HACON

HAFAS.api

Access to HAFAS Journey Planner systems

Version 2.45.2, 2024-09-04

Table of Contents

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 HTTP based interface providing services to interact with HAFAS Journey Planner. Following list provides an overview about the available services.

- Location name (Section 2.3)
- Location Search by Coordinate (Section 2.4)
- Location Search in a bounding box (Section 2.5)
- Location Search (Section 2.6)
- Location Data (Section 2.7)
- Location Details (Section 2.8)
- Address Lookup (Section 2.9)
- Trip Search (Section 2.10)
- Interval Trip Search (Section 2.11)
- Reconstruction (Section 2.12)
- GIS Route by Context (Section 2.13)
- Departure Board (Section 2.14)
- Arrival Board (Section 2.15)
- Multi Departure Board (Section 2.16)
- Multi Arrival Board (Section 2.17)
- Nearby Departure Board (Section 2.18)
- Nearby Arrival Board (Section 2.19)
- Journey Detail (Section 2.20)
- Journey Match (Section 2.21)
- Journey Position (Section 2.22)
- Train Search (Section 2.23)
- Line Search (Section 2.24)
- Line Match (Section 2.25)
- Line Info (Section 2.26)
- Line Schedules (Section 2.27)

- HIM Search (Section 2.28)
- Data Information (Section 2.29)
- Time Table Information (Section 2.30)
- Tariff Stop (Section 2.31)
- Convert zones (Section 2.32)
- Zone From Coordinate (Section 2.33)
- SP Price (Section 2.34)
- SP Price Reconstruction (Section 2.35)
- SP Price for Zones (Section 2.36)
- SP Check (Section 2.37)
- SP Zone Check (Section 2.38)
- SP Daily Zonal Prices (Section 2.39)
- SP Refund (Section 2.40)
- ST Stops (Section 2.41)
- ST Routes (Section 2.42)
- ST Routes Add-on (Section 2.43)
- XSD
- OpenAPI Specification 3.x
- Status
- System Information

The system implements read-only HTTP GET/POST requests which are called by given service URLs and multiple 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.

Any response structure is described in the provided XSD. You may use it as a source to generate binding structures for your specific consumer are. To complete the documentation, the whole API is described as OpenAPI Specification 3.x and can be retrieved from any running instance.

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. Attributes

The HAFAS data model allows the definition of attributes for locations, journeys and footpaths. They are defined in the HAFAS raw data file `attribut`. Each attribute has its own code. Multiple services provide the ability to filter by HAFAS attribute codes.

Sample definition:

```
K1 0 200 10 Only 1st class#
```

- `K1` - attribute code
- `0` - to be used for itinerary
- `200` - priority(0 - 999)
- `10` - sorting key (0 - 99)
- `Only 1st class#` - UTF-8 text of attribute ending with `#`

For more information, take a look at documentation *HAFAS Raw data file format* chapter *Attributes and Meta-Attributes [attribut]*.

1.2.4. Infotexts

The HAFAS data model allows the definition of infotexts for locations, journeys and footpaths. They are defined in the HAFAS raw data file `infotext`. Each infotext has its own code. Multiple services provide the ability to filter by HAFAS infotext codes and may be combined with values.

Sample definition:

```
00000000001 North-Express
00000000002 South-Express
```

To the infotext number 1 belongs this text: North-Express. To the infotext number 2 belongs this text: South-Express.

For more information, take a look at documentation *HAFAS Raw data file format* chapter *Information texts [infotext]*.

1.2.5. Combining Attributes and Infotexts

HAFAS allows a combination of attributes and infotexts. This enables a flexible controlling of attribute texts depending on the underlying context. The mechanism avoids a redundant definition of attribute texts. A user may directly insert infotext codes into attribute texts. These codes augment the underlying attribute texts. Prior to output HAFAS replaces all infotext codes by the assigned texts.

Example:

```
[attribut:]
RO 0 001 50 Restaurant open$IOZ#

[infotext:]
00000003 6PM until 8PM
00000004 6PM until 11PM
```

In the definition of an itinerary, this can be used like this:

```
*Z 00815 4711__
*A RO 008010366 008010097
*I OZ 008010366 008010097 000001 000000003
*I OZ 008010366 008010097 000002 000000004
```

Infotext OZ is assigned to the attribute RO. The file infotext contains the definitions of the underlaying texts. In the example, two entries about opening hours are assigned to infotext codes 000000003 and 000000004, respectively. Finally, a particular service in the time table file fplan contains links to the attribute definition of RO as well as the infotext definition of OZ. All definitions refer to a route section that stretches from stop 008010366 until stop 008010097. Each of the two infotexts refers to different days-of-service. Whereas the first infotext entry refers to bitfield 000001, the second infotext entry refers to bitfield 000002. The further one may refer to weekdays whereas the later one may refer to weekends.

For more information, take a look at documentation *HAFAS Raw data file format* chapter *Attributes and Meta-Attributes [attribut]*.

1.2.6. 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.

1.2.7. 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.

1.2.8. 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.

1.2.9. 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.

1.2.10. Global IDs / Alternative ID spaces

HAFAS supports multiple alternative ID spaces / Global IDs for one dataset.

Let’s assume the following configuration: One with an ID consisting of seven digits, prefixed with "U" and only allowed to be used with that prefix combination in a request. And a second one with a two letter ID, a custom delimiter "-" and allowed to be used without the defined prefix "B".

```
defaultGlobalIdPatternDelimiter: "$$$"
globalIdPatterns:
  - prefix: "U"
    pattern: "^[0-9]{7}$"
    allowWithoutPrefix: false
  - prefix: "B"
    pattern: "^[a-zA-Z]{2}$"
    delimiter: "-"
```

A location may look like this in the response:

```
<StopLocation id="A=1@O=Villach
Westbahnhof@X=13840143@Y=46608098@U=81@L=8100148@B=1@p=1705593192@" extId=
"8100148">
  <altId>U$$$8103656</altId>
  <altId>B-Vf</altId>
</StopLocation>
```

Having those four IDs, following options would work in the request, e.g. in a Departure Board Service:

- id → /departureBoard?id=A=1@O=Villach Westbahnhof@X=13840143@Y=46608098@U=81@L=8100148@B=1@p=1705593192@
- extId → /departureBoard?id=8100148
- U → /departureBoard?id=U$$$8103656
- B with prefix → /departureBoard?id=B-Vf
- B without prefix → /departureBoard?id=Vf

1.2.11. 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.

1.2.12. 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</a></root> becomes { "a": { "value" : "foo" } }

- Nested elements become nested properties

  <root><a><b>foo</b><c>foo</c></a></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><b>foo1</b><b>foo2</b></a><root>

  becomes

  { "a": { "b" : [{"value": foo1" }, {"value": "foo2" }] } }

- Attribute names become object properties

  <root><a atb="foo1">foo2</a><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="2.45.2" dialectVersion="{dialect_version}" xmlns=
"http://hacon.de/hafas/proxy/hafas-proxy" requestId="dhhjwn44">
    <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": "2.45.2",
    "dialectVersion": "{dialect_version}",
    "requestId": "dhhjwn44",
    "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" }
        }
    }]
}
```

1.2.13. 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.

1.2.14. 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
ar	Arabic
ca	Catalan, Valencian
da	Danish
de	German
el	Greek
en	English
es	Spanish
fi	Finnish
fr	French
hi	Hindi
hr	Croatian
hu	Hungarian
it	Italian
nl	Dutch
no	Norwegian
pl	Polish
ru	Russian
sk	Slovak
sl	Slovenian
sv	Swedish
tl	Tagalog
tr	Turkish
ur	Urdu

zh	Chinese

1.2.15. 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 occur in any log as well as in the response like this:

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

```
<?xml version="1.0" encoding="UTF-8"?>
<TripList serverVersion="2.45.2" dialectVersion="2.45-Rejseplanen" xmlns=
"http://hacon.de/hafas/proxy/hafas-proxy" requestId="123456789">
...
</TripList>
```

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

The requestId can be up to 2048 characters long.

1.2.16. OpenAPI Specification

If `enableOpenAPI: true` is set in the configuration file, the HAFAS.api 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.api.

1.2.17. HTTP GET and HTTP POST

Any service available as `HTTP GET` can be addressed by `HTTP POST` as well. You have to set the `Content-Type` header to `application/x-www-form-urlencoded` and put the parameters into the body of the HTTP request.

Example:

```
GET /restproxy/trip?accessId=123&originId=8000261&destId=8011102&date=2020-
09-17 HTTP/1.1
Host: demo.hafas.de
```

can be expressed like this

```
POST /restproxy/trip HTTP/1.1
Host: demo.hafas.de
Content-Type: application/x-www-form-urlencoded

accessId=123&originId=8000261&destId=8011102&date=2020-09-17
```

1.2.18. Technical messages

HAFAS.api will return some technical messages not related to its domain of journey planning but technical integration, monitoring and debugging.

Each service response may have a block like this:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<LocationPreselectionResponse ...>
    <TechnicalMessages>
        <TechnicalMessage key="requestTime">2021-10-06
21:05:05</TechnicalMessage>
    </TechnicalMessages>
    ...
</LocationPreselectionResponse>
```

Following messages may return:

- `requestTime` → Time the request was made.
- `unmappedQueryParams` → Comma separated list of query params not mapped. Use for debugging purposes.

2. Services

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

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 chosen.

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).

Service added in version 1.0.

2.3.1. Request Parameters

- Use M: Mandatory
- Use O: Optional

Name	Use	Range	Default	Est.	Description

accessId	M	-	-	1.0	Access ID for identifying the requesting client.
input	M	-	-	1.0	Search for that token.
maxNo	O	1-1000	10	1.0	Maximum number of returned stops.
type	O	A, ALL, AP, P, S, SA, SP	ALL	1.20	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	O	SLCT_A or SLCT_N	-	1.20	Selection mode for locations. SLCT_N: Not selectable SLCT_A: Selectable
withEquivalentLocations	O	0 or 1	0	2.33	Return equivalent locations.
restrictSelection	O	D, I, S, V	S	2.31	Restrict allowed stations. S: Locations allowed as start. D: Locations allowed as destination. V: Locations allowed as via. I: Ignore all restrictions (Start/Destination/Via) to select locations.

products	O		-	1.15	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. Values are retrievable by Data Information service. 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.
withProducts	O	0 or 1	1	2.28	Return locations with products.
productRepresentatives	O	0 or 1	-	2.28	If activated, only one representative of a location product per category is returned.
coordLat	O	See Section 1.2.1	-	1.20	Latitude of centre coordinate.
coordLong	O	See Section 1.2.1	-	1.20	Longitude of centre coordinate.
r	O		1000	1.20	Search radius in meter around the given coordinate if any.
refineId	O	-	-	1.0	In case of an refinable location, this value takes the ID of the refinable one of a previous result.
meta	O	-	-	1.0	Filter by a predefined meta filter. If the rules of the predefined filter should not be negated, put ! in front of it. Multiple values are separated by comma if definded for POI filtering. If filter values are restricted by provisioning, disallowed values will lead to an error.

stations	O	-	-	1.0	Filter for stations. Matches if the given value is prefix of any station ID. Multiple values are separated by comma.
sattributes	O	See Section 1.2.3	-	1.0	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	O	See Section 1.2.4	-	2.24	Filter locations by one or more station infotext codes and values. Parameter structure is code|value. Multiple attribute codes are separated by comma.
filterMode	O	DIST_PERI, EXCL_PERI, SLCT_PERI	DIST_PERI	2.7	Filter modes for nearby searches. DIST_PERI: Accentuate matches. Matches in the radius are first. SLCT_PERI: Returns matches inside the radius only. EXCL_PERI: Matches in the radius are excluded. Returns matches outside the radius only.
poolId	O	-	-	2.26	Filter locations by pool id (also known as Pool UIC code). Multiple values are separated by comma. To negate put ! in front of the value.
withMastNames	O	0 or 1	1	2.35	Add masts to found stations.

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. Restrict selection for start, stop or via defined in data

In HAFAS data stations can be marked for which usage they are allowed. Markers are

- Start: S
- Destination: D
- Via: V

With the use of the parameter `restrictSelection` you are able to make use of this data driven restriction in your query. Beside the values above, ignoring any restriction with the value `I` is also possible.

2.3.4. 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).

2.3.5. 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

2.3.6. Example

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

Result:

```
<LocationList serverVersion="2.45.2" dialectVersion="2.45-Rejseplanen"
xmlns="http://hacon.de/hafas/proxy/hafas-proxy" requestId="1483016905755">
    <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).

Service added in version 1.0.

2.4.1. Request Parameters

- Use M: Mandatory
- Use O: Optional

Name	Use	Range	Default	Est.	Description
accessId	M	-	-	1.0	Access ID for identifying the requesting client.
originCoordLat	M	See Section 1.2.1	-	1.0	Latitude of centre coordinate.
originCoordLong	M	See Section 1.2.1	-	1.0	Longitude of centre coordinate.

r	O		1000	1.0	Search radius in meter around the given coordinate.
maxNo	O	or 1-1000	10	1.0	Maximum number of returned stops.
type	O	S, P, SP, SE, PE, SPE	S	1.0	Type filter for location types. Values are S: Search for station/stops only P: Search for POIs only SE: Search for stations/stops and entrypoints SP: Search for stations/stops and POIs. PE: Search for POIs and entrypoints. SPE: Search for stations/stops, POIs and entrypoints.
locationSelectionMode	O	SLCT_A or SLCT_N	-	1.20	Selection mode for locations. SLCT_N: Not selectable SLCT_A: Selectable
products	O		-	1.15	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. Values are retrievable by Data Information service. 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.

meta	O	-	-	1.0	Filter by a predefined meta filter. If the rules of the predefined filter should not be negated, put ! in front of it. Multiple values are separated by comma if definded for POI filtering. If filter values are restricted by provisioning, disallowed values will lead to an error.
sattributes	O	See Section 1.2.3	-	1.0	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	O	See Section 1.2.4	-	1.0	Filter locations by one or more station infotext codes and values. Multiple attribute codes are separated by comma the value by pipe |.
poolId	O	-	-	2.26	Filter locations by pool id (also known as Pool UIC code). Multiple values are separated by comma. To negate put ! in front of the value.
date	O	-	-	2.39	Incorporate date. Represented in the format YYYY-MM-DD.
time	O	-	-	2.39	Incorporate time. Represented in the format hh:mm[:ss] in 24h nomenclature. Seconds will be ignored for requests.
zoom	O	0-21	-	2.39	Retrieve stops most relevant to this map zoom level.
poiCategories	O	-	-	2.42	Filter locations by POI types. 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.

2.4.2. Example

Request: Search for stations around the coordinate

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

Result:

```
<LocationList serverVersion="2.45.2" dialectVersion="2.45-Rejseplanen"
xmlns="http://hacon.de/hafas/proxy/hafas-proxy" requestId="1483016905755">
    <StopLocation id="A=1@O=Oslo S@X=10755332@Y=59910200@u=0@U=70
        @L=7600100@" name="Oslo S" lon="10.755332" lat="59.9102"/>
    <StopLocation id="A=1@O=Nydalen st@X=10758982@Y=59915405@u=0
        @U=70@L=7621273@" name="Nydalen st" lon="10.758982"
        lat="59.915405"/>
</LocationList>
```

2.4.3. Type filter

Locations retrievable by this service are either from type S - stop or P - point of interest. If a returned location is an entry point to a stop, it is of type S but has an attribute `entry=true` in the response. API provides the option to filter for stops or POIs only, a combination of stops and POIs and for entry points in combination at least with stops or stops and POIs.

- `type=S` → filter for stops only
- `type=P` → filter for points of interest only
- `type=SP` → filter for stops or points of interest
- `type=SE` → filter for stops and entry points
- `type=SPE` → filter for stops, entry points and points of interest

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).

Service added in version 2.9.3.

2.5.1. Request Parameters

- Use M: Mandatory
- Use O: Optional

Name	Use	Range	Default	Est.	Description
accessId	M	-	-	1.0	Access ID for identifying the requesting client.
llLat	M	See Section 1.2.1	-	2.9.3	Lower left latitude of bounding box.
llLon	M	See Section 1.2.1	-	2.9.3	Lower left longitude of bounding box.
urLat	M	See Section 1.2.1	-	2.9.3	Upper right latitude of bounding box.
urLon	M	See Section 1.2.1	-	2.9.3	Upper right longitude of bounding box.
type	O	S, P, SP, SE, PE, SPE	S	2.9.3	Type filter for location types. Values are S: Search for station/stops only P: Search for POIs only SE: Search for stations/stops and entrypoints SP: Search for stations/stops and POIs. PE: Search for POIs and entrypoints. SPE: Search for stations/stops, POIs and entrypoints.
locationSelectionMode	O	SLCT_A or SLCT_N	-	2.9.3	Selection mode for locations. SLCT_N: Not selectable SLCT_A: Selectable

products	O		-	2.9.3	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. Values are retrievable by Data Information service. 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.
meta	O	-	-	2.9.3	Filter by a predefined meta filter. If the rules of the predefined filter should not be negated, put ! in front of it. Multiple values are separated by comma if definded for POI filtering.
sattributes	O	See Section 1.2.3	-	2.9.3	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	O	See Section 1.2.4	-	2.9.3	Filter locations by one or more station infotext codes and values. Multiple attribute codes are separated by comma the value by pipe |.
poolId	O	-	-	2.26	Filter locations by pool id (also known as Pool UIC code). Multiple values are separated by comma. To negate put ! in front of the value.
poiCategories	O	-	-	2.42	Filter locations by POI types. 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.

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 serverVersion="2.45.2" dialectVersion="2.45-Rejseplanen"
xmlns="http://hacon.de/hafas/proxy/hafas-proxy" requestId="1483016905755">
  <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>
```

2.5.3. Type filter

Locations retrievable by this service are either from type S - stop or P - point of interest. If a returned location is an entry point to a stop, it is of type S but has an attribute `entry=true` in the response. API provides the option to filter for stops or POIs only, a combination of stops and POIs and for entry points in combination at least with stops or stops and POIs.

