs:int	false	Speed in meters per second.
direction	Optional	xs:int	-	Direction corresponding to 0 to 360 degrees starting at north and increasing clockwise.

Field	Use	Range	Default	Description
trainName	Optional	-	-	If known, the train name.
lineName	Optional	-	-	If known, the train number.
product	Optional	-	-	Product mask.

Location

A location can be either a specific station or stop identified by an Id or a generic GeoLocation based on longitude and latitude.

Field	Use	Range	Default	Description
Attributes				
id	Optional	-	-	Internal Id of the stop/station.
extId	Optional	-	-	External Id of the stop/station.
name	Optional	-	-	Name of the stop/station/location.
lon	Optional	See Section 1.2.1	-	Latitude of the geographical position.
lat	Optional	See Section 1.2.1	-	Latitude of the geographical position.

TrackSectionData

Field	Use	Range	Default	Description
Elements				
Departure	Mandatory	-	-	Start location of the section - departure location.
Arrival	Mandatory	-	-	End location of the section - arrival location.
Attributes				
type	Mandatory	-	-	Mode of transportation.
matchTimeSpanBegin	Optional	-	-	Start of the match.
matchTimeSpanEnd	Optional	-	-	End of the match.
departureTime	Optional	-	-	Departure time based on schedule.
departureTimeRt	Optional	-	-	Actual departure time based.
arrivalTime	Optional	-	-	Arrival time based on schedule.
arrivalTimeRt	Optional	-	-	Actual arrival time.
trainName	Optional	-	-	Name of the stop/station/location.
trainNumber	Optional	-	-	Name of the stop/station/location.
trainCategory	Optional	-	-	Name of the stop/station/location.

Probabilities

Additional information regarding the journey can be added using probabilities. A probability entry is a combination of a given key and the likelihood as a whole number ranging from 0 (0% chance of this type of

event having happened) to 100 (100% chance). An entry does not provide implicit information on the probability of the accompening inverse event e.g. change (85%) does not imply no_change (15%).

Key	Description
in_vehicle	Probability of being inside a vehicle
station_enter	Probability of having entered a station
station_leave	Probability of having left a station
enter	Probability of having entered vehicle (public transportation)
leave	Probability of having left a vehicle (public transportation)
change	Probability of having changed vehicles (public transportation)
no_change	Probability of not having changed vehicles (public transportation)
by_foot	Probability of walking by foot
by_strain	Probability of using a subway / train
by_bus	Probability of using a bus
checkin	User checked in (should only take the probability of 100)
checkout	User checked out (should only take the probability of 100)
bein	Be-In detected
beout	Be-Out detected

2.27.2. Example

Request: Reconstruct a connection based on tracked GPS-Coordinates

The coordinates and timestamps are send as individual TrackPoint-Elements:

```
<baseurl>/journeyTrackMatch?accessId=abc&
```

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<TrackData xmlns="http://hacon.de/hafas/proxy/hafas-proxy">
    <TrackPoint timestamp="2018-12-12T11:38:00">
        <Location lon="8.540193" lat="47.378177" />
    </TrackPoint>
    <TrackPoint timestamp="2018-12-12T12:01:00">
        <Location lon="8.562387" lat="47.450379" />
    </TrackPoint>
</TrackData>
```

Result: (abbreviated)

```
<?xml version="1.0" encoding="UTF-8"?>
<JourneyTrackMatch [...]" xmlns="http://hacon.de/hafas/proxy/hafas-proxy">
    <MatchResult>
        <TrackMatchJourneyDetail>
```

```
            <JourneyDetail reachable="true" ref="1|20288|3|95|12122018|D:0">
                <Stops>
                    <Stop name="Zürich HB" id="A=1@O=Zürich
HB@X=8540193@Y=47378168@U=95@L=8503000@" extId="8503000" routeIdx="8" lon=
"8.540193" lat="47.378177" depTime="11:39:00" depDate="2018-12-12">
                        <Notes>
                            <Note key="RA" type="A" priority="999">
RT_BHF</Note>
                        </Notes>
                    </Stop>
                    <Stop name="Zürich Oerlikon" id="A=1@O=Zürich
Oerlikon@X=8544112@Y=47411527@U=95@L=8503006@" extId="8503006" routeIdx="9"
lon="8.544112" lat="47.411527" depTime="11:46:00" depDate="2018-12-12"
arrTime="11:45:00" arrDate="2018-12-12">
                        <Notes>
                            <Note key="RA" type="A" priority="999">
RT_BHF</Note>
                        </Notes>
                    </Stop>
                    <Stop name="Zürich Flughafen" id="A=1@O=Zürich
Flughafen@X=8562378@Y=47450379@U=95@L=8503016@" extId="8503016" routeIdx="10"
lon="8.562387" lat="47.450379" arrTime="11:51:00" arrDate="2018-12-12">
                        <Notes>
                            <Note key="RA" type="A" priority="999">
RT_BHF</Note>
                        </Notes>
                    </Stop>
                </Stops>
                <Names>
                    <Name name="IC     5" number="1517" category="IC"
routeIdxFrom="8" routeIdxTo="8">
                        <Product name="IC     5" num="1517" line="5" lineId=
"1_000011_5" catOut="IC      " catIn="IC" catCode="1" catOutS="IC" catOutL=
"InterCity" operatorCode="SBB" operator="Schweizerische Bundesbahnen SBB"
admin="000011"/>
                    </Name>
                </Names>
                <Directions>
                    <Direction routeIdxFrom="8" routeIdxTo="10">St.
Gallen</Direction>
                </Directions>
                <JourneyStatus>P</JourneyStatus>
            </JourneyDetail>
        </TrackMatchJourneyDetail>
    </MatchResult>
    [...]
</JourneyTrackMatch>
```

Request: Reconstruct a connection based on tracked GPS-Coordinates and Check-In / Check-Out

```
<baseurl>/journeyTrackMatch?accessId=abc&
```

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<TrackData xmlns="http://hacon.de/hafas/proxy/hafas-proxy">
    <TrackPoint timestamp="2018-12-12T11:37:00">
        <Location extId="8503000" />
        <Probability name="checkin" value="100" />
    </TrackPoint>
    <TrackPoint timestamp="2018-12-12T11:38:00">
        <Location lon="8.540193" lat="47.378177" extId="8503000" />
    </TrackPoint>
    <TrackPoint timestamp="2018-12-12T12:01:00">
        <Location lon="8.562387" lat="47.450379" extId="8503016" />
    </TrackPoint>
    <TrackPoint timestamp="2018-12-12T12:02:00">
        <Location extId="8503016" />
        <Probability name="checkout" value="100" />
    </TrackPoint>
</TrackData>
```

Result omitted, since it should be equal to the first example.

