HaCon

Traffic • Software • Service

HAFAS ReST API

Access to HAFAS journey planner systems

Version 1.21

20th April 2015

Michael Frankfurter, Michael Schuschk
HaCon Ingenieurgesellschaft mbH
Lister Straße 15
30163 Hannover
Germany

Contents

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

1 The Interface

1.1 Introduction

1.1.1 Interface Overview

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

- Location services
  - Location name
  - Nearby
  - Neraby POIs
- Trip
  - Scroll
- Interval trip search
- Reconstruction
- Station board services:
  - DepartureBoard
  - ArrivalBoard
- JourneyDetail
- XSD
- Status

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

The system implements read-only GET requests which are called by given service URLs and multiple GET parameters to specify the requested journey planner information. The parameter values need to be UTF-8 URL encoded. The result of each request will be delivered either

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

as XML or JSON response. If the URL parameter encoding is not correct, the behaviour of the system might deliver unexpected results.

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

1.2 General principles

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

1.2.1 Coordinates

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

1.2.2 Date and time formats

Dates are always represented in the format YYYY-MM-DD. This applies both for request parameters as for dates in responses. Times are always represented in the format HH:MM in 24h nomenclature.

1.2.3 Stateless service vs. data dependency

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

1.2.4 Route index

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

1.2.5 Real-time information

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

1.2.6 Versioning

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

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

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

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

1.2.7 Response Format

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

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

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

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

- Element names become object properties
- Text (PCDATA) becomes an object property with name "$"
  <a>foo becomes { "a": { "$" : "foo" } }
- Nested elements become nested properties
  <a>foo<c>foo</c>
  becomes
  { "a": { "b" : { "$": "foo" }, "c": { "$": "foo"} } }
- If there are multiple elements with the same name, the JSON code contains an array for these elements.
  <a>foo1foo2
  becomes
  { "a": { "b" : [{"$": foo1" }, {"$": "foo2" }] } }
- Attribute names become object properties
  foo2
  becomes
  { "a": { "atb" : "foo1", "$" : "foo2" } }

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

XML:

```
<Trip xmlns="hafas_rest_v1">
      <Leg name="Expressbuss 830" type="LOC" id="830"
            direction="Göteborg Nils Ericsonterminal">
```

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

JSON:

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

1.2.8 Authentication

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

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

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

1.2.9 Languages

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

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

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

Code	Language	Code	Language	Code	Language
de	German	fr	French	no	Norwegian
da	Danish	hu	Hungarian	pl	Polish

en	English	it	Italian	sv	Swedish
es	Spanish	nl	Dutch	tr	Turkish

2 Services

2.1 Location Service

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

The response format for all services is defined in `hafasRestLocation.xsd` (see also 2.12 for further details).

2.1.1 Stop weight

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

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

2.2 Location.name Service

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

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

2.2.1 Request Parameters

Name	Use	Range	Default	Description
accessId	Mandatory	-	-	Access ID for identifying the requesting client.
input	Mandatory	See 1.2.1	-	Search for that token
maxNo	Optional	1-1000	10	Maximum number of returned stops
type	Optional		ALL	Type filter for location types: ALL: search in all existing location pools S: Search for stations only A: Search for addresses only P: Search for POI's only SA: Search for stations and addresses SP: search for stations and POIs AP: search for addresses and pois
products	Optional	-	-	Decimal value defining the product classes to be included in the search. It represents a bitmask combining bit number of a product as defined in the HAFAS raw data file zugart. For example, regional trains are product class 2 and local trains are class 3, while busses are 4. If you would like to search for local and regional trains only, you would need a bitmask where bits 2 and 3 are set. Calculation is 2^2 + 2^3 = 12 which would be the parameter value for "products". When searching for busses only, "products" need to be set to 16 = 2^4.

2.2.2 Example

Request:

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

Result:

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

2.3 Location.nearbystops Service

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

2.3.1 Request Parameters

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

Name	Use	Range	Default	Description
products	Optional	-	-	Decimal value defining the product classes to be included in the search. It represents a bitmask combining bit number of a product as defined in the HAFAS raw data file zugart. For example, regional trains are product class 2 and local trains are class 3, while busses are 4. If you would like to search for local and regional trains only, you would need a bitmask where bits 2 and 3 are set. Calculation is 2^2 + 2^3 = 12 which would be the parameter value for "products". When searching for busses only, "products" need to be set to 16 = 2^4.