- `type=S` → filter for stops only
- `type=P` → filter for points of interest only
- `type=SP` → filter for stops or points of interest
- `type=SE` → filter for stops and entry points

- `type=SPE` → filter for stops, entry points and points of interest

2.6. Location Search (location.search)

The `location.search` service returns filtered locations.

Service added in version 1.0.

2.6.1. Request Parameters

- Use M: Mandatory
- Use O: Optional

Name	Use	Range	Default	Est.	Description
accessId	M	-	-	1.0	Access ID for identifying the requesting client.
maxNo	O		10	1.0	Maximum number of returned stops.
scrollCtx	O	-	-	2.24	Scroll context for paginated requests.
scrollSize	O		-	2.24	Enables paginated result with the given size per chunk. Page number will be ceiling maxNo / scrollSize
products	O		-	1.0	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. Values are retrievable by Data Information service. 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.

poiCategories	O	-	-	2.42	Filter locations by POI types. 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.
sattributes	O	See Section 1.2.3	-	1.0	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	O	See Section 1.2.4	-	2.7	Filter locations by one or more station infotext codes and values. Parameter structure is code|value. Multiple attribute codes are separated by comma.

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="2.45.2" dialectVersion="2.45-Rejseplanen"
xmlns="http://hacon.de/hafas/proxy/hafas-proxy" requestId="1483016905755">
    <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. 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).

Service added in version 2.10.

2.7.1. Request Parameters

- Use M: Mandatory
- Use O: Optional

Name	Use	Range	Default	Est.	Description
accessId	M	-	-	1.0	Access ID for identifying the requesting client.
startId	M		-	2.10	Start ID of location id range.
endId	M		-	2.10	End ID of location id range.
attributes	O	0 or 1	0	2.10	Return locations with attributes. See Section 1.2.3.
infotexts	O	0 or 1	0	2.10	Return locations with infotexts. See Section 1.2.4.
products	O	0 or 1	0	2.10	Return locations with products.

2.7.2. Example

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

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<LocationList serverVersion="2.45.2" dialectVersion="2.45-Rejseplanen"
xmlns="http://hacon.de/hafas/proxy/hafas-proxy" requestId="1483016905755">
    <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>
```

2.8. 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).

Service added in version 1.0.

2.8.1. Request Parameters

- Use M: Mandatory
- Use O: Optional

Name	Use	Range	Default	Est.	Description
accessId	M	-	-	1.0	Access ID for identifying the requesting client.
id	O	-	-	1.0	Specifies the list of station/stop IDs for which the details shall be retrieved. Values are separated by |.

extId	O	-	-	2.27	Specifies the list of external station/stop IDs for which the details shall be retrieved. Required if id is not present. Values are separated by |. Such ID can be retrieved from the location.name or location.nearbystops services.
date	O	-	-	1.0	Incorporate date. Represented in the format YYYY-MM-DD.
time	O	-	-	1.0	Incorporate time. Represented in the format hh:mm[:ss] in 24h nomenclature. Seconds will be ignored for requests.
attributes	O	0 or 1	1	1.0	Enables/disables return of location attribtues. See Section 1.2.3
infotexts	O	0 or 1	1	1.0	Enables/disables return of location infotexts. See Section 1.2.4.
products	O	0 or 1	1	1.0	Enables/disables return of products served by this location.
messages	O	0 or 1	0	1.0	Enables/disables return of location messages.
tariffs	O	0 or 1	0	1.0	Enables/disables return of tariff information.
weather	O	0 or 1	0	1.0	Enables/disables return of weather information.
zoom	O	0-21	-	2.39	Return information related to the zoom level.

2.8.2. 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="2.45.2" dialectVersion="2.45-Rejseplanen"
xmlns="http://hacon.de/hafas/proxy/hafas-proxy" requestId="1483016905755">
  <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>
```

2.9. Address Lookup (addresslookup)

The `addresslookup` service returns a list of possible addresses around a given center coordinate. You can filter by type to check for station, address, POI or any combination.

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

Service added in version 1.23.28.

2.9.1. Request Parameters

- Use M: Mandatory
- Use O: Optional

Name	Use	Range	Default	Est.	Description
accessId	M	-	-	1.0	Access ID for identifying the requesting client.
originCoordLat	M	See Section 1.2.1	-	1.23.28	Latitude of coordinate.
originCoordLong	M	See Section 1.2.1	-	1.23.28	Longitude of coordinate.

type	O	A, ALL, AP, C, P, S, SA, SAP, SP	C	2.15	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 C: Search for coordinate based location SA: Search for station/stops and addresses SP: search for station/stops and POIs AP: search for addresses and POIs SAP: search for stop/station, address or POI

2.9.2. Example

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

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<LocationList serverVersion="2.45.2" dialectVersion="2.45-Rejseplanen"
xmlns="http://hacon.de/hafas/proxy/hafas-proxy" requestId="1483016905755">
    <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>
```

2.10. 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.

Service added in version 1.0.

2.10.1. Request Parameters

- Use M: Mandatory
- Use O: Optional

Name	Use	Range	Default	Est.	Description
accessId	M	-	-	1.0	Access ID for identifying the requesting client.
originId	O	See Section 2.3 or Section 2.4	-	1.0	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	O	-	-	1.20	**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	O	See Section 1.2.1 and Section 2.3 or Section 2.4	-	1.0	Latitude of station/stop coordinate of the trip's origin. The coordinate can be retrieved from the location.name or location.nearbystops services.
originCoordLong	O	See Section 1.2.1 and Section 2.3 or Section 2.4	-	1.0	Longitude of station/stop coordinate of the trip's origin. The coordinate can be retrieved from the location.name or location.nearbystops services.
originCoordName	O	-	-	2.13	Name of the trip's origin if coordinate cannot be resolved to an address or poi.

originCoordType	O	A, ALL, C, P, S	-	2.30	Type of the trip’s origin if coordinate cannot be resolved to an address or poi.
destId	O	See Section 2.3 or Section 2.4	-	1.0	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	O	-	-	1.20	**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	O	See Section 1.2.1 and Section 2.3 or Section 2.4	-	1.0	Latitude of station/stop coordinate of the trip’s destination. The coordinate can be retrieved from the location.name or location.nearbystops services.
destCoordLong	O	See Section 1.2.1 and Section 2.3 or Section 2.4	-	1.0	Longitude of station/stop coordinate of the trip’s destination. The coordinate can be retrieved from the location.name or location.nearbystops services.
destCoordName	O	-	-	2.13	Name of the trip’s destination if coordinate cannot be resolved to an address or poi.
destCoordType	O	A, ALL, C, P, S	-	2.30	Type of the trip’s destination if coordinate cannot be resolved to an address or poi.

via	O	-	-	1.0	Complex structure to provide multiple via points separated by semicolon.
					
					This structure is build like this:
					viaId|waittime|viastatus|products|direct|sleepingCar|couchetteCoach|attributes
					viaId: id, extId or altId of the via, mandatory. Routing via coordinates in individual 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. Values are retrievable by Data Information service.
					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. See Section 1.2.3.
					
					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	O	See Section 2.3 or Section 2.4	-	1.0	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	O		0	1.0	Defines the waiting time spent at via station in minutes. If via is used, viaId and viaWaitTime are having no effect.
avoid	O	-	-	1.0	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	O	See Section 2.3 or Section 2.4	-	1.22.2	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.

viaGis	O	-	-	2.14	Complex structure to provide multiple GIS via locations separated by semicolon. This structure is build like this: locationId|locationMode|transportMode|placeType|usageType|mode|durationOfStay locationId: id of the GIS via location locationMode: Defines if location is for departure (pick up / check in) or for arrival (drop off / check out). One of DEP (pick up / check in), ARR (drop off / check out), VIA (use as via) transportMode: Defines which mode of transport is used. One of WALK, BIKE, CAR placeType: Defines kind of mode change point. One of FIX (pick point and drop off is the same), FLEX (flexible mode change point. Drop off may take place at another place of the same type), FREE (free pick up point, drop off can be anywhere, my be restircted by area), RIDE (*-And-Ride place for vehicles), CHARGE (charging station), RETURN (drop off point of an already in use vehicle) usageType: Own vehicle, sharing vehicle, pooled vehicle or other means of transport. One of S (sharing), R (rental), P (pooling), T (taxi), O (own vehicle) mode: Defines if this via is used for first mile, last mile, total trip or a mix. One of F (first mile), B (last mile), FB (first and last mile), T (total), FBT (first mile, last mile and total) durationOfStay: Minimum duration of stay at via before departure or after arrival. Optional. Default to 0.

changeTimePercent	O		100	1.0	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	O		-	1.0	Minimum change time at stop in minutes.
maxChangeTime	O		-	1.20	Maximum change time at stop in minutes.
addChangeTime	O		-	1.0	This amount of minutes is added to the change time at each stop.
maxChange	O	0-11	-	1.0	Maximum number of changes.
date	O	See Section 1.2.2	-	1.0	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	O	See Section 1.2.2	-	1.0	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	O	0 or 1	0	1.0	If set, the date and time parameters specify the arrival time for the trip search instead of the departure time.

numF	O	1-6	5	1.10	Minimum number of trips after or before the search time, depending on search direction. Sum of numF and numB has to be less or equal 6. The maximum value depends on the actual configuration. Please see the section below about the search algorithm for more details. Please see the section below about the search algorithm for more details.
numB	O	0-5	0	1.10	Minimum number of trips before or after the search time, depending on search direction. Sum of numF and numB has to be less or equal 6. The maximum value depends on the actual configuration. Please see the section below about the search algorithm for more details. Please see the section below about the search algorithm for more details.
context	O	See Section 2.10.3	-	1.0	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	O	0 or 1	0	1.11	Enables/disables the calculation of the polyline for each leg of the trip except any GIS route.
polyEnc	O	DLT, GPA, N	N	1.11	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	O	0 or 1	0	1.20	Enables/disables the return of the passlist for each leg of the trip.

products	O		-	1.0	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. Values are retrievable by Data Information service. 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	O	All operator codes or names from HAFAS raw data, accessible via Data Information service.	-	1.12	Only trips provided by the given operators are part of the result. To filter multiple operators, separate the codes by comma. If the operator should not be part of the trip, negate it by putting ! in front of it. Example: Filter for operator A and B: `operators=A,B.`
categories	O	All category codes or names from HAFAS raw data, accessible via Data Information service.	-	2.25	Only trips provided by the given categories are part of the result. To filter multiple categories, separate the codes by comma. If the category should not be part of the trip, negate it by putting ! in front of it. Example: Filter for category A and B: `categories=A,B.`
categoryFlags	O	-	-	2.35	Only trips matching the given category flags are part of the result. To filter multiple category flags, separate the codes by comma. Example: Filter for category flag LOCAL: `categoryFlags=LOCAL.`

attributes	O	See Section 1.2.3	-	1.0	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	O	See Section 1.2.3	-	1.23.7	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	O	See Section 1.2.3	-	2.7	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	O	-	-	1.0	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	O	-	-	1.0	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	O	One or more codes	-	1.12	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	O	-	-	1.0	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	O	-	-	1.0	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 max. of 2.5km default speed: `originBike=1,0,2500,100,0,sharing,callabike`

originCar	O	-	-	1.0	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 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	O	-	-	1.0	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	O	-	-	1.0	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	O	-	-	1.23.25	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	O	-	-	1.0	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	O	-	-	1.0	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	O	-	-	1.0	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	O	-	-	1.0	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	O	-	-	1.0	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	O	-	-	1.23.25	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	O	-	-	1.23.20	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	O	-	-	1.23.20	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	O	-	-	1.23.20	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	O	-	-	1.23.20	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	O	-	-	1.23.25	Enables using one or more predefined individual transport meta profile for a trip. The profiles are defined in the HAFAS installation.

gisProducts	O	-	-	2.19	Filter on GIS product, e.g. specific sharing provider. Currently, only exclusion of certain providers is available by adding ! in front of the provider meta code. Available codes are customer sprecific.
includeIv	O	0 or 1	0	2.6.5	Enables/disables search for individual transport routes.
ivOnly	O	0 or 1	0	1.23.20	Enables/disables search for individual transport routes only.
includeDrt	O	0 or 1	1	2.42	Enables/disables search for DRT routes.
mobilityProfile	O	-	-	1.0	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	O	0 or 1	0	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.
bikeCarriageType	O	SINGLEBIKES, SMALLGROUPS, LARGEGROUPS	-	2.16	Filter for a specific bike carriage type. Allowed types are SINGLEBIKES, SMALLGROUPS and LARGEGROUPS. May be not available in any installation.
sleepingCar	O	0 or 1	0	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	O	0 or 1	0	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	O	0 or 1	0	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	O	0 or 1	0	1.23.8	Enables/disables BAIM search and response.
eco	O	0 or 1	0	1.23.9	**Only supported if legacy environmental calculator is used.** Deprecated. Enables/disables eco value calculation.
ecoCmp	O	0 or 1	0	1.23.9	**Only supported if legacy environmental calculator is used.** Deprecated. Enables/disables eco comparison.
ecoParams	O	-	-	1.23.9	Provide additional eco parameters. For exact values, check your eco documentation if any.

rtMode	O	FULL, INFOS, OFF, REALTIME, SERVER_DEFAULT	-	1.0	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 and all results are combined. 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	O	0 or 1	0	1.23.13	Enables/disables unsharp search mode. For details, see Section 2.10.2.1
trainFilter	O	-	-	1.23.9	Filters a trip search for a certain train. First hit will be taken

economic	O	0 or 1	0	1.23.14	Enables/disables economic search mode. For details, see Section 2.10.2.2
allowFootpathEquivalences	O	0 or 1	1	2.26	If the walk is disabled or a maximum walk distance of 0 is set and all other gis modalities are disabled HAFAS is allowed to use start/destination equivalences that are connected by a data foot path.
groupFilter	O	-	-	1.23.14	Use a predefined group filter to query for certain modes.
blockingList	O	-	-	1.23.18	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.
blockedEdges	O	-	-	2.22	List of edges within the public transport network that should be excluded from the result. Each edge is defined by a tuple of the following style: start location ID|end location ID|bidirectional|blockOnlyIfInOutAllowed A set of tuples can be separated by semicolon.
trainComposition	O	0 or 1	0	1.0	Enables/disables train composition data.
includeEarlier	O	0 or 1	0	1.23.18	Disables search optimization in relation of duration.

withICTAlternatives	O	0 or 1	0	1.23.19	Enables/disables the search for alternatives with individualized change times (ICT).
tariff	O	0 or 1	-	2.7	Enables/disables the output of tariff data. The default is configurable via provisioning.
trafficMessages	O	0 or 1	-	2.7	Enables/disables the output of traffic messages. The default is configurable via provisioning.
travellerProfileData	O	-	-	2.15	Traveller profile data. Structure depends on set up.
withFreq	O	0 or 1	1	2.18	Enables/disables the calculation of frequency information.
withJourneyBoundaryPoints	O	0 or 1	0	2.24	Enables/disables the return of journey boundary stops at public transport legs.

2.10.2. 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 returns 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. If numB is greater than 0 (numB=m), the server server returns not less than the least m possible connections in the past, if searchForArrival=0, or in the future, if searchForArrival=1.

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 smaller 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=5 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.

2.10.3. 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.

2.10.4. 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).

2.10.5. Occupancy data

Depending on the setup, the returned trips and legs can contain occupancy information, which describes capacity information for the journey. The element structure is defined as:

```
<xs:complexType name="OccupancyType">
        <xs:annotation>
            <xs:documentation>Occupany information</xs:documentation>
        </xs:annotation>
        <xs:attribute name="name" type="xs:string" use="optional">
            <xs:annotation>
                <xs:documentation>Name of seat class or
category.</xs:documentation>
            </xs:annotation>
        </xs:attribute>
        <xs:attribute name="v" type="xs:int" use="optional" default="0">
            <xs:annotation>
                <xs:documentation>Seat occupancy value of this class or
category between 0 and 100.</xs:documentation>
            </xs:annotation>
        </xs:attribute>
        <xs:attribute name="number" type="xs:string" use="optional">
            <xs:annotation>
                <xs:documentation>Public number of the car for which the
occupancy data is valid for, if the data is only valid for a single car.
Otherwise this attribute is simply left out.</xs:documentation>
            </xs:annotation>
        </xs:attribute>
        <xs:attribute name="raw" type="xs:int" use="optional">
            <xs:annotation>
                <xs:documentation>Seat occupancy raw
data</xs:documentation>
            </xs:annotation>
        </xs:attribute>
    </xs:complexType>
```

The values depend on the provided data or contingent plan. In case of the contingent data, "raw" and "v" attributes can contain the status number in range 10 - 13:
10: normally not filled and means no occupancy available
11: expected occupancy is low
12: expected occupancy is medium
13: expected occupancy is high

The example below provides occupancy information for two legs of a journey. In the first leg, identified as "M1" and heading towards "Zingster Str.," the expected occupancy for the FIRST class is low, and for the SECOND class is medium. The second leg, labeled as "M2" and directed towards "Oranienburger Tor.," indicates expected high occupancy level for the SECOND class.