2.28. Reachability search services (reachability)

The `reachability` service can be used to perform a reachability search from a stop or coordinate to all stops reachable in a defined amount of time and changes.

The result is a list of possible stops with the duration and number of changes to reach each of them.

Request Parameters

Name	Use	Range	Default	Description
accessId	Mandatory	-	-	Access ID for identifying the requesting client.
originId	Optional	-	-	Specifies the station/stop ID of the origin for the search.
originCoordLat	Optional	-	-	Latitude of station/stop coordinate of the search's origin.
originCoordLong	Optional	-	-	Longitude of station/stop coordinate of the search's origin.

date	Optional	-	-	Sets the start date for which the departures shall be retrieved.
time	Optional	-	-	Sets the start time for which the arrivals shall be retrieved. Represented in the format hh:mm[:ss] in 24h nomenclature. Seconds will be ignored for requests.
maxChange	Optional	0-11	-	Maximum number of changes.
duration	Optional	1-1439	60	Set the interval size in minutes.
forward	Optional	0 or 1	1	If true, search forward else search backward. Default is forward
filterEndWalks	Optional	0 or 1	0	If true, end locations not direct reachable by public transport are left out
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.
operators	Optional	-	-	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, negate it by putting ! in front of it.
attributes	Optional	-	-	Filter boards 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 result, 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 included, negate it by putting ! in front of it.

2.29. Train Search (trainSearch)

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.

2.29.1. 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. **Even if optional, this field should be filled.** Example: • ICE 827 • RE 48 • S1
filters	Optional	-	-	Filter definitions. Read additional documentation if available for your installation.
stations	Optional	-	-	**Deprecated. Please use `station` or `uic`.** Filter for stations. Matches if the given value is prefix of any station in the itinerary. Multiple values are separated by comma.
station	Optional	-	-	Filter for station by extId. Matches if the given value part of at least one station in the itinerary.
uic	Optional	-	-	Filter for UIC prefix of stations. Matches if the given value part of at least one station id 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 operator A and B: operators=A,B.
lines	Optional	All line numbers or names from HAFAS raw data.	-	Filter for lines. To filter multiple lines, separate the names/numbers by comma. If the line should not be part of the be trip, negate it by putting ! in front of it. Example: Filter for line 2 and 5: `lines=2,5.`
date	Optional	See Section 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 considerably.**
time	Optional	See Section 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 boarding in its passlist for each leg of the trip. Needs passlist enbaled.

2.29.2. 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>
```

2.30. Line Search (linesearch)

The `linesearch` service will deliver information about all lines of a given operator. It returns a list (`LineList`) of matched lines .

2.30.1. Request Parameters

Name	Use	Range	Default	Description
accessId	Mandatory	-	-	Access ID for identifying the requesting client.
operators	Mandatory	-	-	List of operators seperated by comma.
date	Optional	-	-	Day of operation. Represented in the format YYYY-MM-DD.

2.30.2. Example

Request: Get the list of lines by operators on a given day of operation

```
<baseurl>/linesearch?accessId=abc&operators=S-Bahn%20Berlin&date=2019-06-01
```

Result: (abbreviated)

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<LineList serverVersion="2.5.0" dialectVersion="1.28" requestId=
"x82w5t5i4g8wyw44" xmlns="http://hacon.de/hafas/proxy/hafas-proxy">
  <Line lineId="4_08_____S1" lineName="S1" lineNameShort="S1">
    <Product name="S1" line="S1" lineId="4_08_____S1" catOut="S" catCode="16"
operator="S-Bahn Berlin">
      <icon res="prod_comm_t">
        <foregroundColor r="255" g="255" b="255"/>
        <backgroundColor r="67" g="152" b="68"/>
      </icon>
    </Product>
  </Line>
  <Line lineId="4_08_____S2" lineName="S2" lineNameShort="S2">
    <Product name="S2" line="S2" lineId="4_08_____S2" catOut="S" catCode="16"
operator="S-Bahn Berlin">
      <icon res="prod_comm_t">
        <foregroundColor r="255" g="255" b="255"/>
        <backgroundColor r="67" g="152" b="68"/>
      </icon>
    </Product>
  </Line>
  <Line lineId="4_08_____S3" lineName="S3" lineNameShort="S3">
    <Product name="S3" line="S3" lineId="4_08_____S3" catOut="S" catCode="16"
operator="S-Bahn Berlin">
      <icon res="prod_comm_t">
        <foregroundColor r="255" g="255" b="255"/>
        <backgroundColor r="67" g="152" b="68"/>
      </icon>
    </Product>
  </Line>
  <Line lineId="4_08_____S5" lineName="S5" lineNameShort="S5">
    <Product name="S5" line="S5" lineId="4_08_____S5" catOut="S" catCode="16"
operator="S-Bahn Berlin">
      <icon res="prod_comm_t">
        <foregroundColor r="255" g="255" b="255"/>
        <backgroundColor r="67" g="152" b="68"/>
      </icon>
    </Product>
  </Line>
  …
</LineList>
```

2.31. Line Match (linematch)

Returns a list (`LineList`) of matched lines .

2.31.1. 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.
operators	Optional	-	-	List of operators seperated by comma.

2.31.2. Example

Request: Get the list of lines by matching S9

```
<baseurl>/linematch?accessId=abc&input=S9
```

Result: (abbreviated)

```
<LineList serverVersion="2.8.1-SNAPSHOT" dialectVersion="1.29" requestId=
"debug">
  <Line lineId="de:rmv:00001324" lineName="S9">
    <Product name="S9" line="S9" lineId="de:rmv:00001324" catOut="S" cls="8"
operator="DB Regio AG S-Bahn Rhein-Main">
      <icon res="prod_gen"/>
    </Product>
  </Line>
  <Line lineId="8004A9_S9" lineName="S9">
    <Product name="S9" line="S9" lineId="8004A9_S9" catOut="S" cls="8"
operator="DB Regio AG Südost">
      <icon res="prod_gen"/>
    </Product>
  </Line>
  <Line lineId="08_____S9" lineName="S9">
    <Product name="S9" line="S9" lineId="08_____S9" catOut="S" cls="8"
operator="S-Bahn Berlin">
      <icon res="prod_gen"/>
    </Product>
  </Line>
</LineList>
```

2.32. Line Info (lineinfo)

The `lineinfo` service will deliver information about a certain line identified by its line ID on a specific date. It returns a list (`LineList`) along with the product information and representative line journeys.

2.32.1. Request Parameters

Name	Use	Range	Default	Description
accessId	Mandatory	-	-	Access ID for identifying the requesting client.
lineId	Mandatory	-	-	Specifies the internal line id of the line shall be retrieved.
date	Mandatory	-	-	Day of operation. Represented in the format YYYY-MM-DD.