```
<LegList>
    <Leg id="1" type="JNY" name="M1" direction="Zingster Str.">
        <Occupancy name="FIRST" raw="11"/>
        <Occupancy name="SECOND" raw="12"/>
    </Leg>
    <Leg id="2" type="JNY" name="M2" direction="Oranienburger Tor.">
        <Occupancy name="SECOND" raw="13"/>
    </Leg>
</LegList>
```

2.10.6. 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

2.10.7. 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

Product filter on via segment

- Nothing set: any product allowed for any segment
- products=32: product filter set for **any** segment, restricted to value of 32
- products=32 and via=1234|||65535: product filter set for first segment, restricted to 32; for second segment, no restriction as all bits are set
- products=32 and via=1234|||64: product filter set for first segment, restricted to 32; for second segment, restriction set to 64
- via=1234|||64: any product allowed for first segment; product filter set for second segment, restricted to 64

Attribute filter on via segment

- Nothing set: no filter for attributes applied
- attributes=A0: all parts of the connection have to have this attribute
- attributes=A0 and via=1234: first segment has to have this attribute. After the via stop, no restriction applied
- attributes=A0 and via=1234||||||A0: restriction applied to first and second segment
- via=1234||||||A0: restriction applied to second segment only

2.10.8. 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>
```

2.11. 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.

Service added in version 1.0.

2.11.1. Request Parameters

- Use M: Mandatory
- Use O: Optional

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. Please note, the upper bound is configurable.
max	Optional	-	-	At least, this amount of connections will be returned by the search within the given period.
combine	Optional	-	0	Pack results.

Following parameters have no effect:

Name	Use	Range	Default	Description
numF	Not used			
numB	Not used			

2.11.2. 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).

If requested with `combine=1`, returned trips are combined to reduce data as well as marking `Trip` elements with attributes `combinedCount` and `combinedMinDuration` if necessary.

2.11.3. Request path

The service listens at `<baseurl>/interval`

2.12. 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.

Service added in version 1.12.

2.12.1. Request Parameters

- Use M: Mandatory
- Use O: Optional

Name	Use	Range	Default	Est.	Description
accessId	M	-	-	1.0	Access ID for identifying the requesting client.
ctx	M	-	-	1.12	Specifies the reconstruction context.
poly	O	0 or 1	0	1.12	Enables/disables the calculation of the polyline for each leg of the trip except any GIS route.
polyEnc	O	DLT, GPA, N	N	1.12	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	O	-	-	1.12	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	O	0 or 1	-	-	Compare based on combined output name - false: Compare parameters (category, line, train number) individually
acceptGaps	O	0 or 1	-	2.6.2	Accept an incomplete description of the connection (with gaps) i.e. missing walks/transfers

allowDummySections	O	0 or 1	-	2.11.0	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	O	0 or 1	-	2.6.2	Should all non-reachable journeys be flagged (true), or only the first one encountered?
matchCatStrict	O	0 or 1	-	2.6.2	Should the category (Gattung) match exactly? Only applicable if useCombinedComparison is false
matchIdNonBlank	O	0 or 1	-	2.6.2	Should the train identifier (Zugbezeichner) without whitespace match?
matchIdStrict	O	0 or 1	-	2.6.2	Should the train identifier (Zugbezeichner) match exactly?
matchNumStrict	O	0 or 1	-	2.6.2	Should the train number (Zugnummer) match exactly? Only applicable if useCombinedComparison is false
matchRtType	O	0 or 1	-	2.6.2	Should the realtime type that journeys are based on (e.g. SOLL, IST, additional, deviation, ...) be considered?
enableRtFullSearch	O	0 or 1	-	2.15	By default, the reconstruction request makes one attempt for each journey within the scheduled data. However, the scheduled data may not necessarily reflect basic realtime properties of the journeys therein. In such a case, one may enable a two-step approach which we call "full search", i.e. search for matching journeys in the scheduled data in a first step. If this fails, then search for matching journeys in the realtime data.

enableReplacements	O	0 or 1	-	2.21.0	If set to true replaces cancelled journeys with their replacement journeys if possible.
arrL	O	0-720	-	2.6.2	Lower deviation in minutes within interval [0, 720] indicating "how much earlier than original arrival"
arrU	O	0-720	-	2.6.2	Upper deviation in minutes within interval [0, 720] indicating "how much later than original arrival"
depL	O	0-720	-	2.6.2	Lower deviation in minutes within interval [0, 720] indicating "how much earlier than original departure"
depU	O	0-720	-	2.6.2	Upper deviation in minutes within interval [0, 720] indicating "how much later than original departure"
passlist	O	0 or 1	0	1.12	Enables/disables the return of the passlist for each leg of the trip.
showPassingPoints	O	0 or 1	0	1.12	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	O	FULL, INFOS, OFF, REALTIME, SERVER_DEFAULT	-	2.10.0	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 and all results are combined. 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.
baim	O	0 or 1	0	2.39	Enables/disables BAIM.
eco	O	0 or 1	0	1.23.9	**Only supported if legacy environmental calculator is used.** Deprecated. Enables/disables eco value calculation.

ecoCmp	O	0 or 1	0	1.23.9	**Only supported if legacy environmental calculator is used.** Deprecated. Enables/disables eco comparison.
ecoParams	O	-	-	1.23.9	Provide additional eco parameters. Values vary.
tariff	O	0 or 1	-	2.7	Enables/disables the output of tariff data. The default is configurable via provisioning.
trafficMessages	O	0 or 1	-	2.7	Enables/disables the output of traffic messages. The default is configurable via provisioning.
travellerProfileData	O	-	-	2.15	Traveller profile data. Structure depends on set up.
withJourneyBoundaryPoints	O	0 or 1	0	2.24	Enables/disables the return of journey boundary stops at public transport legs.
freq	O		-	2.35	Sets the frequency interval length for the search of journey alternatives in minutes. Using a value of 0 results in the suppression of the search for alternatives.

2.12.2. 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
$201409182238$ R10$§G@F$A=1@O=Holmestrand@L=7601505@a=128@$A=2@O=Holmestrand,
Langgaten 2@X=10312173@Y=59492256@u=60@a=128@$201409182238$201409182244
```

2.12.3. Response

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

2.12.4. Partial reconstruction

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

2.12.5. 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 2.10.8.

2.13. 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.

Service added in version 1.0.

2.13.1. Request Parameters

- Use M: Mandatory
- Use O: Optional

Name	Use	Range	Default	Est.	Description
accessId	M	-	-	1.0	Access ID for identifying the requesting client.
ctx	M	-	-	1.0	Specifies the GIS route context.
poly	O	0 or 1	0	1.11	Enables/disables the calculation of the polyline.

polyEnc	O	DLT, GPA, N	N	1.11	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	O	0 or 1	0	1.23.9	Enables/disables eco value calculation.
baim	O	0 or 1	0	2.39	Enables/disables BAIM.

2.13.2. 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="2.45.2" dialectVersion="2.45-Rejseplanen" xmlns=
"http://hacon.de/hafas/proxy/hafas-proxy"
  requestId="1483016905755">
  <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$$"
    checksum="523B5244_4" tripId="C-0" duration="PT0S"
    ecoUrl="demo.hafas.de/bin/pub/bene/eco/query.exe/dn?
&amp;L=ns_hispeed_eco&amp;application=ECOLOGYIN-FO&amp;
requestConnection=1&amp;conReconstruction=1&amp;newEcologyValues=1&amp;ecoF
rom=1062HD Amsterdam, Cornelis Lelylaan 35&amp;ecoFromId=205996696
&amp;ecoFromX=4834021&amp;ecoFromY=52357940&amp;ecoTo=Amsterdam Le-
lylaan&amp;ecoToId=008400079&amp;ecoToX=4833913&amp;ecoToY=52357887&amp;VH=
G@F$A=2@O=1062HD Amsterdam, Cornelis Lelylaan
35@X=4834021@Y=52357940@u=9@a=128@$A=1@O=Amsterdam
Lelylaan@L=8400079@a=0@$201608251236$201608251236$$&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
@2000@120@-1@100@1@1000@0@@@@@false@0@$f@$f@$f@$f@$f@$§bt@0@2000@120@-
1@100@1@1000@0@@@@@false@0@$f@$f@$f@$f@$f@$§tt@0@5000@120@-
1@100@1@2500@0@@@@@false@0@$t@0@25000@120@-
1@100@1@3000@0@@@@@false@0@$f@$f@$f@$f@$§|" />
        <GisRoute dist="8" durS="PT0S">
          <seg dist="8" manTx="Nehmen Sie die Treppe" name="Treppe" ori="E"
            rType="CT" />
          <seg man="TO" manTx="Nehmen Sie die Treppe" />
        </GisRoute>
      </Leg>
    </LegList>
  </Trip>
</TripList>
```

2.14. Departure Board (departureBoard)

The departure 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 last minute found even if the requested maximum was overrun.

Service added in version 1.0.

2.14.1. Request Parameters

- Use M: Mandatory
- Use O: Optional

Name	Use	Range	Default	Est.	Description
accessId	M	-	-	1.0	Access ID for identifying the requesting client.
id	O	See Section 2.3 or Section 2.4	-	1.0	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	O	-	-	1.21	**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	O	See Section 2.3 or Section 2.4	-	1.0	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	O	See Section 1.2.2	-	1.0	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	O	See Section 1.2.2	-	1.0	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	O	0-1439	60	1.0	Set the interval size in minutes.

maxJourneys	O	-1	1.0	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	O	-	1.12	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. Values are retrievable by Data Information service. 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	O	All operator codes or names from HAFAS raw data, accessible via Data Information service.	-	1.0	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 journeys, negate it by putting ! in front of it. Example: Filter for operator A and B: `operators=A,B.`
categories	O	All category codes or names from HAFAS raw data, accessible via Data Information service.	-	2.25	Only journeys provided by the given categories are part of the result. To filter multiple categories, separate the codes by comma. If the category should not be part of the journeys, negate it by putting ! in front of it. Example: Filter for category A and B: `categories=A,B.`
lines	O	-	-	1.0	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.
attributes	O	See Section 1.2.3	-	1.0	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	O	-	-	2.7.2	Filter boards by platform. Multiple platforms are separated by comma.
passlist	O	0 or 1	0	1.0	Include a list of all passed waystops?
passlistMaxStops	O		-	1.0	Maximum number of stops including requested stop and last stop.

minDur	O		-	1.0	Minimum duration a journey has left to be returned.
baim	O	0 or 1	0	2.26	Enables/disables BAIM information.
rtMode	O	OFF or SERVER_DEFAULT	SERVER_DEFAULT	2.34	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. SERVER_DEFAULT – one of the above configured in the HAFAS server back end.
type	M	DEP, DEP_EQUIVS, DEP_MAST, DEP_STATION	DEP	2.12	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.

2.14.2. 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 serverVersion="2.45.2" dialectVersion="2.45-Rejseplanen"
xmlns="http://hacon.de/hafas/proxy/hafas-proxy" requestId="1483016905755">
    <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>
```

2.14.3. 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.

2.14.4. 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).

2.15. 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.

Service added in version 1.0.

2.15.1. Request Parameters

- Use M: Mandatory
- Use O: Optional

Name	Use	Range	Default	Est.	Description
accessId	M	-	-	1.0	Access ID for identifying the requesting client.
id	O	See Section 2.3 or Section 2.4	-	1.0	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	O	-	-	1.21	**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	O	See Section 2.3 or Section 2.4	-	1.0	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	O	See Section 1.2.2	-	1.0	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	O	See Section 1.2.2	-	1.0	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	O	0-1439	60	1.0	Set the interval size in minutes.
maxJourneys	O		-1	1.0	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	O		-	1.12	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. Values are retrievable by Data Information service. 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	O	All operator codes or names from HAFAS raw data, accessible via Data Information service.	-	1.0	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 journeys, negate it by putting ! in front of it. Example: Filter for operator A and B: `operators=A,B.`
categories	O	All category codes or names from HAFAS raw data, accessible via Data Information service.	-	2.25	Only journeys provided by the given categories are part of the result. To filter multiple categories, separate the codes by comma. If the category should not be part of the journeys, negate it by putting ! in front of it. Example: Filter for category A and B: `categories=A,B.`

lines	O	-	-	1.0	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.
attributes	O	See Section 1.2.3	-	1.0	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	O	-	-	2.7.2	Filter boards by platform. Multiple platforms are separated by comma.
passlist	O	0 or 1	0	1.0	Include a list of all passed waystops?
passlistMaxStops	O		-	1.0	Maximum number of stops including requested stop and last stop.
minDur	O		-	1.0	Minimum duration a journey has left to be returned.
baim	O	0 or 1	0	2.26	Enables/disables BAIM information.
rtMode	O	OFF or SERVER_DEFAULT	SERVER_DEFAULT	2.34	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. SERVER_DEFAULT – one of the above configured in the HAFAS server back end.

type	M	ARR, ARR_EQUIVS, ARR_MAST, ARR_STATION	ARR	2.12	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.

2.15.2. 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 serverVersion="2.45.2" dialectVersion="2.45-Rejseplanen"
xmlns="http://hacon.de/hafas/proxy/hafas-proxy" requestId="1483016905755">
    <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>
```

2.15.3. 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.

2.15.4. 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).

2.16. Multi Departure Board (multiDepartureBoard)

Departure boards for multiple stations can be retrieved by a call to the `multiDepartureBoard` 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 last minute found even if the requested maximum was overrun. All departures are mixed into one single list.

Service added in version 2.35.

2.16.1. Request Parameters

- Use M: Mandatory
- Use O: Optional

Name	Use	Range	Default	Est.	Description
accessId	M	-	-	1.0	Access ID for identifying the requesting client.
idList	M	-	-	2.35	Specifies a list of station/stop IDs for which the departures shall be retrieved. Split by `|`.
direction	O	See Section 2.3 or Section 2.4	-	1.0	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	O	See Section 1.2.2	-	1.0	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	O	See Section 1.2.2	-	1.0	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	O	0-1439	60	1.0	Set the interval size in minutes.
maxJourneys	O		-1	1.0	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	O		-	1.12	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. Values are retrievable by Data Information service. 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	O	All operator codes or names from HAFAS raw data, accessible via Data Information service.	-	1.0	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 journeys, negate it by putting ! in front of it. Example: Filter for operator A and B: `operators=A,B.`
categories	O	All category codes or names from HAFAS raw data, accessible via Data Information service.	-	2.25	Only journeys provided by the given categories are part of the result. To filter multiple categories, separate the codes by comma. If the category should not be part of the journeys, negate it by putting ! in front of it. Example: Filter for category A and B: `categories=A,B.`
lines	O	-	-	1.0	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.
attributes	O	See Section 1.2.3	-	1.0	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	O	-	-	2.7.2	Filter boards by platform. Multiple platforms are separated by comma.
passlist	O	0 or 1	0	1.0	Include a list of all passed waystops?
passlistMaxStops	O		-	1.0	Maximum number of stops including requested stop and last stop.

minDur	O		-	1.0	Minimum duration a journey has left to be returned.
baim	O	0 or 1	0	2.26	Enables/disables BAIM information.
rtMode	O	OFF or SERVER_DEFAULT	SERVER_DEFAULT	2.34	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. SERVER_DEFAULT – one of the above configured in the HAFAS server back end.
type	M	DEP, DEP_EQUIVS, DEP_MAST, DEP_STATION	DEP	2.12	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 all 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.

2.16.2. Example

Request: Departure board for three stations with IDs 12345, 98765 and 456234 on June 1st, 2020, at 6 pm

```
<baseurl>/multiDepartureBoard?idList=12345|98765|456234&date=2020-06-01&time=18:00
```

Result: (abbreviated)

```
<DepartureBoard serverVersion="2.45.2" dialectVersion="2.45-Rejseplanen"
xmlns="http://hacon.de/hafas/proxy/hafas-proxy" requestId="1483016905755">
    <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>
```

2.16.3. 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.

2.16.4. 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).

2.17. Multi Arrival Board (multiArrivalBoard)

Arrival boards for multiple stations can be retrieved by a call to the `multiArrivalBoard` 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. All arrivals are mixed into one single list.

Service added in version 2.35.

2.17.1. Request Parameters

- Use M: Mandatory
- Use O: Optional

Name	Use	Range	Default	Est.	Description
accessId	M	-	-	1.0	Access ID for identifying the requesting client.
idList	M	-	-	2.35	Specifies a list of station/stop IDs for which the arrivals shall be retrieved. Split by `|`.
direction	O	See Section 2.3 or Section 2.4	-	1.0	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	O	See Section 1.2.2	-	1.0	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	O	See Section 1.2.2	-	1.0	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	O	0-1439	60	1.0	Set the interval size in minutes.
maxJourneys	O		-1	1.0	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	O		-	1.12	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. Values are retrievable by Data Information service. 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	O	All operator codes or names from HAFAS raw data, accessible via Data Information service.	-	1.0	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 journeys, negate it by putting ! in front of it. Example: Filter for operator A and B: `operators=A,B.`
categories	O	All category codes or names from HAFAS raw data, accessible via Data Information service.	-	2.25	Only journeys provided by the given categories are part of the result. To filter multiple categories, separate the codes by comma. If the category should not be part of the journeys, negate it by putting ! in front of it. Example: Filter for category A and B: `categories=A,B.`
lines	O	-	-	1.0	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.
attributes	O	See Section 1.2.3	-	1.0	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	O	-	-	2.7.2	Filter boards by platform. Multiple platforms are separated by comma.
passlist	O	0 or 1	0	1.0	Include a list of all passed waystops?
passlistMaxStops	O		-	1.0	Maximum number of stops including requested stop and last stop.

minDur	O		-	1.0	Minimum duration a journey has left to be returned.
baim	O	0 or 1	0	2.26	Enables/disables BAIM information.
rtMode	O	OFF or SERVER_DEFAULT	SERVER_DEFAULT	2.34	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. SERVER_DEFAULT – one of the above configured in the HAFAS server back end.
type	M	ARR, ARR_EQUIVS, ARR_MAST, ARR_STATION	ARR	2.12	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.