2.32.2. Example

Request: Get the info for line with id `3(BART)` at the specific date `2019-06-01`.

```
<baseurl>/lineinfo?accessId=abc&lineId=3(BART)&date=2019-06-01
```

Result: (abbreviated)

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<LineList serverVersion="2.5.0-SNAPSHOT" dialectVersion="1.28" requestId=
"uc2agtnm4iwywg4c">
  <Line lineId="3(BART)" lineName="Warm Springs/South Fremont - Richmond"
lineNameShort="Orange">
    <Product name="Warm Springs/South Fremont - Richmond" line="Orange"
lineId="3(BART)" catOut="Metro" catCode="128" operator="Bay Area Rapid
Transit">
      <icon res="prod_sub">
        <foregroundColor r="0" g="0" b="0"/>
        <backgroundColor r="237" g="166" b="26"/>
      </icon></Product>
    <Journey direction="Richmond">
      <Stop name="Warm Springs/South Fremont BART Station, Fremont" id=
"A=1@O=Warm Springs/South Fremont BART Station, Fremont@X=-
121939311@Y=37502169@U=1@L=100013323@" extId="100013323" routeIdx="0" lon="
-121.939311" lat="37.502169"></Stop>
      <Stop name="Fremont BART Station, Fremont" id="A=1@O=Fremont BART
Station, Fremont@X=-121976598@Y=37557462@U=1@L=100013296@" extId="100013296"
routeIdx="1" lon="-121.976607" lat="37.557462"></Stop>
      <Stop name="Union City BART Station, Union City" id="A=1@O=Union City
BART Station, Union City@X=-122017382@Y=37590632@U=1@L=100013322@" extId=
"100013322" routeIdx="2" lon="-122.017283" lat="37.590767"></Stop>
      <Stop name="South Hayward BART Station, Hayward" id="A=1@O=South Hayward
BART Station, Hayward@X=-122057177@Y=37634374@U=1@L=100013320@" extId=
"100013320" routeIdx="3" lon="-122.057204" lat="37.634365"></Stop>
      <Stop name="Hayward BART Station, Hayward" id="A=1@O=Hayward BART
```

```
Station, Hayward@X=-122087013@Y=37669719@U=1@L=100013299@" extId="100013299"
routeIdx="4" lon="-122.087013" lat="37.669737"></Stop>
      <Stop name="Bay Fair BART Station, San Leandro" id="A=1@O=Bay Fair BART
Station, San Leandro@X=-122126511@Y=37696921@U=1@L=100013285@" extId=
"100013285" routeIdx="5" lon="-122.126502" lat="37.696939"></Stop>
      <Stop name="San Leandro BART Station, San Leandro" id="A=1@O=San Leandro
BART Station, San Leandro@X=-122160841@Y=37721947@U=1@L=100013317@" extId=
"100013317" routeIdx="6" lon="-122.16085" lat="37.721947"></Stop>
      <Stop name="Coliseum BART Station, Oakland" id="A=1@O=Coliseum BART
Station, Oakland@X=-122196861@Y=37753661@U=1@L=100013289@" extId="100013289"
routeIdx="7" lon="-122.19687" lat="37.753661"></Stop>
      <Stop name="Fruitvale BART Station, Oakland" id="A=1@O=Fruitvale BART
Station, Oakland@X=-122224170@Y=37774839@U=1@L=100013297@" extId="100013297"
routeIdx="8" lon="-122.224143" lat="37.774893"></Stop>
      <Stop name="Lake Merritt BART Station, Oakland" id="A=1@O=Lake Merritt
BART Station, Oakland@X=-122265179@Y=37797025@U=1@L=100013301@" extId=
"100013301" routeIdx="9" lon="-122.26508" lat="37.797213"></Stop>
      <Stop name="12th St. Oakland City Center BART Station, Oakland" id=
"A=1@O=12th St. Oakland City Center BART Station, Oakland@X=-
122271453@Y=37803766@U=1@L=100013278@" extId="100013278" routeIdx="10" lon="
-122.271516" lat="37.803811"></Stop>
      <Stop name="19th St. Oakland BART Station, Oakland" id="A=1@O=19th St.
Oakland BART Station, Oakland@X=-122268595@Y=37808351@U=1@L=100013280@" extId
="100013280" routeIdx="11" lon="-122.268694" lat="37.808405"></Stop>
      <Stop name="MacArthur BART Station, Oakland" id="A=1@O=MacArthur BART
Station, Oakland@X=-122267040@Y=37829062@U=1@L=100013302@" extId="100013302"
routeIdx="12" lon="-122.267093" lat="37.829071"></Stop>
      <Stop name="Ashby BART Station, Berkeley" id="A=1@O=Ashby BART Station,
Berkeley@X=-122270060@Y=37852803@U=1@L=100013283@" extId="100013283" routeIdx
="13" lon="-122.269889" lat="37.852731"></Stop>
      <Stop name="Downtown Berkeley BART Station, Berkeley" id="A=1@O=Downtown
Berkeley BART Station, Berkeley@X=-122268127@Y=37870107@U=1@L=100013292@"
extId="100013292" routeIdx="14" lon="-122.268064" lat="37.870107"></Stop>
      <Stop name="North Berkeley BART Station, Berkeley" id="A=1@O=North
Berkeley BART Station, Berkeley@X=-122283436@Y=37873963@U=1@L=100013306@"
extId="100013306" routeIdx="15" lon="-122.283418" lat="37.874026"></Stop>
      <Stop name="El Cerrito Plaza BART Station, El Cerrito" id="A=1@O=El
Cerrito Plaza BART Station, El Cerrito@X=-
122298897@Y=37902630@U=1@L=100013313@" extId="100013313" routeIdx="16" lon="
-122.298924" lat="37.902612"></Stop>
      <Stop name="El Cerrito del Norte BART Station, El Cerrito" id="A=1@O=El
Cerrito del Norte BART Station, El Cerrito@X=-
122316786@Y=37925085@U=1@L=100013293@" extId="100013293" routeIdx="17" lon="
-122.316813" lat="37.925076">
      </Stop><Stop name="Richmond BART Station, Richmond" id="A=1@O=Richmond
BART Station, Richmond@X=-122353093@Y=37936852@U=1@L=100013315@" extId=
"100013315" routeIdx="18" lon="-122.353093" lat="37.936852"></Stop>
    </Journey>
```

```
    <Journey direction="Richmond">
    ...
    </Journey>
  </Line>
</LineList>
```

2.33. Line Schedules (linesched)

The `lineschedules` service will deliver information about a certain line identified by its line ID on a specific date. It returns a list (`LineList`) along with the product information and all line journeys and times running at the given date.

2.33.1. Request Parameters

Name	Use	Range	Default	Description
accessId	Mandatory	-	-	Access ID for identifying the requesting client.
lineId	Mandatory	-	-	Specifies the internal line id of the line shall be retrieved.
date	Mandatory	-	-	Day of operation. Represented in the format YYYY-MM-DD.
orderBy	Optional	DEPARTURE_ASC, DEPARTURE_DESC, ARRIVAL_ASC, ARRIVAL_DESC	-	Sort the resulting list of journeys per line. DEPARTURE_ASC: sort by departure on first stop ascending DEPARTURE_DESC: sort by departure on first stop descending ARRIVAL_ASC: sort by arrival on last stop ascending ARRIVAL_DESC: sort by arrival on last stop descending
includeHim	Optional	0 or 1	0	Enables/disables the return of HIM messages.

2.33.2. Example

Request: Get the info for line with id `3(BART)` at the specific date `2019-06-01`.

```
<baseurl>/linesched?accessId=abc&lineId=3(BART)&date=2019-06-01
```

Result: (abbreviated)

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<LineList serverVersion="2.5.0-SNAPSHOT" dialectVersion="1.28" requestId=
"uc2agtnm4iwywg4c">
  <Line lineId="3(BART)" lineName="Warm Springs/South Fremont - Richmond"
lineNameShort="Orange">
    <Product name="Warm Springs/South Fremont - Richmond" line="Orange"
lineId="3(BART)" catOut="Metro" catCode="128" operator="Bay Area Rapid
Transit">
      <icon res="prod_sub">
        <foregroundColor r="0" g="0" b="0"/>
        <backgroundColor r="237" g="166" b="26"/>
      </icon></Product>
    <Journey direction="Richmond">
      <Stops>
        <Stop name="Warm Springs/South Fremont BART Station, Fremont" id=
"A=1@O=Warm Springs/South Fremont BART Station, Fremont@X=-
121939311@Y=37502169@U=1@L=100013323@" extId="100013323" routeIdx="0" lon="
-121.939311" lat="37.502169" depTime="09:27:00"></Stop>
        <Stop name="Fremont BART Station, Fremont" id="A=1@O=Fremont BART
Station, Fremont@X=-121976598@Y=37557462@U=1@L=100013296@" extId="100013296"
routeIdx="1" lon="-121.976607" lat="37.557462" depTime="09:33:00" arrTime=
"09:33:00"></Stop>
        <Stop name="Union City BART Station, Union City" id="A=1@O=Union City
BART Station, Union City@X=-122017382@Y=37590632@U=1@L=100013322@" extId=
"100013322" routeIdx="2" lon="-122.017283" lat="37.590767" depTime="09:38:00"
arrTime="09:38:00"></Stop>
        <Stop name="South Hayward BART Station, Hayward" id="A=1@O=South
Hayward BART Station, Hayward@X=-122057177@Y=37634374@U=1@L=100013320@" extId
="100013320" routeIdx="3" lon="-122.057204" lat="37.634365" depTime="09:43:00"
arrTime="09:43:00"></Stop>
        <Stop name="Hayward BART Station, Hayward" id="A=1@O=Hayward BART
Station, Hayward@X=-122087013@Y=37669719@U=1@L=100013299@" extId="100013299"
routeIdx="4" lon="-122.087013" lat="37.669737" depTime="09:47:00" arrTime=
"09:47:00"></Stop>
        <Stop name="Bay Fair BART Station, San Leandro" id="A=1@O=Bay Fair
BART Station, San Leandro@X=-122126511@Y=37696921@U=1@L=100013285@" extId=
"100013285" routeIdx="5" lon="-122.126502" lat="37.696939" depTime="09:51:00"
arrTime="09:51:00"></Stop>
        <Stop name="San Leandro BART Station, San Leandro" id="A=1@O=San
Leandro BART Station, San Leandro@X=-122160841@Y=37721947@U=1@L=100013317@"
extId="100013317" routeIdx="6" lon="-122.16085" lat="37.721947" depTime=
"09:54:00" arrTime="09:54:00"></Stop>
        <Stop name="Coliseum BART Station, Oakland" id="A=1@O=Coliseum BART
Station, Oakland@X=-122196861@Y=37753661@U=1@L=100013289@" extId="100013289"
routeIdx="7" lon="-122.19687" lat="37.753661" depTime="09:58:00" arrTime=
"09:58:00"></Stop>
        <Stop name="Fruitvale BART Station, Oakland" id="A=1@O=Fruitvale BART
Station, Oakland@X=-122224170@Y=37774839@U=1@L=100013297@" extId="100013297"
```

routeIdx="8" lon="-122.224143" lat="37.774893" depTime="10:02:00" arrTime="10:02:00"></Stop>
<Stop name="Lake Merritt BART Station, Oakland" id="A=1@O=Lake Merritt BART Station, Oakland@X=-122265179@Y=37797025@U=1@L=100013301@" extId="100013301" routeIdx="9" lon="-122.26508" lat="37.797213" depTime="10:06:00" arrTime="10:06:00"></Stop>
<Stop name="12th St. Oakland City Center BART Station, Oakland" id="A=1@O=12th St. Oakland City Center BART Station, Oakland@X=-122271453@Y=37803766@U=1@L=100013278@" extId="100013278" routeIdx="10" lon="-122.271516" lat="37.803811" depTime="10:09:00" arrTime="10:09:00"></Stop>
<Stop name="19th St. Oakland BART Station, Oakland" id="A=1@O=19th St. Oakland BART Station, Oakland@X=-122268595@Y=37808351@U=1@L=100013280@" extId="100013280" routeIdx="11" lon="-122.268694" lat="37.808405" depTime="10:11:00" arrTime="10:11:00"></Stop>
<Stop name="MacArthur BART Station, Oakland" id="A=1@O=MacArthur BART Station, Oakland@X=-122267040@Y=37829062@U=1@L=100013302@" extId="100013302" routeIdx="12" lon="-122.267093" lat="37.829071" depTime="10:14:00" arrTime="10:14:00"></Stop>
<Stop name="Ashby BART Station, Berkeley" id="A=1@O=Ashby BART Station, Berkeley@X=-122270060@Y=37852803@U=1@L=100013283@" extId="100013283" routeIdx="13" lon="-122.269889" lat="37.852731" depTime="10:18:00" arrTime="10:18:00"></Stop>
<Stop name="Downtown Berkeley BART Station, Berkeley" id="A=1@O=Downtown Berkeley BART Station, Berkeley@X=-122268127@Y=37870107@U=1@L=100013292@" extId="100013292" routeIdx="14" lon="-122.268064" lat="37.870107" depTime="10:20:00" arrTime="10:20:00"></Stop>
<Stop name="North Berkeley BART Station, Berkeley" id="A=1@O=North Berkeley BART Station, Berkeley@X=-122283436@Y=37873963@U=1@L=100013306@" extId="100013306" routeIdx="15" lon="-122.283418" lat="37.874026" depTime="10:22:00" arrTime="10:22:00"></Stop>
<Stop name="El Cerrito Plaza BART Station, El Cerrito" id="A=1@O=El Cerrito Plaza BART Station, El Cerrito@X=-122298897@Y=37902630@U=1@L=100013313@" extId="100013313" routeIdx="16" lon="-122.298924" lat="37.902612" depTime="10:25:00" arrTime="10:25:00"></Stop>
<Stop name="El Cerrito del Norte BART Station, El Cerrito" id="A=1@O=El Cerrito del Norte BART Station, El Cerrito@X=-122316786@Y=37925085@U=1@L=100013293@" extId="100013293" routeIdx="17" lon="-122.316813" lat="37.925076" depTime="10:28:00" arrTime="10:28:00"></Stop>
<Stop name="Richmond BART Station, Richmond" id="A=1@O=Richmond BART Station, Richmond@X=-122353093@Y=37936852@U=1@L=100013315@" extId="100013315" routeIdx="18" lon="-122.353093" lat="37.936852" arrTime="10:32:00">
</Stop></Stops>
</Journey>
<Journey direction="Richmond"></Journey>
<Journey direction="Richmond"></Journey>
<Journey direction="Richmond"></Journey>
<Journey direction="Richmond"></Journey>
<Journey direction="Richmond"></Journey>

```
      ...
    </Line>
  </LineList>
```