2.17.2. Example

Request: Arrival board for three stations with IDs 12345, 98765 and 456234 on June 1st, 2020, at 6 pm

```
<baseurl>/multiArrivalBoard?idList=12345|98765|456234&date=2020-06-01&time=18:00
```

Result: (abbreviated)

```
<ArrivalBoard serverVersion="2.45.2" dialectVersion="2.45-Rejseplanen"
xmlns="http://hacon.de/hafas/proxy/hafas-proxy" requestId="1483016905755">
    <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>
```

2.17.3. 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.

2.17.4. 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).

2.18. Nearby Departure Board (nearbyDepartureBoard)

Departure boards for stations in a certain radius nearby a given coordinate can be retrieved by a call to the `nearbyDepartureBoard` 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. The default radius is 1000m and the default amount of stations taken into account is 30.

Note: The result list always contains all departures running the last minute found even if the requested maximum was overrun. All departures are mixed into one single list.

Service added in version 2.37.

2.18.1. Request Parameters

- Use M: Mandatory
- Use O: Optional

Name	Use	Range	Default	Est.	Description
accessId	M	-	-	1.0	Access ID for identifying the requesting client.
originCoordLat	M	See Section 1.2.1	-	2.37	Latitude of centre coordinate.
originCoordLong	M	See Section 1.2.1	-	2.37	Longitude of centre coordinate.
r	O		1000	2.37	Search radius in meter around the given coordinate.
direction	O	See Section 2.3 or Section 2.4	-	2.37	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	O	See Section 1.2.2	-	2.37	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	O	See Section 1.2.2	-	2.37	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	O	0-1439	60	2.37	Set the interval size in minutes.
maxJourneys	O		-1	2.37	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).
maxStations	O		30	2.37	Maximum number of stations near the center location to be considered.

products	O		-	2.37	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. Values are retrievable by Data Information service. 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	O	All operator codes or names from HAFAS raw data, accessible via Data Information service.	-	2.37	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 journeys, negate it by putting ! in front of it. Example: Filter for operator A and B: `operators=A,B.`
categories	O	All category codes or names from HAFAS raw data, accessible via Data Information service.	-	2.37	Only journeys provided by the given categories are part of the result. To filter multiple categories, separate the codes by comma. If the category should not be part of the journeys, negate it by putting ! in front of it. Example: Filter for category A and B: `categories=A,B.`

lines	O	-	-	2.37	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.
attributes	O	See Section 1.2.3	-	2.37	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	O	-	-	2.37	Filter boards by platform. Multiple platforms are separated by comma.
passlist	O	0 or 1	0	2.37	Include a list of all passed waystops?
passlistMaxStops	O		-	2.37	Maximum number of stops including requested stop and last stop.
minDur	O		-	2.37	Minimum duration a journey has left to be returned.
baim	O	0 or 1	0	2.37	Enables/disables BAIM information.
rtMode	O	OFF or SERVER_DEFAULT	SERVER_DEFAULT	2.37	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. SERVER_DEFAULT – one of the above configured in the HAFAS server back end.

type	M	DEP, DEP_EQUIVS, DEP_MAST, DEP_STATION	DEP	2.37	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 all 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.

2.18.2. Example

Request: Departure board for three stations with IDs 12345, 98765 and 456234 on June 1st, 2020, at 6 pm

```
<baseurl>/nearbyDepartureBoard?originCoordLong=&originCoordLat=&r=&date=2020-06-
01&time=18:00
```

Result: (abbreviated)

```xml
<DepartureBoard serverVersion="2.45.2" dialectVersion="2.45-Rejseplanen"
xmlns="http://hacon.de/hafas/proxy/hafas-proxy" requestId="1483016905755">
    <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>
```

2.18.3. 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.

2.18.4. 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).

2.19. Nearby Arrival Board (nearbyArrivalBoard)

Arrival boards for stations in a certain radius nearby a given coordinate can be retrieved by a call to the `nearbyArrivalBoard` 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. The default radius is 1000m and the default amount of stations taken into account is 30.

Note: The result list always contains all arrivals running the the last minute found even if the requested maximum was overrun. All arrivals are mixed into one single list.

Service added in version 2.37.

2.19.1. Request Parameters

- Use M: Mandatory
- Use O: Optional

Name	Use	Range	Default	Est.	Description
accessId	M	-	-	1.0	Access ID for identifying the requesting client.
originCoordLat	M	See Section 1.2.1	-	2.37	Latitude of centre coordinate.
originCoordLong	M	See Section 1.2.1	-	2.37	Longitude of centre coordinate.
r	O		1000	2.37	Search radius in meter around the given coordinate.
direction	O	See Section 2.3 or Section 2.4	-	2.37	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	O	See Section 1.2.2	-	2.37	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	O	See Section 1.2.2	-	2.37	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	O	0-1439	60	2.37	Set the interval size in minutes.
maxJourneys	O		-1	2.37	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).
maxStations	O		30	2.37	Maximum number of stations near the center location to be considered.

products	O		-	2.37	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. Values are retrievable by Data Information service. 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	O	All operator codes or names from HAFAS raw data, accessible via Data Information service.	-	2.37	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 journeys, negate it by putting ! in front of it. Example: Filter for operator A and B: `operators=A,B.`
categories	O	All category codes or names from HAFAS raw data, accessible via Data Information service.	-	2.37	Only journeys provided by the given categories are part of the result. To filter multiple categories, separate the codes by comma. If the category should not be part of the journeys, negate it by putting ! in front of it. Example: Filter for category A and B: `categories=A,B.`

lines	O	-	-	2.37	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.
attributes	O	See Section 1.2.3	-	2.37	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	O	-	-	2.37	Filter boards by platform. Multiple platforms are separated by comma.
passlist	O	0 or 1	0	2.37	Include a list of all passed waystops?
passlistMaxStops	O		-	2.37	Maximum number of stops including requested stop and last stop.
minDur	O		-	2.37	Minimum duration a journey has left to be returned.
baim	O	0 or 1	0	2.37	Enables/disables BAIM information.
rtMode	O	OFF or SERVER_DEFAULT	SERVER_DEFAULT	2.37	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. SERVER_DEFAULT – one of the above configured in the HAFAS server back end.

type	M	ARR, ARR_EQUIVS, ARR_MAST, ARR_STATION	ARR	2.37	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.

2.19.2. Example

Request: Arrival board for xxx on xxx

```
<baseurl>/nearbyArrivalBoard?originCoordLong=&originCoordLat=&r=&date=2020-06-
01&time=18:00
```

Result: (abbreviated)

```
<ArrivalBoard serverVersion="2.45.2" dialectVersion="2.45-Rejseplanen"
xmlns="http://hacon.de/hafas/proxy/hafas-proxy" requestId="1483016905755">
    <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>
```

2.19.3. 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.

2.19.4. 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).

2.20. 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.

Service added in version 1.0.

2.20.1. Request Parameters

- Use M: Mandatory
- Use O: Optional

Name	Use	Range	Default	Est.	Description
accessId	M	-	-	1.0	Access ID for identifying the requesting client.
id	M	-	-	1.0	Specifies the internal journey id of the journey shall be retrieved. Maximum length 512. It may be necessary to escape the | character by its URL encoding %7C.
date	O	See Section 1.2.2	-	1.0	Day of operation By default the current server date is used
poly	O	0 or 1	0	1.11	Enables/disables the calculation of the polyline for each leg of the trip except any GIS route.
polyEnc	O	DLT, GPA, N	N	1.11	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	O	0 or 1	0	1.0	Enables/disables the return of stops having no alighting and no boarding in its passlist for each leg of the trip.

rtMode	O	FULL, INFOS, OFF, REALTIME, SERVER_DEFAULT	-	1.0	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 and all results are combined. 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.
fromId	O	-	-	2.3	Specifies the station/stop ID the partial itinerary shall start from.
fromIdx	O		-	2.3	Specifies the station/stop index the partial itinerary shall start from.

toId	O	-	-	2.3	Specifies the station/stop ID the partial itinerary shall end at.
toIdx	O		-	2.3	Specifies the station/stop index the partial itinerary shall end at.
baim	O	0 or 1	0	2.7.3	Enables/disables BAIM search and response.

2.20.2. 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)

```xml
<JourneyDetail serverVersion="2.45.2" dialectVersion="2.45-Rejseplanen"
xmlns="http://hacon.de/hafas/proxy/hafas-proxy" requestId="1483016905755">
    <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>
```

2.21. 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.

Service added in version 1.0.

2.21.1. Request Parameters

- Use M: Mandatory
- Use O: Optional

Name	Use	Range	Default	Est.	Description
accessId	M	-	-	1.0	Access ID for identifying the requesting client.
match	O	-	-	1.0	Matching criteria like train name, number or both. **Even if optional, this field should be filled.** Example: • ICE 827 • RE 48 • S1
filters	O	-	-	1.0	Filter definitions. Read additional document if available for your installation.
stations	O	-	-	1.0	**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	O	-	-	1.0	Filter for station by extId. Matches if the given value part of at least one station in the itinerary.
uic	O	-	-	1.0	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	O	All operator codes or names from HAFAS raw data, accessible via Data Information service.	-	1.23.19	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	O	All line numbers or names from HAFAS raw data.	-	2.7.2	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	O	See Section 1.2.4	-	1.0	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	O	See Section 1.2.2	-	1.0	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	O	See Section 1.2.2	-	1.0	Time the service operates according to scheduled data. If not provided, the whole day is taken into account.
showPassingPoints	O	0 or 1	0	1.23.15	Enables/disables the return of stops having no alighting and boarding.

2.21.2. 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="2.45.2" ref="1|26883|4|85|14112016"
dialectVersion="2.45-Rejseplanen" xmlns="http://hacon.de/hafas/proxy/hafas-
proxy" requestId="1483016905755">
  <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=
"FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFE7FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFF0000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000" />
  <ServiceDays sDaysR="nicht täglich" sDaysI="14. Nov bis 10. Dez 2016
täglich"
    sDaysB=
"FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFF0000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000" />
  <lastPassRouteIdx>0</lastPassRouteIdx>
  <lastPassStopRef>0</lastPassStopRef>
</JourneyDetail>
```

2.22. 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.

Service added in version 1.0.

2.22.1. Request Parameters

- Use M: Mandatory
- Use O: Optional

Name	Use	Range	Default	Est.	Description
accessId	M	-	-	1.0	Access ID for identifying the requesting client.
llLat	M	See Section 1.2.1	-	1.0	Lower left latitude of bounding box.
llLon	M	See Section 1.2.1	-	1.0	Lower left longitude of bounding box.
urLat	M	See Section 1.2.1	-	1.0	Upper right latitude of bounding box.
urLon	M	See Section 1.2.1	-	1.0	Upper right longitude of bounding box.
operators	O	All operator codes or names from HAFAS raw data, accessible via Data Information service.	-	1.0	Filter for operators. To filter multiple operators, separate the codes by comma. Example: Filter for operator A and B: `operators=A,B`.

products	O		-	1.0	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. Values are retrievable by Data Information service. 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	O	See Section 1.2.3	-	1.0	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	O	-	-	1.0	Filter journeys by one or more journey id. Multiple journey ids are separated by comma.
lines	O	-	-	1.0	Only journeys running the given line are part of the result. To filter multiple lines, separate the codes by comma.
infotexts	O	See Section 1.2.4	-	1.0	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	O	or 1-1000	1000	1.0	Maximum number of journeys in response.
periodSize	O		30000	2.24	Size of interval journey positions should be retrieved in milliseonds.
periodStep	O		2000	2.24	Size of interval steps.

date	O	See Section 1.2.2	-	1.0	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	O	See Section 1.2.2	-	1.0	Time the service operates according to scheduled data. If not provided, the whole day is taken into account.
positionMode	O	CALC, CALC_REPORT, REPORT_ONLY	REPORT_ONLY	1.0	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; Default REPORT_ONLY REPORT_ONLY: Only get back reported positions. CALC_REPORT: Use reported position if available, calculate if not. CALC: Calculate all positions

2.22.2. 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&inf
otexts=CT|TGV
```

Result: (abbreviated)

```
<?xml version="1.0" encoding="UTF-8"?>
<JourneyList serverVersion="2.45.2" dialectVersion="2.45-Rejseplanen"
xmlns="http://hacon.de/hafas/proxy/hafas-proxy" requestId="1483016905755">
    <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>
```

2.23. 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.

Service added in version 1.0.

2.23.1. Request Parameters

- Use M: Mandatory
- Use O: Optional

Name	Use	Range	Default	Est.	Description
accessId	M	-	-	1.0	Access ID for identifying the requesting client.
match	O	-	-	1.0	Matching criteria like train name, number or both. **Even if optional, this field should be filled.** Example: • ICE 827 • RE 48 • S1
filters	O	-	-	1.0	Filter definitions. Read additional document if available for your installation.
stations	O	-	-	1.0	**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	O	-	-	1.0	Filter for station by extId. Matches if the given value part of at least one station in the itinerary.

uic	O	-	-	1.0	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	O	All operator codes or names from HAFAS raw data, accessible via Data Information service.	-	1.23.19	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	O	All line numbers or names from HAFAS raw data.	-	2.7.2	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	O	See Section 1.2.4	-	1.0	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	O	See Section 1.2.2	-	1.0	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	O	See Section 1.2.2	-	1.0	Time the service operates according to scheduled data. If not provided, the whole day is taken into account.
dateB	O	See Section 1.2.2	-	2.26	First day of period.
dateE	O	See Section 1.2.2	-	2.26	Last day of period.
timeB	O	See Section 1.2.2	-	2.26	Period start time the journey needs to cover partly at least.

timeE	O	See Section 1.2.2	-	2.26	Period end time the journey needs to cover partly at least.
showPassingPoints	O	0 or 1	0	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.

2.23.2. 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="2.45.2" dialectVersion="2.45-Rejseplanen"
xmlns="http://hacon.de/hafas/proxy/hafas-proxy" requestId="1483016905755">
  <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>
```

2.24. Line Search (linesearch)

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

Service added in version 2.5.

2.24.1. Request Parameters

- Use M: Mandatory
- Use O: Optional

Name	Use	Range	Default	Est.	Description
accessId	M	-	-	1.0	Access ID for identifying the requesting client.
operators	M	All operator codes or names from HAFAS raw data, accessible via Data Information service.	-	2.5	List of operators seperated by comma.
date	O	-	-	2.5	Day of operation. Represented in the format YYYY-MM-DD.

2.24.2. 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="2.45.2" dialectVersion="2.45-Rejseplanen" xmlns=
"http://hacon.de/hafas/proxy/hafas-proxy" requestId="1483016905755">
  <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>
```

2.25. Line Match (linematch)

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

Service added in version 2.8.1.

2.25.1. Request Parameters

- Use M: Mandatory
- Use O: Optional

Name	Use	Range	Default	Est.	Description
accessId	M	-	-	1.0	Access ID for identifying the requesting client.
match	M	-	-	2.8.1	Matching criteria like train name, number or both.
operators	O	All operator codes or names from HAFAS raw data, accessible via Data Information service.	-	2.8.1	List of operators seperated by comma.

2.25.2. Example

Request: Get the list of lines by matching S9

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

Result: (abbreviated)

```
<LineList serverVersion="2.45.2" dialectVersion="2.45-Rejseplanen" xmlns=
"http://hacon.de/hafas/proxy/hafas-proxy" requestId="1483016905755">
  <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>
```

2.26. 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.

Service added in version 2.5.

2.26.1. Request Parameters

- Use M: Mandatory
- Use O: Optional

Name	Use	Range	Default	Est.	Description
accessId	M	-	-	1.0	Access ID for identifying the requesting client.
lineId	M	-	-	2.5	Specifies the internal line id of the line shall be retrieved.
date	M	-	-	2.5	Day of operation. Represented in the format YYYY-MM-DD.

2.26.2. 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
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<LineList serverVersion="2.45.2" dialectVersion="2.45-Rejseplanen" xmlns=
"http://hacon.de/hafas/proxy/hafas-proxy" requestId="1483016905755">
  <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>
```

2.27. 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.

Service added in version 2.5.

2.27.1. Request Parameters

- Use M: Mandatory
- Use O: Optional

Name	Use	Range	Default	Est.	Description
accessId	M	-	-	1.0	Access ID for identifying the requesting client.
lineId	M	-	-	2.5	Specifies the internal line id of the line shall be retrieved.
date	M	-	-	2.5	Day of operation. Represented in the format YYYY-MM-DD.
orderBy	O	DEPARTURE_ASC, DEPARTURE_DESC, ARRIVAL_ASC, ARRIVAL_DESC	-	2.5.3	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	O	0 or 1	0	2.7.2	Enables/disables the return of HIM messages.

2.27.2. 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="2.45.2" dialectVersion="2.45-Rejseplanen" xmlns=
"http://hacon.de/hafas/proxy/hafas-proxy" requestId="1483016905755">
  <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>
```

2.28. 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.

Service added in version 1.0.