2.34. HIM Search (himsearch)

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

2.34.1. Request Parameters

Name	Use	Range	Default	Description
accessId	Mandatory	-	-	Access ID for identifying the requesting client.
dateB	Optional	See Section 1.2.2	-	Sets the event period start date.
dateE	Optional	See Section 1.2.2	-	Sets the event period end date.
timeB	Optional	See Section 1.2.2	-	Sets the event period start time.
timeE	Optional	See Section 1.2.2	-	Sets the event period end time.
weekdays	Optional	-	-	Bitmask for validity of HIM messages based on weekdays. Each character represents a weekday starting on monday. Example: Only find HIM Messages valid from monday to friday: 1111100
himIds	Optional	-	-	List of HIM message IDs seperated by comma.
hierarchicalView	Optional	0 or 1	0	Return parent messages with childs.
operators	Optional	-	-	List of operators seperated by comma.
categories	Optional	-	-	List of train categories seperated by comma.
channels	Optional	-	-	List of channels seperated by comma.
companies	Optional	-	-	List of companies seperated by comma.

lines	Optional	-	-	Only HIM messages for the given line are part of the result. To filter multiple lines, separate the codes by comma.
lineids	Optional	-	-	Only HIM messages for the given line (identified by its line ID) are part of the result. To filter multiple lines, separate the line IDs by comma.
stations	Optional	-	-	List of (external) station ids to be filtered for seperated by comma.
fromstation	Optional	-	-	Filter messages by line segment starting at this station given as (external) station id.
tostation	Optional	-	-	Filter messages by line segment travelling in direction of this station given as (external) station id.
bothways	Optional	0 or 1	-	If enabled, messages in both directions - from 'himfromstation' to 'himtostation' as well as from 'himtostation' to 'himfromstation' are returned
trainnames	Optional	-	-	List of train name to be filtered for seperated by comma.
metas	Optional	-	-	List of predefined filters seperated by comma.
himcategory	Optional	-	-	HIM category, e.g. Works and/or Disturbance. Value depends on your HAFAS server data.
himtags	Optional	-	-	HIM Tags. Value depends on your HAFAS server data.
himtext	Optional	-	-	Filter for Him messages containing the given free text message seperated by comma.
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	MATCH, NOMATCH, TFMATCH	-	HIM search mode. NOMATCH iterate over all HIM messages available. MATCH iterate over all trips to find HIM messages. TFMATCH uses filters defined **metas** parameter
affectedJourneyMode	Optional	ALL or OFF	-	Define how to return affected journeys OFF: do not return affected journeys. ALL: return affected journeys
affectedJourneyStopMode	Optional	ALL, IMP, OFF	-	Define how to return stops of affected journeys IMP: return important stops of affected journeys. OFF: do not return stops of affected journeys. ALL: return all affected stops of affected journeys
orderBy	Optional	-	-	Define the Order the returned messages by fields and directions. Multiple, comma separated entries are supported HID_ASC|HID_DESC: Sort based on HIM-ID. LMOD_ASC|LMOD_DESC: Sort based on timestamp of last update. EVT_BEG_ASC|EVT_BEG_DESC: Sort based on begin of the events period. EVT_END_ASC|EVT_BEG_DESC: Sort based on end of the events period. PRIO_ASC|PRIO_DESC: Sort based on priority of the event
minprio	Optional	-	-	Filter for HIM messages having at least this priority.

maxprio	Optional	-	-	Filter for HIM messages having this priority as maximum.

2.34.2. Example

Request: Get the HIM messages for february and march 2017

```
<baseurl>/himsearch?accessId=123&dateB=2017-02-01&dateE=2017-03-31
```

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 Verbindung 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>
```

2.35. RSS Feed (feed)

The `feed` service will deliver a list of HIM messages if matched by the given criteria as well as affected products if any. The output format is news feed in RSS 2.0 format.

2.35.1. Request Parameters

Request parameters are equivalent to HIM Search (see Section 2.34). The values of format and jsonpCallback are ignored for this service, since the output is fixed.

2.35.2. Example

Request: Get the RSS Feed for june 2017

```
<baseurl>/feed?accessId=123&dateB=2017-06-01&dateE=2017-06-30
```

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 eingerichtet. 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>
```

2.36. Real Time Archive Gateway (rtarchive)

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.

2.36.1. Request Parameters

Name	Use	Range	Default	Description
accessId	Mandatory	-	-	Access ID for identifying the requesting client.
id	Mandatory	-	-	Specifies the internal journey id of the journey shall be retrieved.
date	Optional	See Section 1.2.2	-	Sets the start date for which the departures shall be retrieved. Represented in the format YYYY-MM-DD.

2.36.2. 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>
```

2.37. Data Information (datainfo)

This service provides detailed information about all operators, administrations, products and product categories loaded by the underlying HAFAS server.

Most of the values are usable in various services as filter options.

2.37.1. Request Parameters

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

2.37.2. Example

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

Result: (abbreviated)

```
<DataInfo begin="2019-06-17" end="2019-09-15" serverVersion="2.5.0"
dialectVersion="1.28" requestId="nx2extyk6qx4h2cc">
  <Operator name="Arriva Kent & Surrey Ltd (GM)" nameL="Arriva  ">
    <administration>ASCGM_</administration>
  </Operator>
  <Operator name="Arriva Yorkshire (WA)" nameL="Arriva  ">
    <administration>AYKWA_</administration>
  </Operator>
  <Operator name="Arriva Merseyside (GRE)" nameL="Arriva  "></Operator>
  <Operator name="Arriva Merseyside (STH)" nameL="Arriva  "></Operator>
  <Operator name="Arriva Yorkshire (DE)" nameL="Arriva  "></Operator>
  ...
  <Product name="class05" catOut="Bus     " catIn="PBL" cls="32" catOutL="Bus
"/>
  ...
  <ProductCategory name="class05" cls="32">
    <Product name="class05" catOut="Bus     " catIn="PBL" cls="32" catOutL=
"Bus"/>
  </ProductCategory>
  ...
</DataInfo>
```

2.38. Time Table Information (tti)

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`.

In case of an ST typed pool, the validity period is returned with the begin and end date as well.

2.38.1. Request Parameters

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

2.38.2. Example

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

Result: (abbreviated)

```
<TimetableInfoList serverVersion="1.7.21" dialectVersion="1.23" requestId=
"1544174154581" begin="2017-12-10" end="2019-12-14">
  <TimetableInfo ident="s43bl" date="2018-11-27" time="13:19:59" type="ST"
begin="2017-12-10" end="2018-12-08"/>
  <TimetableInfo ident="s6qzl" date="2018-11-28" time="10:48:47" type="ST"
begin="2018-12-09" end="2019-12-14"/>
  <TimetableInfo ident="nbyyp" date="2018-08-06" time="15:07:27" type="ADR"/>
  <TimetableInfo ident="ifqyf" date="2018-05-03" time="11:16:53" type="POI"/>
</TimetableInfoList>
```

2.39. Walking Links (walkinglinks)

This service provides easy access to all walking links loaded by the underlying HAFAS server. It returns a list (`WalkingLink`) along with its attributes, polyline and HIM messages if any.

Request Parameters

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

fattributes	Optional	-	-	Filter walking links by one or more attribute codes. Multiple attribute codes are separated by comma. If the attribute should not be part of the walking link, negate it by putting ! in front of it.

2.39.2. Example

Request: Get all walking links having the attribute AU (elevator).