2.28.1. Request Parameters

- Use M: Mandatory
- Use O: Optional

Name	Use	Range	Default	Est.	Description
accessId	M	-	-	1.0	Access ID for identifying the requesting client.
dateB	O	See Section 1.2.2	-	1.0	Sets the event period start date.
dateE	O	See Section 1.2.2	-	1.0	Sets the event period end date.
timeB	O	See Section 1.2.2	-	1.0	Sets the event period start time.
timeE	O	See Section 1.2.2	-	1.0	Sets the event period end time.
weekdays	O	-	-	2.4	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	O	-	-	1.0	List of HIM message IDs seperated by comma.
hierarchicalView	O	0 or 1	0	2.8.1	Return parent messages with childs.
operators	O	All operator codes or names from HAFAS raw data, accessible via Data Information service.	-	1.0	List of operators seperated by comma.
categories	O	-	-	1.0	List of train categories seperated by comma.
channels	O	-	-	1.0	List of channels seperated by comma.
companies	O	-	-	1.0	List of companies seperated by comma.
lines	O	-	-	2.4	Only HIM messages for the given line are part of the result. To filter multiple lines, separate the codes by comma.
lineids	O	-	-	2.11	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	O	-	-	2.4	List of (external) station ids to be filtered for seperated by comma.
fromstation	O	-	-	2.4	Filter messages by line segment starting at this station given as (external) station id.
tostation	O	-	-	2.4	Filter messages by line segment travelling in direction of this station given as (external) station id.
bothways	O	0 or 1	-	2.4	If enabled, messages in both directions - from 'fromstation' to 'tostation' as well as from 'tostation' to 'fromstation' are returned

trainnames	O	-	-	2.4	List of train name to be filtered for seperated by comma.
metas	O	-	-	2.4	List of predefined filters seperated by comma.
himcategory	O	-	-	1.23.9	HIM category, e.g. Works and/or Disturbance. Value depends on your HAFAS server data.
himtags	O	-	-	2.10	HIM Tags. Value depends on your HAFAS server data. Return HIM messages having these tag(s) only. Multiple values are separated by comma. Note: HIM tags differ from HIM text tags.
regions	O	-	-	2.16	Filter for HIM messages based on regions defined in HAFAS raw data. Seperated by comma. Available regions can be retrieved by /datainfo service.
himtext	O	-	-	2.4	Filter for HIM messages containing the given free text message seperated by comma.
himtexttags	O	-	-	2.16	Return HIM texts having this text tag(s) only. Multiple values are separated by comma. Note: HIM text tags differ from HIM tags.

exthimtext	O	-	-	2.4	Extended filter based on tags and and corresponding localized text fragments. The base filter structure follows the format 'tag|text'. For a case-insensitive search, append '|i' to the filter ('tag|text|i'). To exclude a combination of tag and text from the search, prepend the filter with '!' ('!tag|text'). When filtering messages that must satisfy multiple conditions (AND), use commas to separate filters (e.g., 'tag1|text1,tag2|text2'). For filtering messages that need to meet at least one of the given conditions (OR), separate filters with semicolons (e.g., 'tag1|text1;tag2|text2').
additionalfields	O	-	-	2.22	List of additional fields and values to be filtered for. Two filter options are available: Filter by key only: additionalfields=key or filter by key and value: additionalfields=key:value. Multiple filters are separated by comma like additionalfields=keyA,keyB or additionalfields=key:value.
extInfo	O	-	-	2.38	Filter for HIM messages based on external ids and id sources. Different items are seperated by comma. Value seperated by pipe symbol. To negate put ! in front of the item.
poly	O	0 or 1	0	1.23.12	Enables/disables returning of geo information for affected edges and regions if available and enabled in the backend.

searchmode	O	MATCH, NOMATCH, TFMATCH	-	1.0	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	O	ALL or OFF	-	2.5	Define how to return affected journeys OFF: do not return affected journeys. ALL: return affected journeys
affectedJourneyStopMode	O	ALL, IMP, OFF	-	2.5	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	O	-	-	2.4.1	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	O		-	2.4	Filter for HIM messages having at least this priority.
maxprio	O		-	2.4	Filter for HIM messages having this priority as maximum.
llLat	O	See Section 1.2.1	-	2.24.0	Lower left latitude of bounding box.
llLon	O	See Section 1.2.1	-	2.24.0	Lower left longitude of bounding box.
urLat	O	See Section 1.2.1	-	2.24.0	Upper right latitude of bounding box.
urLon	O	See Section 1.2.1	-	2.24.0	Upper right longitude of bounding box.

2.28.2. 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
<?xml version="1.0" encoding="UTF-8"?>
<HimMessages serverVersion="2.45.2" dialectVersion="2.45-Rejseplanen"
xmlns="http://hacon.de/hafas/proxy/hafas-proxy" requestId="1483016905755">
  <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>
```

2.28.3. Request hierarchical view / Event view

HAFAS HIM provides the ability to group messages by events. If there is a root cause which leads to more actions, messages based on that root cause event can be added as child messages.

If you want HIM Search service to return that relation between HIM messages, you need to set the parameter `hierarchicalView=1`.

The response than has messages on the top level which do not have a parent. In `message` child elements the direct ancestors are available if any.

2.29. 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.

Service added in version 2.5.

Since version 2.20, map layer information are provided if configured.

2.29.1. Request Parameters

- Use M: Mandatory
- Use O: Optional

Name	Use	Range	Default	Est.	Description
accessId	M	-	-	1.0	Access ID for identifying the requesting client.

2.29.2. Example

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

Result: (abbreviated)

```xml
<DataInfo begin="2019-06-17" end="2019-09-15" serverVersion="2.45.2"
dialectVersion="2.45-Rejseplanen" xmlns="http://hacon.de/hafas/proxy/hafas-
proxy" requestId="1483016905755">
  <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>
```

2.30. 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.

Service added in version 1.23.8.

2.30.1. Request Parameters

- Use M: Mandatory
- Use O: Optional

Name	Use	Range	Default	Est.	Description
accessId	M	-	-	1.0	Access ID for identifying the requesting client.

2.30.2. Example

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

Result: (abbreviated)

```
<TimetableInfoList serverVersion="2.45.2" dialectVersion="1.23" requestId=
"1544174154581" begin="2017-12-10" end="2019-12-14">
  <TimetableInfo poolId="100" ident="s43bl" date="2018-11-27" time=
"13:19:59" type="ST" begin="2017-12-10" end="2018-12-08" comment="optional
comment"/>
  <TimetableInfo poolId="101" ident="s6qzl" date="2018-11-28" time=
"10:48:47" type="ST" begin="2018-12-09" end="2019-12-14" comment="optional
comment"/>
  <TimetableInfo poolId="102" ident="nbyyp" date="2018-08-06" time=
"15:07:27" type="ADR" comment="optional comment"/>
  <TimetableInfo poolId="103" ident="ifqyf" date="2018-05-03" time=
"11:16:53" type="POI" comment="optional comment"/>
</TimetableInfoList>
```

2.31. Tariff Stop (trfStop)

This service returns the "default" stop (stop number and name) for the given zone.

2.31.1. Request Parameters

- Use M: Mandatory
- Use O: Optional

Name	Use	Range	Default	Est.	Description
accessId	M	-	-	1.0	Access ID for identifying the requesting client.
z	M	-	-	2.35	Zone number (4-digit format) for which the "major" stop has to be determined

2.31.2. Example

Request: Get the default stop

`<baseurl>/trfStop?z=32784385&format=json`

Result: (abbreviated)

```
{
    "res": {
        "id": 8600617,
        "name": "Roskilde St."
    }
}
```

2.31.3. Error codes

Error code	Description
1100	No zone found for the coordinates
1105	No stop for the zone
1604	No tariff data
9900	Missing or invalid parameters

Error structure and the list of all tariff-related error codes can be found here Section 4.2

2.32. Convert Zones (convertZones)

The convert zones service returns tariff details for a specified tariff context.

2.32.1. Request Parameters

- Use M: Mandatory
- Use O: Optional

Name	Use	Range	Default	Est.	Description
accessId	M	-	-	1.0	Access ID for identifying the requesting client.
uc	O	fixed, school, combined, self	fixed	2.35	Defines the use case to use.
if	M	4, 8, A	-	2.35	Input format (4: 4-digit zones, 8: 8-digit zones, A: Name/Number zones like TH01)
of	M	4, 8, A	-	2.35	Output format (4: 4-digit zones, 8: 8-digit zones, A: Name/Number zones like TH01)

o	O	-	-	2.35	Allows specifying the origin zone explicitly. The converted origin zone is returned separately and it is also removed from the normal zone list if it is specified there. (must be set, if parameter d is set)
d	O	-	-	2.35	Allows specifying the destination zone explicitly. The converted destination zone is returned separately and it is also removed from the normal zone list if it is specified there. (must be set, if parameter o is set)
z	O	-	-	2.35	Comma separated list of zones
zg	O	-	-	2.35	Comma separated list of zone groups. Only for if=8.
fa	O	-	-	2.35	Fare Area. Only for if=A.
gt	O	-	27	2.35	Grouping Threshold (try to group zones to this number or below). Only applicable for of=8. To disable grouping, pass in a very high number (e.g. 9999). For uc=self, if o/d is not specified, the default is 24

2.32.2. Example

Request: Get the tariff details

```
<baseurl>/convertZones?accessId=abc&&if=4&of=4format=json
```

Result: (abbreviated)

```
{
    "res": {
        "z": [
            1041
        ],
        "zg": [],
        "fs": 1
    }
}
```

2.32.3. Error codes

Error code	Description
1104	Missing zone group
1105	No zone group for zone
1604	No tariff data
1806	It was not possible to group the zones as required
9900	Missing or invalid parameters

Error structure and the list of all tariff-related error codes can be found here Section 4.2

2.33. Zone From Coordinate (zoneFromCoordinate)

This service returns the zone the given coordinate is in.

2.33.1. Request Parameters

- Use M: Mandatory
- Use O: Optional

Name	Use	Range	Default	Est.	Description
accessId	M	-	-	1.0	Access ID for identifying the requesting client.
lat	M	-	-	2.35	Latitude value in WGS84 format
long	M	-	-	2.35	Longitude value in WGS84 format

2.33.2. Example

Request: Get the zone for coordinates lat=55.629924, long=12.649656

`<baseurl>/zoneFromCoordinate?lat=55.629924&long=12.649656&format=json`

Result: (abbreviated)

```
{
    "res":{
        "z":1004
    }
}
```

2.33.3. Error codes

Error code	Description
1100	No zone found for coordinates
9900	Missing or invalid parameters

Error structure and the list of all tariff-related error codes can be found here Section 4.2

2.34. SP Price (spPrice)

This service calculates the price and validity zones for season pass products based on the origin and destination addresses/stop points.

2.34.1. Request Parameters

- Use M: Mandatory
- Use O: Optional

Name	Use	Range	Default	Est.	Description
accessId	M	-	-	1.0	Access ID for identifying the requesting client.
context	O	-	-	2.35	Optional context that can be passed in and is returned in the response.
mode	O	gvss	-	2.35	Support for special modes of the spPrice service. Currently only supports the GVSS SELF API.

oaddr	M	-	-	2.35	Origin address/stop point (zones are not supported).
daddr	M	-	-	2.35	Destination address/stop point (zones are not supported).
vaddr	O	-	-	2.35	Optional via address/stop point (zones are not supported).
psgs	O	-	["A"]	2.35	Array of passenger types. "A": Adult, "C": Child, "B": Bike, "P": Pensioner, "Y": Young, "H": Handicapped, "D" Example: ["A","C"]
with_purse	O	0 or 1	0	2.35	Specifies what kind of card is used. 1: Combi card (i.e., Season Pass with an additional purse). 0: A simple Season Pass Alternative values: true/false
issue_on	O	-	-	2.35	Issue date of the season pass. Also used to select the Rejsekort EOD data set to use. If not specified, the current date is used.
valid_from	O	-	-	2.35	First day of validity. Also defines the date for the 24h search. Always the next normal business day on or the given date is used. Default: Same as the issue_on date.
valid_to	O	-	-	2.35	Last day of validity. If not specified, a 30 day season pass is assumed.
journey_info	O	0, 1, 2, 3	0	2.35	Level of information about journeys: 0: No journey information. 1: Basic Journey Information without leg-information. 2: Additional basic leg information (only origin/destination for each leg). 3: All locations for legs.

only_valid_journeys	O	0 or 1	1	2.35	1: Journeys must be valid for the (relational) season pass. Otherwise the journey is not in the result. In this case no "valid" fields are set in the response. 0: All journeys for which a (relational) season pass exists (based on origin/destination), are returned. The valid flag is set to "false" if the journey is not inside the validity zones. Alternative values: true/false
add_zonals	O	0 or 1	0	2.35	1: Return zonal season passes in addition to relational season passes if available (i.e., less than 9 zones selected). 0: Only return zonal season passes if available and no relational season pass is available for the relation. Alternative values: true/false
max_products	O	-	10	2.35	The maximum number of products returned. This parameter restricts the number of products for all cases but is mostly applicable to the longest distance season passes in Takst Vest where all combinations of products are calculated.
fs	O	-	-	2.35	Array of fare sets for which to consider journeys and calculate prices. Default: All fare sets. Example: [8,40,21]
ignore_min_zones	O	0 or 1	0	2.35	1: Return season pass options (with 2 zones each) if the journey is inside 1 zone (works for Takst Vest and Zealand). 0: Do not return season passes if the journey is inside 1 zone. Alternative values: true/false

spPriceOutRet	O	0 or 1	0	2.35	1: Do the interval search in both directions and combine the results. 0: Do the interval search only in the specified direction.The default of this parameter can be changed in the cgi.cfg on the server. The parameter in the URL always takes precedence over the configuration.
maxWalkOrigin	O	-	2000	2.35	Maximum length in meters of walk at the beginning of the journey.
maxWalkDest	O	-	2000	2.35	Maximum length in meters of walk at the destination of the journey.
minFreq	O	-	-	2.35	Minimum frequency for the product. Filters out all products with a aggregated frequency of all assigned journeys below the specified number..

sort	O	-	-	2.35	Specify the order of products. The following fields can be used for filtering: frequency: The frequency of the product based on the sum of the frequencies from all assigned journeys distanceInZones: The pay zone distance (as in field "nzones"). numberOfValidZones: The number of zones the season pass is valid in (as number of entries in field "zones"). Ascending and descending sort order can be specified by appending "" for ascendingMultiple fields can be specified with different ascending or descending order. All fields have to be specified in a semicolon separated list. The order in which sort fields are specified defines the order: First order after the first field, if this field is the same, order after the second sort field, etc. E.g., "frequency-;distanceInZones;numberOfValidZones-" (i.e., highest frequency first, with same frequency prefer lower distance (i.e., lower price) and secondary prefer higher number of zones). Example: frequency-;distanceInZones+;numberOfValidZones-
withCtxRecon	O	0 or 1	1	2.35	Specifies if the reconstruction context is returned in the result. 1: Return the reconstruction context in the field res.products.ctxRecon of the result. 0: Do not return the reconstruction context in the result. Alternative values: true/false

2.34.2. Example

Request: Get the price and validity zones

```
<baseurl>/spPrice?format=json&context=123456&oaddr=Ringsted&daddr=Roskilde&psgs=["A",
"C"]&fs=[8]&issue_on=2016-07-27&valid_from=2016-08-01&valid_to=2016-08-
31&spPriceOutRet=1
```

Result: (abbreviated)