```
<baseurl>/walkinglinks?accessId=abc&fattributes=AU
```

Result:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<WalkingLinks serverVersion="2.10.0" dialectVersion="1.29" requestId=
"kcw82p7mk8gg9kw8"
  xmlns="http://hacon.de/hafas/proxy/hafas-proxy">
  <WalkingLink extId="110021168">
    <from id="A=1@O=Frankfurt (Main)
Hauptbahnhof@X=8663318@Y=50108030@U=80@L=320001076@" extId="320001076" name=
"Frankfurt (Main) Hauptbahnhof" lon="8.663318" lat="50.10803"/>
    <to id="A=1@O=Frankfurt (Main)
Hauptbahnhof@X=8663318@Y=50108039@U=80@L=320001075@" extId="320001075" name=
"Frankfurt (Main) Hauptbahnhof" lon="8.663318" lat="50.108039"/>
    <Note key="AU" type="A" priority="920" txtN="Aufzug">Aufzug</Note>
    <Note key="TB" type="A" priority="357" txtN="Taktile Bedienelemente am
Aufzug">Taktile Bedienelemente am Aufzug</Note>
    <Note key="aU" type="A" priority="352" txtN="Aufzug aufwärts">Aufzug
aufwärts</Note>
    <Note key="fD" type="A" priority="355" txtN="Tiefe des Aufzuges 203 cm"
>Tiefe des Aufzuges 203 cm</Note>
    <Note key="fE" type="A" priority="356" txtN="Maximale Belastung des
Aufzuges 1000 kg">Maximale Belastung des Aufzuges 1000 kg</Note>
    <Note key="fF" type="A" priority="950" txtN="Spiegel auf Gegenseite"
>Spiegel auf Gegenseite</Note>
    <Note key="gW" type="A" priority="290" txtN="mit Aufzug vom Bahnsteig U4
Enkheim/U5 Preungesheim (C-Ebene) zur Passage ( B-Ebene)">mit Aufzug vom
Bahnsteig U4 Enkheim/U5 Preungesheim (C-Ebene) zur Passage ( B-Ebene)</Note>
    <Note key="xq" type="A" priority="354" txtN="Aufzugbreite 90 cm"
>Aufzugbreite 90 cm</Note>
    <Note key="xs" type="A" priority="353" txtN="Lichte Breite des Aufzugs
(Tür) 90 cm">Lichte Breite des Aufzugs (Tür) 90 cm</Note>
    <PolylineGroup>
      <polylineDesc delta="true" dim="2" crdEncYX="evypHw`{s@A?" crdEncS="NN
"/>
```

```
    </PolylineGroup>
  </WalkingLink>
  <WalkingLink extId="110021168">
    <from id="A=1@O=Frankfurt (Main)
Hauptbahnhof@X=8663318@Y=50108039@U=80@L=320001075@" extId="320001075" name=
"Frankfurt (Main) Hauptbahnhof" lon="8.663318" lat="50.108039"/>
    <to id="A=1@O=Frankfurt (Main)
Hauptbahnhof@X=8663318@Y=50108030@U=80@L=320001076@" extId="320001076" name=
"Frankfurt (Main) Hauptbahnhof" lon="8.663318" lat="50.10803"/>
    <Note key="AU" type="A" priority="920" txtN="Aufzug">Aufzug</Note>
    <Note key="Au" type="A" priority="351" txtN="Aufzug abwärts">Aufzug
abwärts</Note>
    <Note key="TB" type="A" priority="357" txtN="Taktile Bedienelemente am
Aufzug">Taktile Bedienelemente am Aufzug</Note>
    <Note key="fD" type="A" priority="355" txtN="Tiefe des Aufzuges 203 cm"
>Tiefe des Aufzuges 203 cm</Note>
    <Note key="fE" type="A" priority="356" txtN="Maximale Belastung des
Aufzuges 1000 kg">Maximale Belastung des Aufzuges 1000 kg</Note>
    <Note key="fF" type="A" priority="950" txtN="Spiegel auf Gegenseite"
>Spiegel auf Gegenseite</Note>
    <Note key="gV" type="A" priority="290" txtN="mit Aufzug von Passage (B-
Ebene) zum  Bahnsteig U4 Enkheim/U5 Preungesheim (C-Ebene)">mit Aufzug von
Passage (B-Ebene) zum  Bahnsteig U4 Enkheim/U5 Preungesheim (C-Ebene)</Note>
    <Note key="xq" type="A" priority="354" txtN="Aufzugbreite 90 cm"
>Aufzugbreite 90 cm</Note>
    <Note key="xs" type="A" priority="353" txtN="Lichte Breite des Aufzugs
(Tür) 90 cm">Lichte Breite des Aufzugs (Tür) 90 cm</Note>
    <PolylineGroup>
      <polylineDesc delta="true" dim="2" crdEncYX="gvypHw`{s@@?" crdEncS="NN
"/>
    </PolylineGroup>
  </WalkingLink>
  <WalkingLink extId="110021169">
    <from id="A=1@O=Frankfurt (Main)
Hauptbahnhof@X=8663489@Y=50108138@U=80@L=320001078@" extId="320001078" name=
"Frankfurt (Main) Hauptbahnhof" lon="8.663489" lat="50.108138"/>
    <to id="A=1@O=Frankfurt (Main)
Hauptbahnhof@X=8663489@Y=50108156@U=80@L=320001077@" extId="320001077" name=
"Frankfurt (Main) Hauptbahnhof" lon="8.663489" lat="50.108156"/>
    <Note key="AU" type="A" priority="920" txtN="Aufzug">Aufzug</Note>
    <Note key="TB" type="A" priority="357" txtN="Taktile Bedienelemente am
Aufzug">Taktile Bedienelemente am Aufzug</Note>
    <Note key="aU" type="A" priority="352" txtN="Aufzug aufwärts">Aufzug
aufwärts</Note>
    <Note key="fD" type="A" priority="355" txtN="Tiefe des Aufzuges 203 cm"
>Tiefe des Aufzuges 203 cm</Note>
    <Note key="fE" type="A" priority="356" txtN="Maximale Belastung des
Aufzuges 1000 kg">Maximale Belastung des Aufzuges 1000 kg</Note>
```

```
    <Note key="fF" type="A" priority="950" txtN="Spiegel auf Gegenseite"
>Spiegel auf Gegenseite</Note>
    <Note key="gY" type="A" priority="290" txtN="mit Aufzug vom Bahnsteig U4
Bockenheimer Warte (C-Ebene) zur Passage (B-Ebene)">mit Aufzug vom Bahnsteig
U4 Bockenheimer Warte (C-Ebene) zur Passage (B-Ebene)</Note>
    <Note key="xq" type="A" priority="354" txtN="Aufzugbreite 90 cm"
>Aufzugbreite 90 cm</Note>
    <Note key="xs" type="A" priority="353" txtN="Lichte Breite des Aufzugs
(Tür) 90 cm">Lichte Breite des Aufzugs (Tür) 90 cm</Note>
    <PolylineGroup>
      <polylineDesc delta="true" dim="2" crdEncYX="{vypHya{s@C?" crdEncS="NN
"/>
    </PolylineGroup>
  </WalkingLink>
  <WalkingLink extId="110021169">
    <from id="A=1@O=Frankfurt (Main)
Hauptbahnhof@X=8663489@Y=50108156@U=80@L=320001077@" extId="320001077" name=
"Frankfurt (Main) Hauptbahnhof" lon="8.663489" lat="50.108156"/>
    <to id="A=1@O=Frankfurt (Main)
Hauptbahnhof@X=8663489@Y=50108138@U=80@L=320001078@" extId="320001078" name=
"Frankfurt (Main) Hauptbahnhof" lon="8.663489" lat="50.108138"/>
    <Note key="AU" type="A" priority="920" txtN="Aufzug">Aufzug</Note>
    <Note key="Au" type="A" priority="351" txtN="Aufzug abwärts">Aufzug
abwärts</Note>
    <Note key="TB" type="A" priority="357" txtN="Taktile Bedienelemente am
Aufzug">Taktile Bedienelemente am Aufzug</Note>
    <Note key="fD" type="A" priority="355" txtN="Tiefe des Aufzuges 203 cm"
>Tiefe des Aufzuges 203 cm</Note>
    <Note key="fE" type="A" priority="356" txtN="Maximale Belastung des
Aufzuges 1000 kg">Maximale Belastung des Aufzuges 1000 kg</Note>
    <Note key="fF" type="A" priority="950" txtN="Spiegel auf Gegenseite"
>Spiegel auf Gegenseite</Note>
    <Note key="gX" type="A" priority="290" txtN="mit Aufzug von Passage (B-
Ebene) zum Bahnsteig U4  Bockenheimer Warte (C-Ebene)">mit Aufzug von Passage
(B-Ebene) zum Bahnsteig U4  Bockenheimer Warte (C-Ebene)</Note>
    <Note key="xq" type="A" priority="354" txtN="Aufzugbreite 90 cm"
>Aufzugbreite 90 cm</Note>
    <Note key="xs" type="A" priority="353" txtN="Lichte Breite des Aufzugs
(Tür) 90 cm">Lichte Breite des Aufzugs (Tür) 90 cm</Note>
    <PolylineGroup>
      <polylineDesc delta="true" dim="2" crdEncYX="_wypHya{s@B?" crdEncS="NN
"/>
    </PolylineGroup>
  </WalkingLink>
</WalkingLinks>
```

2.40. Traffic Messages (Datex II) (trafficmessages/datex2)

This `datex2` service provides the up-to-date Datex II data.

2.40.1. Request Parameters

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

2.40.2. Response

As a result, the service returns the data in the Datex II format.

2.40.3. Example

Request: `<baseurl>/trafficmessages/datex2?lang=en&source=CdT`

Response:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<d2LogicalModel xmlns:ns2="http://datex2.eu/schema/2/2_0">
    <ns2:exchange>
        <ns2:supplierIdentification>
            ...
        </ns2:supplierIdentification>
    </ns2:exchange>
    <ns2:payloadPublication xsi:type="ns2:SituationPublication" lang="en"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
        ...
    </ns2:payloadPublication>
</d2LogicalModel>
```

2.41. Download Service (downloads)

This `downloads` service enables download of files protected by the accessId. If no path is added, the list of downloadable files is presented, otherwise the download starts immediately.

2.41.1. Sample call

Listing: `<baseurl>/downloads?accessId=8022dea6-55b9-4667-956c-e4de9297c86c`

File: `<baseurl>/downloads/HAFAS_ReST_Interface_-_Dokumentation.pdf?accessId=8022dea6-55b9-4667-956c-e4de9297c86c`

2.42. XSD Service

The XSD service will return a certain XML Schema Definition of a certain version. Calling the XSD service with no path or with the query parameter `list`, a list of all available XSD files will be returned in HTML format.

2.42.1. Example

Request: List all available XSD files