```
{
  "c" : "123456",
  "res" : {
    "date" : "2016-07-12",
    "oaddr" : {"name":"Aarhus H","uic":" 008600053"},
    "daddr" :{"name":"Thisted St.","uic":"008600404"},
    "products" : [{
        "ctxRecon" : "<product reconstruction context>",
        "id" : "SP1",
        "prod" : 8192096,
        "prodname" : "Pilot Periode Kombi Hovedstad (ZONAL)",
        "type" : "Z",
        "gvm" : "Z",
        "ozone" : 1101,
        "ozone8" : 32850021,
        "dzone" : 1210,
        "dzone8" : 32833746,
        "fs" : 7,
        "fsname" : "Sjælland",
        "nzones" : 4,
        "prcZones" : [1101, 1204],
        "prcZones8" : [32850021, 32833740],
        "zones" : [1101, 1203, 1204, 1210],
        "zones8" : [32850021, 32833739, 32833740, 32833746],
        "zonegroups8" : {"err":{"no":1806,"msg":"convertZones: Unable to
group zones sufficiently"}},
        "fares" : [{
            "psg" : "A",
            "price" : 1500,
            "bcprice" : 2000,
            "bcregionalprice" : 1600,
            "options" : [{"type" : "metro", "price" : 8000}]
          }
        ],
        "journeys" : [{
            "id" : "C0-0",
            "freq" : 3,
            "valid" : true,
            "zones" : [1101, 1146, 1203, 1204],
            "legs" : [{
                "trans" : "T",
                "locs" : [{
                    "name" : "Aarhus H",
                   "uic" : "008600053",
                    "long" : 10204761,
                    "lat" : 56150444
                  }, {
                    "name" : "Skive St.",
                    "uic" : "008600183"
                  }
                ]
              }
            ]
          }
        ]
      }
    ]
  }
}
```

2.34.3. Error codes

Error code	Description
1	HAFAS error (i.e., no journey can be found for the given addresses at the specified date for some reason, this error can also mean that the server is not available)
1001	Mandatory request parameters missing for 24h-search
1500	No specific single ticket/season pass for the fare set/date/passenger type/zone istance
1604	No tariff data set for the date (same as below)
1701	Inconsistent tariff data (no data for the fare set)
1803	All journeys use too many zones for the zonal season pass.
1806	Number of zones passed by the journey is below the minimum.
2010	No season pass available in the area
3514	Invalid request parameters for the tariff request
3519	All products removed by filter (e.g., because of the frequency filter)
9901	Internal server error

Error structure and the list of all tariff-related error codes can be found here Section 4.2

2.35. SP Price Reconstruction (spPriceReconstruction)

This service calculates the price and validity zones for season pass products based on the journey reconstruction context from a previous request.

2.35.1. Request Parameters

- Use M: Mandatory
- Use O: Optional

Name	Use	Range	Default	Est.	Description
accessId	M	-	-	1.0	Access ID for identifying the requesting client.
context	O	-	-	2.35	Optional context that can be passed in and is returned in the response.
mode	O	gvss	-	2.35	Support for special modes of the spPrice service. Currently only supports the GVSS SELF API.
ctxRecons	M	-	-	2.35	The journey reconstruction context from a previous request.
psgs	O	-	["A"]	2.35	Array of passenger types. "A": Adult, "C": Child, "B": Bike, "P": Pensioner, "Y": Young, "H": Handicapped, "D" Example: ["A","C"]
with_purse	O	0 or 1	0	2.35	Specifies what kind of card is used. 1: Combi card (i.e., Season Pass with an additional purse). 0: A simple Season Pass Alternative values: true/false
issue_on	O	-	-	2.35	Issue date of the season pass. Also used to select the Rejsekort EOD data set to use. If not specified, the current date is used.
valid_to	O	-	-	2.35	Last day of validity. If not specified, a 30 day season pass is assumed.

journey_info	O	0, 1, 2, 3	0	2.35	Level of information about journeys: 0: No journey information. 1: Basic Journey Information without leg-information. 2: Additional basic leg information (only origin/destination for each leg). 3: All locations for legs.
only_valid_journeys	O	0 or 1	1	2.35	1: Journeys must be valid for the (relational) season pass. Otherwise the journey is not in the result. In this case no "valid" fields are set in the response. 0: All journeys for which a (relational) season pass exists (based on origin/destination), are returned. The valid flag is set to "false" if the journey is not inside the validity zones. Alternative values: true/false
add_zonals	O	0 or 1	0	2.35	1: Return zonal season passes in addition to relational season passes if available (i.e., less than 9 zones selected). 0: Only return zonal season passes if available and no relational season pass is available for the relation. Alternative values: true/false
max_products	O	-	10	2.35	The maximum number of products returned. This parameter restricts the number of products for all cases but is mostly applicable to the longest distance season passes in Takst Vest where all combinations of products are calculated.
fs	O	-	-	2.35	Array of fare sets for which to consider journeys and calculate prices. Default: All fare sets. Example: [8,40,21]

ignore_min_zones	O	0 or 1	0	2.35	1: Return season pass options (with 2 zones each) if the journey is inside 1 zone (works for Takst Vest and Zealand). 0: Do not return season passes if the journey is inside 1 zone. Alternative values: true/false
minFreq	O	-	-	2.35	Minimum frequency for the product. Filters out all products with a aggregated frequency of all assigned journeys below the specified number..
sort	O	-	-	2.35	Specify the order of products. The following fields can be used for filtering: frequency: The frequency of the product based on the sum of the frequencies from all assigned journeys distanceInZones: The pay zone distance (as in field "nzones"). numberOfValidZones: The number of zones the season pass is valid in (as number of entries in field "zones"). Ascending and descending sort order can be specified by appending "" for ascendingMultiple fields can be specified with different ascending or descending order. All fields have to be specified in a semicolon separated list. The order in which sort fields are specified defines the order: First order after the first field, if this field is the same, order after the second sort field, etc. E.g., "frequency-;distanceInZones;numberOfValidZones-" (i.e., highest frequency first, with same frequency prefer lower distance (i.e., lower price) and secondary prefer higher number of zones). Example: frequency-;distanceInZones+;numberOfValidZones-

withCtxRecon	O	0 or 1	1	2.35	Specifies if the reconstruction context is returned in the result. 1: Return the reconstruction context in the field res.products.ctxRecon of the result. 0: Do not return the reconstruction context in the result. Alternative values: true/false

2.35.2. Example

Request: Get the price and validity zones

```
<baseurl>/spPriceReconstruction?format=json&ctxRecon=123456ABCDE&psgs=["A","C"]&fs=[8
]&issue_on=2016-07-27&valid_to=2016-08-31
```

Result: (abbreviated)

```
{
  "c" : "123456",
  "res" : {
    "date" : "2016-07-12",
    "oaddr" : {"name":"Aarhus H","uic":" 008600053"},
    "daddr" :{"name":"Thisted St.","uic":"008600404"},
    "products" : [{
        "ctxRecon" : "<product reconstruction context>",
        "id" : "SP1",
        "prod" : 8192096,
        "prodname" : "Pilot Periode Kombi Hovedstad (ZONAL)",
        "type" : "Z",
        "gvm" : "Z",
        "ozone" : 1101,
        "ozone8" : 32850021,
        "dzone" : 1210,
        "dzone8" : 32833746,
        "fs" : 7,
        "fsname" : "Sjælland",
        "nzones" : 4,
        "prcZones" : [1101, 1204],
        "prcZones8" : [32850021, 32833740],
        "zones" : [1101, 1203, 1204, 1210],
        "zones8" : [32850021, 32833739, 32833740, 32833746],
        "zonegroups8" : {"err":{"no":1806,"msg":"convertZones: Unable to
group zones sufficiently"}},
        "fares" : [{
            "psg" : "A",
            "price" : 1500,
            "bcprice" : 2000,
            "bcregionalprice" : 1600,
            "options" : [{"type" : "metro", "price" : 8000}]
          }
        ],
        "journeys" : [{
            "id" : "C0-0",
            "freq" : 3,
            "valid" : true,
            "zones" : [1101, 1146, 1203, 1204],
            "legs" : [{
                "trans" : "T",
                "locs" : [{
                    "name" : "Aarhus H",
                   "uic" : "008600053",
                    "long" : 10204761,
                    "lat" : 56150444
                  }, {
                    "name" : "Skive St.",
                    "uic" : "008600183"
                  }
                ]
              }
            ]
          }
        ]
      }
    ]
  }
}
```

2.35.3. Error codes

Error code	Description
1	HAFAS error (i.e., no journey can be found for the given addresses at the specified date for some reason, this error can also mean that the server is not available)
1001	Mandatory request parameters missing for 24h-search
1500	No specific single ticket/season pass for the fare set/date/passenger type/zone distance
1604	No tariff data set for the date (same as below)
1701	Inconsistent tariff data (no data for the fare set)
1803	All journeys use too many zones for the zonal season pass.
1806	Number of zones passed by the journey is below the minimum.

2010	No season pass available in the area
3514	Invalid request parameters for the tariff request
3519	All products removed by filter (e.g., because of the frequency filter)
9901	Internal server error

Error structure and the list of all tariff-related error codes can be found here Section 4.2

2.36. SP Price For Zones (spPriceForZones)

This service calculates the price per day for an adult season pass in the specified fare set, on the specified date for the given number of zones (or zone distance).

2.36.1. Request Parameters

- Use M: Mandatory
- Use O: Optional

Name	Use	Range	Default	Est.	Description
accessId	M	-	-	1.0	Access ID for identifying the requesting client.
td	O	-	-	2.35	Defines the tariff date to use. (format: yyyyMMdd, default: current server date)
fs	M	-	-	2.35	The fare set ID (e.g., 7 for Zealand).
zn	M	-	-	2.35	The number of zones or zone distance for which to determine the price.

2.36.2. Example

Request: Get the price

```
<baseurl>/spPriceForZones?td=20230220&fs=8&zn=5&format=json
```

Result: (abbreviated)

```
{
    "res": {
        "td": 20221229,
        "fs": 43,
        "zn": 12,
        "price": 6700
    }
}
```

2.36.3. Error codes

Error code	Description
1106	No data for fare set
1604	No tariff data
9900	Missing or invalid parameters

Error structure and the list of all tariff-related error codes can be found here Section 4.2

2.37. SP Check (spCheck)

A service for checking season passes in SELF.

2.37.1. Request Parameters

- Use M: Mandatory
- Use O: Optional

Name	Use	Range	Default	Est.	Description
accessId	M	-	-	1.0	Access ID for identifying the requesting client.
onlyVal	O	0 or 1	1	2.35	Enable (1)/disable (0) recreating the season pass from the data passed in Alternative values: true/false
o	M	-	-	2.35	Origin Zone (8-digit)
d	M	-	-	2.35	Destination Zone (8-digit)

z	O	-	-	2.35	Comma separated list of 8-digit zones. If all zones are covered by zone groups for the season ass, this field can be omitted.
zg	O	-	-	2.35	Comma separated list of 8-digit zone groups) If no zone groups exist for the season pass, this field can be omitted.
zt	O	0 or 1	0	2.35	Enable (1) zoneTags or not (0) (currently only for the METRO zone tag). Alternative values: true/false
pt	O	-	-	2.35	Rejsekort Passenger type as defined in EOD data to select special rules (e.g., 1: Adult Strict, 2: Child Strict, 4: Pensioner Strict
ps	O	0 or 1	-	2.35	Purse Status (0: normal season pass, 1: combicard)
l	O	-	-	2.35	Language code based on ISO code
tval	O	fixed or sliding	sliding	2.35	Passed through to the result tval
date	O	-	-	2.35	The validity start date for time validity type "fixed". Only needed and used if tval=fixed. Format: YYYYMMDD

2.37.2. Example

Request: Check season pass

`<baseurl>/spCheck?o=11111111&d=22222222&format=json`

Result: (abbreviated)

```
{
    "status": "OK",
    "rzone.1": 32784385,
    "rzone.3": 32850050,
    "zones": [
        32784386,
        32784416
    ],
    "zonegroups": [],
    "zoneTags": [],
    "tval": "sliding",
    "fs": 7,
    "gval": "zones",
    "calculationmethod": 3,
    "selectedLanguage": "en-GB"
}
```

Error structure:

```
{
    "status": "ERROR",
    "errorCode": "ZONES_NOT_CONNECTED",
    "rzone.1": 32784385,
    "rzone.3": 32850050,
    "zones": [
        32784386,
        32784416
    ],
    "zonegroups": [],
    "zoneTags": [],
    "tval": "sliding",
    "fs": 7,
    "gval": "zones",
    "calculationmethod": 3,
    "selectedLanguage": "en-GB"
}
```

2.37.3. Error codes

Error code	Description
DESTINATION_CHANGED	Calculated destination is different to given destination
INVALID_REQUEST	Missing or invalid parameters
INVALID_TVAL	Time validity not allowed for this fare set

INVALID_ZONES	Invalid zones specified
NEW_GROUPING	Calculated zone groups are different from given zone groups
NO_RELATION	No relation found in OD-matrix (only relevant on Zealand)
NO_SEASON_PASSES_AVAILABLE	No season pass available for the given parameters
NO_TARIFFDATA_FOR_DATE	No tariff data
NOT_GROUPABLE	Grouping is not possible
ORIGIN_CHANGED	Calculated origin is different to given origin
RELATED_ZONES_MISSING	Required zones are missing
SEASON_PASS_ERROR	Invalid zones selected
UNDEFINED	Unexpected error
ZONAL_TOO_LONG	Too many zones for zonal season pass
ZONES_ABOVE_MAXIMUM	Too many zones for a season pass
ZONES_BELOW_MINIMUM	Not enough zones for a valid season pass
ZONES_NOT_CONNECTED	Not all zones are connected

Error structure and the list of all tariff-related error codes can be found here Section 4.2

2.38. SP Zone Check (spZoneCheck)

This services determines the fare set for a number of zones and checks if all zones are connected and if there is any metro zone among them.

2.38.1. Request Parameters

- Use M: Mandatory
- Use O: Optional

Name	Use	Range	Default	Est.	Description
accessId	M	-	-	1.0	Access ID for identifying the requesting client.
z	M	-	-	2.35	Comma separated list of zones

2.38.2. Example

Request: Get the tariff result

```
<baseurl>/spZoneCheck?z=32784385,32784416&format=json
```

Result: (abbreviated)

```
{
    "res":{
        "connected":true,
        "fs":7,
        "metro":true
    }
}
```

2.38.3. Error codes

Error code	Description
1	Unexpected error
2	Missing or invalid parameters
3	No zones specified
4	No tariff data
5	No data for fare set

Error structure and the list of all tariff-related error codes can be found here Section 4.2

2.39. SP Daily Zonal Prices (spDailyZonalPrices)

The method calculates the daily prices for a zonal season pass (i.e. short distance) for all possible numbers of zones (and business class zones) and passenger types. The daily price for a passenger type is defined by the Rejsekort EOD version (based on the date), the fare set and the number of zones (and business class zones). The service returns prices for all zone distances available in the Rejsekort EOD for the fare set (i.e., also for distances above the short distance threshold). This service is designed for Takst Sjælland. Technically it can be used for the Takst Vest fare sets but it is probably not needed there as no zonal tickets are available.

2.39.1. Request Parameters

- Use M: Mandatory
- Use O: Optional

Name	Use	Range	Default	Est.	Description
accessId	M	-	-	1.0	Access ID for identifying the requesting client.
context	O	-	-	2.35	Optional context that can be passed in and is returned in the response.

date	O	-	-	2.35	Date for the request (selects the tariff data version that will be used). Default: current server date.
fs	M	-	-	2.35	Fare Set RefNumber (from Rejsekort EOD) for which to return prices

2.39.2. Example

Request: Get a list of the daily prices for a zonal season pass (i.e. short distance) for all possible numbers of zones

`<baseurl>/spDailyZonalPrices?format=json&context=123456&date=2016-08-31&fs=7`

Result: (abbreviated)

```
{
  "c":"123456",
  "res": {
    "date":"2016-07-12",
    "fs":7,
    "fsname":"S...",
    "fares":[
      {
       "psg":"A",
       "prices":[150,200,250,300,400,500,750,1000,1500],
       "bcprices":[75,100,125,150,200,250,375,500,750],
       "bcregionalprices":[25,30,40,50,65,80,125,175,250],
       "metro":267
      },
      {
       "psg":"C",
       "prices":[75,100,125,150,200,250,375,500,750],
       "bcprices":[38,50,63,75,100,125,188,250,375],
       "bcregionalprices":[15,15,20,25,30,40,60,85,125],
       "metro":133
       }
    ]
  }
}
```

2.39.3. Error codes

Error code	Description

1106	No season pass data for the fare set
1500	No specific season pass for the fare set/date/passenger type
1604	No tariff data set for the date
9900	Missing or malformed request parameters
9903	Internal error (unsupported server)

Error structure and the list of all tariff-related error codes can be found here Section 4.2

2.40. Get Refund (spRefund)

Calculate the refund value of an existing season pass. Cash ticket and season pass prices are calculated as for the Rejseplanen journey planner fare information.

2.40.1. Request Parameters

- Use M: Mandatory
- Use O: Optional

Name	Use	Range	Default	Est.	Description
accessId	M	-	-	1.0	Access ID for identifying the requesting client.
context	O	-	-	2.35	Optional context that can be passed in and is returned in the response.
refund_on	O	-	-	2.35	The date when the season pass should be refunded. If left empty, the current date is used.
issued_on	M	-	-	2.35	The date the season pass was issued.
valid_from	M	-	-	2.35	The start date of the season pass.
valid_to	M	-	-	2.35	The end date of the season pass.
psg	O	A, B, C, D, H, P, Y	A	2.35	Passenger type ("A": Adult, "C": Child, "B": Bike, "P": Pensioner, "Y": Young, "H": Handicapped, "D": Dog)
class	O	S, B, R	S	2.35	"S": Standard, "B": DSB 1', "R": DSB 1' Regional.

metro	O	0 or 1	0	2.35	0: No Metro supplement. 1: Metro supplement included. Alternative values: false/true
fs	M	-	-	2.35	Fare Set RefNumber (as in Rejsekort EOD).
nzones	M	-	-	2.35	The number of zones that defined the price of the season pass..

2.40.2. Example

Request: Calculate the refund value of an existing season pass

```
<baseurl>/spRefund?context=123456&refund_on=2017-02-28&issued_on=2017-01-01&valid_from=2017-01-01&valid_to=2017-03-01&nzones=14&format=json
```

Result: (abbreviated)

```
{
  "c":"123456",
  "res":{
    "price":456000,
    "days_valid":60,
    "days_used":59,
    "refund":0,
    "refund_std":0,
    "refund_bc":0,
    "refund_metro":0,
    "deductions":{
      "sum":501760,
      "day1":26400,
      "day2":26400,
      "day3":26400,
      "day4till22":141360,
      "day23till30":60800,
      "day31tillEnd":220400,
      "bc":0,
      "metro":0
    }
  }
}
```

2.40.3. Error codes

Error code	Description
1106	No season pass data for the fare set
1500	No specific season pass for the fare set/date/passenger type
1503	No single tickets/season passes in fare set
1604	Noo tariff data set for the date
9900	Missing or malformed request parameters
9903	Internal error (unsupported server)

Error structure and the list of all tariff-related error codes can be found here Section 4.2

2.41. Get Stops (stStops)

This service returns a list of tariff zones and associated stops with additional information according to the specified mode. Only mode "DSB" is supported.

2.41.1. Request Parameters

- Use M: Mandatory
- Use O: Optional

Name	Use	Range	Default	Est.	Description
accessId	M	-	-	1.0	Access ID for identifying the requesting client.
context	O	-	-	2.35	Optional context that can be passed in and is returned in the response.
date	O	-	-	2.35	Date for the request (selects the tariff data version that will be used). Default: current server date.
mode	O	DSB	DSB	2.35	Name of mode for determining stops and zones.

2.41.2. Example

Request: Get a list of tariff zones and associated stops

```
<baseurl>/stStops?format=json&context=123456&date=20171231&mode=DSB
```

Result: (abbreviated)

```
{
    "res":{
    "date":20230224,
    "zones":[
      {
        "number":1001,
        "name":"K...)",
        "maxDaysServiced":-1,
        "aliases":["1","1001"],
        "stops":[
          {
            "number":123,
            "name":"K...",
            "relevance":321,
            "aliases":
            [
                "C...",
                "C... C... S...",
                "KH","K...",
                "K...s H...d","kbh"
            ]
          }
        ]
      }
    ]
  }
}
```

2.41.3. Error codes

Error code	Description
1109	No zone data found with the given parameters
1604	No tariff data set for the date
3514	Invalid request parameters for tariff request
9901	Internal server error

Error structure and the list of all tariff-related error codes can be found here Section 4.2

2.42. Get Routes (Trip search)

This service returns a list of routes, each based on one or more journeys. It requires origin and destination stops to be passed in. This service uses a TripSearch.

2.42.1. Request Parameters

- Use M: Mandatory
- Use O: Optional

Name	Use	Range	Default	Est.	Description
accessId	M	-	-	1.0	Access ID for identifying the requesting client.
context	O	-	-	2.35	Optional context that can be passed in and is returned in the response.
mode	O	default, DSB, DSB-1, DSB-2, DSB-3	default	2.35	Special mode for different users.
travel_on	O	-	-	2.35	Date for the interval-search or "now" (to select current date). Default: same as current server date.
issue_on	O	-	-	2.35	Date selecting the tariff data version that will be used or "now" (to select current date). Default: same as "travel_on". This allows specifying if the tariff data valid on the travel date or the sale date has to be used (if the travel date is not today). Supports extxml.
time	O	-	0000	2.35	Start time for the interval search (format: HHMM) or "now" to select the current server time.
max_interval	O	-	1440	2.35	The time interval for the interval-search in minutes. The time interval can be decreased to improve performance.
oStop	M	-	-	2.35	Origin Stop Number.
dStop	M	-	-	2.35	Destination Stop Number (can be omitted if mode=DSB, then a result with the zone(s) for the origin stop is returned).

vStop	O	-	-	2.35	Via Stop Number (currently not supported).
psgs	O	-	["A"]	2.35	Array of passenger types. "A": Adult, "C": Child, "B": Bike, "P": Pensioner, "Y": Young, "H": Handicapped, "D": Dog Example: ["A","C"]

2.42.2. Example

Request: Get a list of routes for given origin and destination stops

```
<baseurl>/stRoutes?format=json&context=123456&mode=DSB&travel_on=20171231&time=1000&i
nterval=1440&oStop=8600626&dStop=8600617&psgs=["A"]
```

Result: (abbreviated)

```
{
    "stRoutes": {
        "res": {
            "td": 20230224,
            "sd": 20230224,
            "searchTime": 0,
            "searchInterval": 1440,
            "origin": {
                "number": 8600266,
                "name": "Silkeborg St.",
                "zones": [4330]
            },
            "dest": {
                "number": 8600053,
                "name": "Aarhus H",
                "zones": [4301]
            },
            "routes": [{
                    "classes": ["S"],
                    "product": {
                        "trfType": "LD",
                        "valType": "LD",
                        "prcDistance": 9,
                        "prcZones": [4330, 4301],
                        "viaZones": [4857],
                        "fares": [
                          {"psg": "A","price": 10200}
                        ]
                    },
                    "jnys": [{
                            "sections": [{
                                    "fa": "TV_TRAIN",
                                    "trans": "T",
                                    "cat": {"id": "030","name": "RA      "
,"productClass": "2"},
                                    "zones": [4330, 4331, 4338, 4857, 4318,
4340, 4342, 4307, 4303, 4301],
                                    "stops": [8600266]
                                }
                            ]
                        }
                    ]
                }, {
                    "classes": ["S"],
                    "product": {
                        "trfType": "RZ",
                        "valType": "RZ",
                        "prcDistance": 9,
                        "maxDuration": 145,
                        "prcZones": [4330, 4301],
                        "viaZones": [4331],
                        "fares": [
                          {"psg": "A","price": 9200}
                        ]
                    },
                    "jnys": [{
                            "sections": [{
                                    "fa": "MT",
                                    "trans": "B",
                                    "cat": {"id": "013","name": "Bus     "
,"productClass": "5"
                                    },
                                    "zones": [4330, 4331],
                                    "stops": [743700501, 751000102]
                                }
                            ]
                        }
                    ]
                }
            ]
        }
    }
}
```

2.42.3. Error codes

Error code	Description
1101	No zone found for stop
1604	No tariff data set for the date
1702	No fare set found for journey zones
1803	No via data found with the given parameters
2003	No trains on journeys (only relevant for DSB mode)
2004	No relational single ticket for fare set
2006	Missing zones on journey
2007	No fares for journeys
2010	No single ticket fares defined in the fare set
3509	No tariff data for the sale date
3514	Invalid request parameters for tariff request
3525	No single ticket fares valid in fare set
9901	Internal server error
9902	No journey plan data for the date
9903	No journeys found
9999	Undefined error

Error structure and the list of all tariff-related error codes can be found here Section 4.2

2.43. Get Routes (Tariff search)

This service returns a list of routes, each based on one or more journeys. It requires origin stop number to be passed in. This service uses a TariffSearch "stRoutes" (old genericFareInfoRequest).

2.43.1. Request Parameters

- Use M: Mandatory
- Use O: Optional

Name	Use	Range	Default	Est.	Description
accessId	M	-	-	1.0	Access ID for identifying the requesting client.
context	O	-	-	2.35	Optional context that can be passed in and is returned in the response.

mode	O	default, DSB, DSB-1, DSB-2, DSB-3	default	2.35	Special mode for different users.
travel_on	O	-	-	2.35	Date for the interval-search or "now" (to select current date). Default: same as current server date.
issue_on	O	-	-	2.35	Date selecting the tariff data version that will be used or "now" (to select current date). Default: same as "travel_on". This allows specifying if the tariff data valid on the travel date or the sale date has to be used (if the travel date is not today). Supports extxml.
time	O	-	0000	2.35	Start time for the interval search (format: HHMM) or "now" to select the current server time.
max_interval	O	-	1440	2.35	The time interval for the interval-search in minutes. The time interval can be decreased to improve performance.
oStop	M	-	-	2.35	Origin Stop Number.
psgs	O	-	["A"]	2.35	Array of passenger types. "A": Adult, "C": Child, "B": Bike, "P": Pensioner, "Y": Young, "H": Handicapped, "D": Dog Example: ["A","C"]

2.43.2. Example

Request: Get a list of routes for given origin stop number

`<baseurl>/stRoutesAddon?mode=DSB&oStop=8600626&psgs=["A","C"]&format=json`

Result: (abbreviated)

```
{
     "stRoutes": {
        "res": {
            "td": 20230419,
            "sd": 20230419,
            "origin": {
                "number": 12,
                "name": "Gladsaxevej (Gladsaxe Ringvej)",
                "zones": [
                    1031
                ]
            },
            "dest": {
                "number": 12,
                "name": "Gladsaxevej (Gladsaxe Ringvej)",
                "zones": [
                    1031
                ]
            },
            "routes": [
                {
                    "classes": [
                        "S",
                        "B"
                    ],
                    "product": {
                        "trfType": "RZ",
                        "valType": "RZ",
                        "prcDistance": 1,
                        "prcZones": [
                            1031,
                            1031
                        ],
                        "viaZones": [],
                        "maxDuration": 75,
                        "fares": [
                            {
                                "psg": "A",
                                "price": 1200
                            }
                        ]
                    },
                    "products": [],
                    "jnys": []
                }
            ]
        }
    }
}
```

2.43.3. Error codes

Error code	Description
1101	No zone found for stop
1604	No tariff data set for the date
1702	No fare set found for journey zones
1803	No via data found with the given parameters
2003	No trains on journeys (only relevant for DSB mode)
2004	No relational single ticket for fare set
2006	Missing zones on journey
2007	No fares for journeys
2010	No single ticket fares defined in fare set
3509	No tariff data for the sale date
3514	Invalid request parameters for tariff request
3525	No single ticket fares valid in fare set
9901	Internal server error
9902	No journey plan data for the date
9903	No journeys found
9999	Undefined error

Error structure and the list of all tariff-related error codes can be found here Section 4.2

2.44. 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.

2.44.1. Example

Request: List all available XSD files

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

Response:

- `rest-2.45-Rejseplanen.xsd`

Request: Return XSD

```
<baseurl>/xsd/rest-2.45-Rejseplanen.xsd
```

3. Responses

All services return their responses either in XML or JSON format (see Section 1.2.12). All possible response elements are defined in `rest-2.45-Rejseplanen.xsd` which could be retrieved via `<baseURL>/xsd?rest-2.45-Rejseplanen.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 response codes and messages

If the request encounters a failure or yields no results, the response will include an HTTP status code, error code and a corresponding textual description (error text), as outlined in the table below. Errors are categorized into distinct groups:

- Client errors - related to user input or authorization;
- Server errors - associated with internal problems within the backend infrastructure
- Service-specific errors - exclusive to particular service contexts.

Important: Starting from API Server version 2.42, there have been modifications to the returned HTTP status codes. To enable legacy mode and revert to the previous status codes, set the *httpStatusMode* property in the configuration file to *legacy*.

If the service hits a timeout, HTTP status code 503 is send.

4.1. ReST Request Errors

Code	HTTP status code (default)	HTTP status code (legacy)	Description
Client errors			
API_AUTH	403	403	Access denied for 'key' on 'service'
API_QUOTA	400	400	Quota exceeded for 'key' on 'service'
API_TOO_MANY_REQUESTS	403	403	Too many requests
API_PARAM	400	400	Required parameter <<name>> is missing
API_PARAM	400	400	numB wrong, only number in range [0,6] allowed
API_PARAM	400	400	numF wrong, only number in range [0,6] allowed
API_PARAM	400	400	numF + numB not greater than [6] allowed
API_FORMAT	400	400	Response format not supported
SVC_PARAM	400	400	Request parameter missing or invalid
SVC_LOC	400	400	Location missing or invalid / Nearby to the given address stations could not be found
SVC_LOC_ARR	400	400	Arrival location missing or invalid
SVC_LOC_DEP	400	400	Departure location missing or invalid
SVC_LOC_VIA	400	400	Unknown change stop
SVC_LOC_EQUAL	400	400	Start/destination or vias are equal
SVC_LOC_NEAR	400	400	Start and destination too close
SVC_DATATIME	400	400	Date/time missing or invalid

Code	HTTP status code (default)	HTTP status code (legacy)	Description
SVC_DATATIME_PERIOD	400	400	Date/time not in timetable or allowed period
SVC_PROD	400	400	Product field missing or invalid
SVC_CTX	400	400	Context invalid
SVC_MAIL_ADR	400	400	Sender/receiver mail address invalid or missing
SVC_SMS_NUM	400	400	Receiver sms phone number invalid or missing
SVC_NO_RESULT	400	200	No result found
Server errors			
BAIM_ERROR	500	500	Configuration or routing error in BAIM supporting setup
SVC_MAIL	500	500	Failed to send mail
SVC_SMS	500	500	Failed to send sms
SVC_FAILED_SEARCH	500	500	Unsuccessful search
SVC_NO_MATCH	422	422	No match found
SVC_TOO_MANY	400	422	Too many result items
INT_ERR	500	500	Internal error
INT_HAFAS_CONNECTION_ERROR	503	503	Connection to [HOST]:[PORT] closed by HAFAS server. (ConnectionException)
INT_HAFAS_CONNECTION_ERROR	503	503	Connection to [HOST]:[PORT] is not possible: connection refused: [HOST]:[PORT] (ConnectionException)
INT_HAFAS_CONNECTION_ERROR	503	503	Connection to [HOST]:[PORT] is not possible: connection timed out: [HOST]:[PORT] (ConnectionException) (hit connectTimeout of HAFAS Server)
INT_HAFAS_CONNECTION_OPENING_ERROR	503	503	Error message from HAFAS server [HOST]:[PORT]: error=1&[CONNECTION_STRING] (In most cases, this error occurs if the queue in the underlying HAFAS Server is full)
INT_TIMEOUT	503	503	Timeout during service processing (hit timeout at HAFAS.api)
INT_TIMEOUT	503	503	Timeout during backend communication (hit timeout for all retries at backend)
Service specific errors			
SOT_AT_DEST	400	400	Trip already arrived
SOT_BEFORE_START	400	400	Trip not started
SOT_CANCELLED	400	400	Trip cancelled

Code	HTTP status code (default)	HTTP status code (legacy)	Description
SOT_ALL_TRAINS_FILTERED	400	400	All trips filtered
SOT_STAY_IN_CURRENT_CONNECTION	400	200	No change. Stay on trip
PARTIALSEARCH_INCORRECT_PARAM	400	400	An invalid parameter combination was requested, i.e. the defined range of stable segments encompassed all public transport sections or it was attempted to search forward/backward from the end/beginning of the connection.
SOT_TRANSIT_LOCATION_MISSING_FOR_JOURNEY	400	400	Requested transit location is not part of the journey

4.2. Tariff Errors

Tariff error codes are send in a tariff-specific error structure as a part of common API-Server response structure:

```
<spPrice c="123456" serverVersion="2.xx" dialectVersion="2.xx" requestId=
"xxx">
    <TechnicalMessages>
    </TechnicalMessages>
    <err no="1106" msg="no data for fare set"/>
</spRefund>
```

List of all possible error codes, passed in err **no** attribute field:

Code	HTTP status code	Description
1	200	HAFAS error (i.e., no journey can be found for the given addresses at the specified date for some reason, this error can also mean that the server is not available) Unexpected error
2	200	Missing or invalid parameters
3	200	No zones specified
4	200	No tariff data
5	200	No data for the fare set
1001	200	Mandatory request parameters missing for 24h-search
1100	200	No zone found for coordinates

Code	HTTP status code	Description
1101	200	No zone found for the stop
1104	200	Missing zone group
1105	200	No zone group for the zone No stop for the zone
1106	200	No season pass data for the fare set
1109	200	No zone data found with the given parameters
1500	200	No specific season pass for the fare set/date/passenger type/zone distance
1503	200	No single tickets/season passes in fare set
1604	200	No tariff data No tariff data for the date
1701	200	Inconsistent tariff data (no data for the fare set)
1702	200	No fare set found for journey zones
1803	200	All journeys use too many zones for the zonal season pass No via data found with the given parameters
1806	200	It was not possible to group the zones as required Number of zones passed by the journey is below the minimum.
2003	200	No trains on journeys (only relevant for DSB mode)
2004	200	No relational single ticket for fare set
2006	200	Missing zones on journey
2007	200	No fares for journeys
2010	200	No season pass available in the area No single ticket fares defined in the fare set
3509	200	No tariff data for the sale date
3514	200	Invalid request parameters for the tariff request
3519	200	All products removed by filter (e.g., because of the frequency filter)
3525	200	No single ticket fares valid in fare set
9900	200	Missing or invalid parameters
9901	200	Internal server error
9902	200	No journey plan data for the date
9903	200	Internal error (unsupported server)
9999	200	Undefined error

5. Warning messages

In some cases, a service may return data along with some warnings. Those are packed into `Warnings` element.

5.1. Warnings

Code	Description
TOO_MANY	The result contains too many elements. Some elements are discarded.
TOO_MANY_SCROLL_CTX	The result contains too many scroll context elements. Some elements are discarded.
H887	Timeout exceeded. The result may be incomplete.
H889	Too many itineraries found. Some elements are discarded.

6. 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.

Date	Version	Author	Remarks
01.09.2017	1.23.14	mfr	• Remove parameter tripId from Journey match and Train search. • Refine description of Journey match and Train search. • Refine description of Station board services. • Add economic and groupFilter parameter to Trip service. • Add detailed description of result values for barrier free information and capacity utilization.
10.10.2017	1.23.15	mfr	• Add description of coordinate request parameters to location.name service. • Add type parameter description to location.nearbystops service. • Add showPassingPoints description to JourneyMatch service.
22.12.2017	1.23.16	mfr	Add options to HIM search service.
15.02.2018	1.23.17	mfr	Changed documented time pattern to hh:mm[:ss].
27.03.2018	1.23.18	mfr	• Add Search on trip service description • Add blockingList, includeEarlier description to trip search service. • Extend time table info service result sample.
08.05.2018	1.23.19	mfr	• Add operators filter to Journey Match and Train Search service. • Add viaId parameter to Search on trip service. • Add withICTAlternatives to Trip service.
14.05.2018	1.23.20	mfr	Add ivOnly, totalWalk, totalBike, totalCar, totalTaxi to Trip service.
31.05.2018	1.23.21	mfr	Changed range of changeTimePercent in Trip service.
11.06.2018	1.23.22	mfr	Add rounding method to changeTimePercent.
22.06.2018	1.23.23	kha	• Add Journey Validation service. • Correct codes of 1.2.12 Capacity information.
11.07.2018	1.23.24	kha	• Add RSS feed service. • Removed RSS response type from HIM search.

Date	Version	Author	Remarks
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
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

Date	Version	Author	Remarks
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
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

Date	Version	Author	Remarks
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
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

Date	Version	Author	Remarks
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
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

Date	Version	Author	Remarks
09.11.2020	2.13.0	mfr	• Add internalName, defName, addName to ProductType • Add MCP properties to notes • Add new service Tariff on transport association (tariff.transportassociation) • Add new service Reconstruction Converter (reconstructionConvert) • Add mechanism to overwrite HAFAS AID using provisioning • Add mechanism to Consul self registration • Add OriginDestType::trainCompositionMarker • Add Name::addName • Add attlist.Journey::addName from prodCtx::addName • Add Leg::addName • Add JourneyDetail::dayOfOperation • Add Leg::journeyDetail (sets @ref and @dayOfOperation only) • Add sotContext to any TripList result • Add productAtStop to Arrival and Departure • Add validity days to Messages • Extend HIM Message XSD documentation and map iconRes • Change log level for domain error codes to info.
07.12.2020	2.14.0	mfr	• Add hasAlt to TripType indicating if a trip has an alternative connection. • Add HIM Search result mapping for edge and region. See MessageEdgeType and MessageRegionType in XSD. • Add HCI 1.44 support. • Add new ecoCmp(0..*) to Trip • Add enableRtFullSearch to reconMatch service. • Add parameter viaGis to trip and interval search. • Fix documentation on minChangeTime and maxChangeTime.

Date	Version	Author	Remarks
12.02.2021	2.15.0	mfr	• Add attributes to via param doc. • Add overall rate limiter to configuration. • Add poly mapping to HIM Search. • Add Notes mapping to Leg of type=TRSF. • Add ConnectionStatus mapping → TripStatus at Trip level. • Moved Leg cardinality from sequence to Leg itself in XSD. • Return planrtts as formatted date-time. • Swagger: Change type int to integer. • Swagger: Fix return type schema. • Maps SOT_ALL_TRAINS_FILTERED and SOT_STAY_IN_CURRENT_CONNECTION return codes. • Remove rtMode parameter from any board service • Add dailyStartingAt, dailyDuration to Messages • Add enableRtFullSearch to recon service. • Add type parameter to Address Lookup service • Removed maxNo parameter from Address Lookup service • Extend partialtripsearch.v2 with beginLocation@date and endLocation@date to respect over midnight journey • Add new error PARTIALSEARCH_INCORRECT_PARAM to partialtripsearch.v2: An invalid parameter combination was requested, i.e. the defined range of stable segments encompassed all public transport sections or it was attempted to search forward/backward from the end/beginning of the connection. • Supports latest HCI version 1.45 • Add attribute Message@baseType • JourneyDetail INT_ERR maps to SVC_NO_RESULT with HTTP status 400. • Add externalContent of type ExternalContentType to TariffResult, fareItem and ticket • Add new parameter travellerProfileData to any routing service Attributes alighting and boarding remain optional but default to true

Date	Version	Author	Remarks
25.03.2021	2.16.0	mfr	• Supports HCI version 1.46 • Add new services GeoFeature nearby and in bounding box. • Add bikeCarriageType to any relevant routing service. Functionality depends on data. • Add region element to Data Information service response structure • Add provisionable region parameter to HIM Search service • Add provisionable himtexttags parameter to HIM Search service
03.05.2021	2.17.0	mfr	• Removed allowDisabledStops from any reconstruction service • Supports HCI version 1.48 • Add new flag trackHidden parallel to any track information • Correct description of Interval search service parameter `max`. • Changed allowed values for bikeCarriageType to SINGLEBIKES, SMALLGROUPS und LARGEGROUPS. • Add new structure ParallelJourneyRefType to be used in Departure and JourneyDetail
03.06.2021	2.18.0	mfr	• Supports HCI version 1.49 • Adds withFreq parameter to trip and interval search. • Extend interval search: Add parameter "combine". Add response attributes combinedCount and combinedMinDuration. • Add durS to GisRouteSegment • Check for duplicates during combination of message lists in hierarchical view of HIM Search response

Date	Version	Author	Remarks
24.06.2021	2.19.0	mfr	• Add support for tracing headers. See Installation and Administration Manual. • Extend mapping of Origin and Destination notes. • Add mapping of location type MCP at location.details service. • `/systeminfo` will check backends if requested with `type=full` only • Add `arrHide`/`depHide` at stops as well as `hide` at `Origin`/`Destination` • Enable direct SSL support. See Installation and Administration Manual. • Add Add gisProduct filter option to Trip Search, Interval Search, Partial Trip Search v2, Many to many • Add Real Time Archive service v2 (using ReST interface instead of SOAP) • Remove support for Real Time Archive service v1
06.09.2021	2.20.0	mfr	• Supports HCI version 1.52 • Set fromIdx and toIdx at Message elements if available • Add map information to data info service response. See MapInfoType structure in XSD. • Map direction flag at Leg • Map direction information at JourneyDetail • Extend direction information at arrival/departure • Map new error code H_9381 to SVC_LOC_EQUAL • Handle H_883 • Set fromIdx and toIdx at Message elements if available • Add new CalculationType RETRY_UNSHARP_NEW_RADIUS • Restrict length of jid to 512 • Map TETA ConSection to Leg like JNY

Date	Version	Author	Remarks
01.10.2021	2.21.0	mfr	• Supports HCI version 1.53 • Add routeIdx to validFromStop and validToStop to HIM Message • Fix HIM Message mapping • Add partial search contexts to OriginDestType • Extend EcoType • Add new note types FN and TLN • Add enableReplacements to recon services • Add via list to TripType • Extend RSS Feed service with predefined filters • Add HAFAS error codes to error message
03.11.2021	2.22.0	mfr	• Supports HCI version 1.53 & 1.54 • Add new service Location Preselection • Add new type entrypoint (E) to Location Search by Coordinate (see Section 2.4) and Location Search in a bounding box (see Section 2.5) • Add `description` attribute to `StopLocation` and `CoordLocation` which can carry additional description of location, e.g. address. • Add `rating` attribute to `EcoType`. • Add request count by response type for Manager usage. • Add option `blockedEdges` to trip and interval search. • Add option `addionalfields` to him search. • Library upgrade.

Date	Version	Author	Remarks
25.11.2021	2.23.0	mfr	• Adds uncertainDelay at leg, journey, origin, dest, stops. • Service journeyPos gets configurable maxJny and maxJnyDefault. • Setup supports automated BAIM redirect. • Removed deprecated filterEquiv parameter from station board services. Use `type=DEP_EQUIVS` instead. • Adds child locations to coord locations. • api-doc support for duration datatype.

Date	Version	Author	Remarks
31.01.2022	2.24.0	mfr	• Unmapped query params will be returned as comma separated list in a technical message with key unmappedQueryParams if any. • Add withJourneyBoundaryPoints parameter to trip search, interval trip search, reconstruction, reconstruction match, partial trip search and trip alternatives services. Add JourneyOrigin and JourneyDestination to Leg structure. • Add isTurningPoint to any journey stop. • Add bounding box parameters to HIM Search service. • Add Messages to GisRouteSegment. • Add new TrafficMessageType structure at GisRouteSegment. • Add param list to TariffResult structure. • Removed mapping of traffic messages to Message structure at Leg level, added use of TrafficMessageType instead. • Migrate to Apache Camel 3. • Extends JourneyPos service: Adds periodSize and periodStep service parameter; Extends JourneyDetail with JourneyPath structure. • Add `Warnings` structure. • Add length restriction to any string based input parameter. • Add infotext filter to Location Search by Name service. • Supports HCI version 1.55 & 1.56
25.02.2022	2.25.0	mfr	• Add new service Tariff Validation • Add `categories` filter option to any journey related service like Trip, Interval, Search on trip, Station board, … • Extend response structure of StopLocation with list of equivalentStopLocation and entryPointLocation which will be filled if and only in case of Location Details service.

Date	Version	Author	Remarks
30.03.2022	2.26.0	mfr	• Removed required flags from most HIM Message attributes to fit source contract • Add textInternal to HIM Message response structure • Extend main mast information with coordinates • Add new parameter allowFootpathEquivalences to trip and interval search • Add main mast information to Arrival and Departure types • Add parameters dateB, dateE, timeB, timeE to Train Search service for period search • Add baim parameter to Station Board services. • Add poolId parameter to Section 2.3, Section 2.4 and Section 2.5. • Add poolId and comment to response of Section 2.30. • Add lon and lat to Arrival and Departure structure. • Add altitude attribute (alt or mainMastAlt) to any location structure. • Remove negative filter for line and jid from Journey Position service. Only positive filtering is allowed. • Map new error code H_9220_R to SVC_LOC.

Date	Version	Author	Remarks
29.04.2022	2.27.0	mfr	• Add JourneyOrigin and JourneyDestination to JourneyType structure. In trip search service, it is part of ParallelJourney list and in HIM Messages, it is type of AffectedJourneys. • Replaced INT_GATEWAY error code by more specific INT_HAFAS_CONNECTION_ERROR and INT_HAFAS_CONNECTION_OPENING_ERROR. Check chapter Section 4.1 for more details. • Supports HCI version 1.58 • Add support for wildcard provisioning at `meta` on location.name, location.nearbystops and location.boundingbox services • Extend interval search: Add parameter "combineProducts". Add response element CombinedProduct to Leg structure. • Add support for extId on Location Details service. • Add `description` attribute to OriginDestType and StopType.
31.05.2022	2.28.0	mfr	• Supports HCI version 1.60 • Search on trip service: Removed lastStopId, lastStopDate, lastStopTime because not used. Added transitionLocId, transitionLocType, transitionLocDate, transitionLocTime. • Location Search by Name, Location Search in a bounding box, Location Search by Coordinate: Default for `type` is configurable now. • Location Search by Name: add new parameters `withProducts` and `productRepresentatives` • Station Board: add new parameters `passlistMaxStops` and `minDur`
12.07.2022	2.29.0	mfr	• Location Search by Name: parameter `productRepresentatives` has no explicit default now. If not present, HAFAS default is used. • Fix minor mapping issue on GeoFeature service.

Date	Version	Author	Remarks
31.08.2022	2.30.0	mfr	• Trip search: Add originCoordType and destCoordType to explicitly set type if coordinate cannot be resolved to an address or poi. • Location name / nearby / bounding box search: Allow multiple meta filters per request if defined for POI filtering. • Introduce platform structure as future replacement of track attributes at stop locations. • Data information service: Add MobilityServiceProviderInfo structure. • Supports HCI version 1.63
30.09.2022	2.31.0	mfr	• Location service: Add url attribute to LocationNote. Was filled to value, now as own attribute. • Location name service: Add restriction filter `restrictSelection` to select stations allowed for start, destination or via only. Explicit ignoring any restriction is also possible. • Introduce new field `operatorInfo` of type `OperatorType` at `ProductType`. Existing fields `operator` and `operatorCode` marked as deprecated. • Trip services: Add `id` field to `Leg` which can be referred e.g. by tariff information. • Location Details: Add weather information structures and new parameter `weather`. • Add weather information structures to origin + destination at trip services. • Add new RealtimeDataSourceType values: ARAMIS, RTABO2

Date	Version	Author	Remarks
28.10.2022	2.32.0	mfr	• Fix generation of api-doc. some elements were missing. • Remove GraphQL support. • Remove Consul support. • Add new GeoFeatureTypes BICYCLE_PATH and STATION_AREA. • Add `fromLegId`, `toLegId` to tariff/ticket structures referring to `id` field of certain `Leg`. • Supports HCI version 1.65
05.01.2023	2.33.0	mfr	• Supports HCI version 1.66 • Add `gisProducts` parameter to Interval Search. • Add `isMainMast` to StopLocation, Arrival, Departure, Stop and OriginDestType. • Add `meta` parameter to GeoFeature nearby and in bounding box services. • Add `calBurned` attribute to TripType response structure. • System information returns tariff information of loaded tariff data sets. • Add `withEquivalentLocations` parameter to Location Search by Name. Return equivalent locations, default is `false`. • Remove `fareSetItem::ticketParam` as it is not used anymore.

Date	Version	Author	Remarks
09.02.2023	2.34.0	mfr	• Supports HCI version 1.67 • Add `rtMode` parameter to board services. Only `SERVER_DEFAULT` and `OFF` allowed if any. • Add `SortingType` to each trip search service to allow client side sorting by various types, pre-sorted at HAFAS. • Add new field `TripType.hasDelayInfo`. If this field is set to true, delay prognoses are available for this connection. • Add `mainMast` element to StopLocation, Arrival, Departure, StopType and OriginDestType. Mark hasMainMast, mainMastId, mainMastExtId, mainMastLon, mainMastLat, mainMastAlt, mainMastAltId deprecated. • Add `outputControl` to `ReconstructionMatchRequest` type.
08.05.2023	2.35.0	mfr	• Fix mapping error on external content of tariff structure. • New services Section 2.16 and Section 2.17. • New value `PE` for `type` parameter of Section 2.4 and Section 2.5: Search for POIs and entrypoints. • New parameter `withMastNames` for service Section 2.3: Add masts to found stations.
01.06.2023	2.36.0	mfr	• Fix mapping error of `zoomMax` in MapLayerType type. • New services: Section 2.31, Section 2.32, Section 2.33, Section 2.34, Section 2.35, Section 2.36, Section 2.37, Section 2.38, Section 2.39, Section 2.40, Section 2.41, Section 2.42, Section 2.43 • Enhanced mapper for TripSearch and ReconMatch tariff requests

Date	Version	Author	Remarks
19.07.2023	2.37.0	mfr	• Add nearby station board services Section 2.18 and Section 2.19. • Add `tariffSystemList` parameter to [service_tariff-transportassociation]. • Remove default value of 0 from `price` attribute in `farItem` and `ticket`. • Add missing location mappings for types C, HL and LM. • Supports HCI version 1.72
30.08.2023	2.38.0	mfr	• Update default values for SP Price and SP Refund requests • Resolve encoding issues • Update api-docs with duration type • Supports HCI version 1.73 • Add external ID support to HIM Message service • Extend CommonResponseType with error related fields `internalErrorCode`, `internalErrorText` and `internalErrorTextOut`. • Add new result field `changeRequired` at Leg element • Introduce new result fields to Trip Alternatives (trip.alternatives) / Search on Trip by context (sot-ctx) services. See `SotContextType` definition for details.

Date	Version	Author	Remarks
27.09.2023	2.39.0	mfr	• Add Traveller Profile, date and GIS meta filter support in `Partial Trip Search V2` service • Add planningPeriodBegin and planningPeriodEnd mappings in JourneyPos service • Add directionFlag JourneyType which is used in LineType • Add new attribute waiting to Leg type • Add baim parameter for reconstruction and gisroute service • Add zoom, date and time parameter to location.nearbystops + location.boundingbox services • Add zoom parameter to location.details service • Supports HCI version 1.74 • Namespace of rest.xml changed to `xmlns:jaxb="https://jakarta.ee/xml/ns/jaxb"`
15.11.2023	2.40.0	mfr	• Supports searching for multiple locations in location.details service • Add date and gis meta filter elements to PartialSearch request • Add planningPeriodBegin and planningPeriodEnd mapping in Journey Position (journeyPos) • Documentation improvements • Extend (HIM) Message structure with descriptive text for affected section. • Supports HCI version 1.76 • Add SOT_TRANSIT_LOCATION_MISSING_FOR_JOURNEY error message to Search on Trip (sot) • Improve validation rules for input parameters

Date	Version	Author	Remarks
16.01.2024	2.41.0	mfr	• Add new service Notes Localization ([service_l10n-notes]) • Add new GisRouteManoeuvre RR • Extend `StopLocation`, `StopType` and `OriginDestType` with new attribute `minimumChangeDuration`: Minimum duration to change at this stop/station. • Extend `Leg` with new attribute `minimumChangeDuration`: Minimum duration to change to the next leg. • Extend `JourneyType`, `JourneyDetail`, `ParallelJourneyRefType` with attribute `parallelJourneyLinkType` of type `` `ParallelJourneyLinkTypeType `` • Change `EcoType` attribute `type` to enum `EcoTypeType` instead of string. See XSD for details. • Update Swagger UI to version 4.19.1

Date	Version	Author	Remarks
22.04.2024	2.42.0	jbi	• Supports HCI version 1.80 • Update Swagger UI to version 5.12.0 • Added new IconShapeTypes: B, H • Added new RealtimeDataSourceTypes: ACCESS_RESTRICTION, GENERIC, REROUTING • Added new filter in `HIM Search` service: exthimtext • Added `filterFootpaths` parameter to `Reconstruction Match` service request • Added `includeDrt` parameter to `Trip Search` service request • Introduce new attribute `waitingState` in `Leg`. Attribute `waiting` marked as deprecated. • Align request parameters of Trip Search service with Search on Trip and Trip Alternatives services • Request parameters can now be sent simultaneously in the query and as x-www-form-urlencoded body. • Modifications of the returned HTTP status codes. See details: Section 4.1 • Added poiCategory filter to `location.search`, `location.nearbystops` and `location.boundingbox` services • Added support for additional fields in TravellerProfile in TripSearch (dl, pt, pn)

Date	Version	Author	Remarks
15.08.2024	2.45.0	jbi	• Supports HCI version up to 1.84. • Extended support for Global IDs / Alternative ID spaces. See details: Section 1.2.10. • Extended `Leg` and `JourneyType` with `ParallelJourneyRef` elements • Update Swagger UI to version 5.17.2. • Added `rtActivated` and `rtDeactivated` attributes to Notes and LocationNotes • Fix validation of `maxJourneys` parameter in multiBoard and nearbyBoard services • `CoordLocation` extended by entryLocation list. • Removed withMastNames parameter from `location.name` service • Add new service Trias 1.2 ([service_trias_v1-2]) • Changed allowed max length of requestId parameter to 2048 • Add direction mapping to him event → MessageEventType::direction • HIM Search improvement • Reconstruction Match Service: Extract reference date from either departureTime or departureTimeRt • Reconstruction Match Service: Validate for trainName or (@trainNumber and @trainCategory) to be set before mapping • Request parameter `includeDrt` changed default behavior: defaults to `true`.
30.08.2024	2.45.1	jbi	• Fixed validation of reconMatch requests
04.09.2024	2.45.2	jbi	• Fixed issue causing products to be duplicated in Leg structure