```
<baseurl>/xsd?list
```

Response:

- `rest-1.29.xsd`

Request: Return XSD

```
<baseurl>/xsd/rest-1.29.xsd
```

2.43. GraphQL Service

The *QL* endpoint supports Queries written in GraphQL against most of the previously listed Services. Noteworthy exceptions are:

- Feed Service
- Documentation Services (XSD, WADL, Swagger, ...)

The GraphQL API is a full implementation of all provided by the Rest-Interface. Communication is done via JSON.

3. Responses

All services return their responses either in XML or JSON format (see Section 1.2.8). All possible response elements are defined in `rest-1.29.xsd` which could be retrieved via `<baseURL>/xsd?rest-1.29.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 jour-ney). 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 503 is send.

4.1. ReST Request Errors

Code	HTTP status code	Description
Client errors		
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 too 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_SMS_NUM	400	receiver sms phone number invalid or missing
SVC_NO_RESULT	400	no result found
Server errors		
SVC_MAIL	500	failed to send mail
SVC_SMS	500	failed to send sms
SVC_FAILED_SEARCH	500	unsuccessful search
SVC_NO_MATCH	422	no match found
INT_ERR	500	internal error

Code	HTTP status code	Description
INT_GATEWAY	503	communication error with backend systems
INT_TIMEOUT	503	timeout during service processing
Service specific errors		
SOT_AT_DEST	400	Trip already arrived
SOT_BEFORE_START	400	Trip not started
SOT_CANCELLED	400	Trip cancelled

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.

Date	Version	Author	Remarks
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

Date	Version	Author	Remarks
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.

Date	Version	Author	Remarks
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.

Date	Version	Author	Remarks
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.
11.07.2018	1.23.24	kha	• Add RSS feed service. • Removed RSS response type from HIM search.
22.07.2018	1.23.25	kha	Support for multiple entries in trip service for originMeta, destMeta and totalMeta
13.08.2018	1.23.25	mfr	Adjust numF and numB description in trip search.
06.09.2018	1.23.26	mfr	• Add chapter Trip Alternatives service • Add request path chapter for Interval trip search service
28.09.2018	1.23.27	mfr	Extend Trip Alternatives service parameters

Date	Version	Author	Remarks
04.12.2018	1.23.28	mfr	• Add new reverse geocoding service • Add parameters to restrict Journey Details stop list • Add new filter for bike carriage type on trip search services
07.12.2018	1.23.29	mfr	• Add new fields on time table info service. • Add new error codes related to Search on trip and Trip Alternatives service. • Make the versioning chapter more explicit.
07.03.2019	2.2.2	kha	• Add graphql interface • Add api-doc inteface
13.03.2019	2.3.0	kha	Add new Journey Track Match service
10.05.2019	2.4.0	kha	• Add swagger-ui and api-doc endpoints • Add new filters for HIM search and Feed service • Expanded description of format and accessId parameters • Cleanup of Service parameters.
06.06.2019	2.4.2	kha	• Add description for custom parameters in gis routing parameters (e.g originWalk, originBike, ...) for Trip and Trip Alternatives service • Added description of probabilities for Journey Track Match service
19.06.2019	2.5.0	kha	Added new error codes related to backend communication and timeouts
20.06.2019	2.5.0	mfr	• Add new Line search service • Add new Line info service
28.06.2019	2.5.0	kha	Marked extId parameters as deprecated.
09.08.2019	2.5.3	kha	Added orderBy to Line schedules service
25.09.2019	2.6.0	kha	Clarified XML to JSON mapping
26.09.2019	2.6.1	kha	Grouped errors, refined location service type filters
08.11.2019	2.6.2	kha	Added Reconstruction Match service

Date	Version	Author	Remarks
08.11.2019	2.6.5	mfr	• Map TOO_MANY error to HTTP 400 status code • Service location.nearbystops gets configurable maxLoc and maxLocDefault • Add via parameter to search on trip • Add parameter inlcudeIV to trip search service
20.12.2019	2.6.6	mfr	Adds product information in case of leg type gis
28.01.2020	2.6.7	mfr	• Adds provisioning for `meta` on location.name and location.nearbystops service • Adds provisioning for `groupFilter` on trip searches
06.02.2020	2.7.0	mfr	• Adds provisioning for `tariff` on trip searches • Add fattributes filter on trip searches • Adds filterMode on location.name service
11.02.2020	2.7.1	mfr	Adds provisioning for `rtMode`
03.03.2020	2.7.2	mfr	• Adds mapping for stop cancellation for arrival and destination of itinerary stops • Adds depDir to any itinerary stop if present • Adds filter for lines on train search and journey match services • Adds includeHim option to Line Schedules service, add HIM Messages structure to Line output • Adds platform filter to board requests • Adds timezone information for stations
20.03.2020	2.7.3	mfr	• Adds baim option to Journey Detail service • Eco value response: default values 0.0 removed. • Add withICTAlternatives option to trip searches
01.04.2020	2.7.4	mfr	• Add optional attribute "matchId" to ProductType • Add attribute "entry" to attlist.StopLocation
08.04.2020	2.7.5	mfr	• SVC_NO_RESULT returns HTTP 400 instead of 500 • Add name for gis route if available • Add occupancy structure to Trip, Leg, OriginDest

Date	Version	Author	Remarks
14.04.2020	2.8.0	mfr	• Remove trainNumber and trainCategory from Arrival and Departure • Make Product 0..* on Arrival and Departure • Add displayNumber to ProductType
20.05.2020	2.8.1	mfr	• Add Line Match service • Add Reachability search services • Add Partial Trip Search service • Add Occupancy to Stops, Departure, Arrival, Journey • Add Direction list to LineType • Add includeEarlier option to trip services • Add hierarchicalView option to Him Search service • Add rtDuration to TripType
28.05.2020	2.8.2	mfr	Platform filter is positive only. Negation by ! is not interpreted.
01.07.2020	2.8.4	mfr	• Add N:M Search service • SVC_NO_MATCH set HTTP return code to 422
16.06.2020	2.9.0	mfr	• Adds customText attribute to Message • Adds provisioning to himCategory • Adds reliability to Trip (HAFAS Time Machine support)
23.06.2020	2.9.1	mfr	• Adds Origin and Destination to Trip • SVC_NO_MATCH set HTTP return code to 422
06.07.2020	2.9.2	mfr	Internal
06.07.2020	2.9.3	mfr	• New service Location search in bounding box • Add chapter HAFAS Time Machine support
20.07.2020	2.10.0	mfr	• Add new service Walking Links • Add new service Walking Links by Location • Add new service Downloads • Add new service Location Data • Add rtMode option to Reconstruction service • Add provisioned himtags option to HIM Search service

Date	Version	Author	Remarks
15.09.2020	2.11.0	mfr	• Add allowDummySections option to Reconstruction and Reconstruction Match services • Add new service Partial Trip Search V2 service
22.09.2020	2.12.0	mfr	• Add type option to station board services • Split station board service documentation • Add totalUphill and totalDownhill to GIS response if returned by GIS router