2.3.2 Example

Request: Search for stations around the coordinate

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

Result:

```
<LocationList xmlns="hafas_rest_v1">
    <StopLocation id="A=1@O=Oslo S@X=10755332@Y=59910200@u=0@U=70
        @L=7600100@" name="Oslo S" lon="10.755332" lat="59.9102"/>
    <StopLocation id="A=1@O=Nydalen st@X=10758982@Y=59915405@u=0
        @U=70@L=7621273@" name="Nydalen st" lon="10.758982"
        lat="59.915405"/>
</LocationList>
```

2.4 Trip service

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

2.4.1 Request Parameters

Name	Use	Range	Default	Description
accessId	Mandatory	-	-	Access ID for identifying the requesting client.
originId	Mandatory if originExtId and coordinate aren't specified	See 2.2 or 2.3	-	Specifies the station/stop ID (location reconstruction context) of the origin for the trip. Such ID can be retrieved from the location.name or location.nearbystops services.
originExtId	Mandatory if originId and coordinate aren't specified			Specifies the external station/stop ID of the origin for the trip. Such ID can be retrieved from the location.name or location.nearbystops services
originCoordLat	Mandatory if ID isn't specified	See 1.2.1 and 2.2 or 2.3	-	Latitude of station/stop coordinate of the trip's origin. The coordinate can be retrieved from the location.name or location.nearbystops services.
originCoordLong	Mandatory if ID isn't specified	See 1.2.1 and 2.2 or 2.3	-	Longitude of station/stop coordinate of the trip's origin. The coordinate can be retrieved from the location.name or location.nearbystops services.
destId	Mandatory if destExtId and coordinate aren't specified	See 2.2 or 2.3	-	Specifies the station/stop ID (location reconstruction context) of the destination for the trip. Such ID can be retrieved from the location.name or location.nearbystops services.
destExtId	Mandatory if destId and coordinate aren't specified			Specifies the external station/stop ID of the destination for the trip. Such ID can be retrieved from the location.name or location.nearbystops services

destCoordLat	Mandatory if ID isn't specified	See 1.2.1 and 2.2 or 2.3	-	Latitude of station/stop coordinate of the trip's destinationn. The coordinate can be retrieved from the location.name or location.nearbystops services.
destCoordLong	Mandatory if ID isn't specified	See 1.2.1 and 2.2 or 2.3	-	Longitude of station/stop coordinate of the trip's destination. The coordinate can be retrieved from the location.name or location.nearbystops services.
viaId	Optional	See 2.2 or 2.3		ID of a station/stop used as a via for the trip. Specifying a via station forces the trip search to look for trips which must pass through this station. Such IDs can be retrieved from the location.name or location.nearbystops services.
viaWaitTime	Optional	See 1.2.2	0	Defines the waiting time spent at via station in minutes.
changeTimePercent	Optional	100 - 500	100	Configures the walking speed when changing from one leg of the journey to the next one. It extends the time required for changes by a specified percentage. A value of 200 doubles the change time as initially calculated by the system.
maxChange	Optional	1-3		Max no of changes.
date	Optional	See 1.2.2	Current server date	Sets the departure date for the search.
time	Optional	See 1.2.2	Current server time	Sets the departure time for the search.
searchForArrival	Optional	0 or 1	0	If set, the date and time parameters specify the arrival time for the trip search instead of the departure time.

numF	Optional	0 to 6	4	Minimum number of trips after the search time. Sum of **numF** and **numB** has to be less or equal 6. Please see the section below about the search algorithm for more details.
numB	Optional	0 to 6	1	Minimum number of trips before the search time. Sum of **numF** and **numB** has to be less or equal 6. Please see the section below about the search algorithm for more details.
products	Optional	-	-	Decimal value defining the product classes to be included in the search. It represents a bitmask combining bit number of a product as defined in the HAFAS raw data file *zugart*. For example, regional trains are product class 2 and local trains are class 3, while busses are 4. If you would like to search for local and regional trains only, you would need a bitmask where bits 2 and 3 are set. Calculation is 2^2 + 2^3 = 12 which would be the parameter value for "products". When searching for busses only, "products" need to be set to 16 = 2^4.
context	Optional	See 2.4	-	Defines the starting point for the scroll back or forth operation. Use the scrB value from a previous result to scroll backwards in time and use the scrF value to scroll forth.
poly	Optional	0 or 1	0	Enables/disables the calculation of the polyline for each leg of the trip.
passlist	Optional	0 or 1	0	Enables/disables the return of the passlist for each leg of the trip.

operators	Optional	All operator codes from HAFAS raw data file *betrieb*.	-	Only trips provided by the given operators are part of the result. To filter multiple operators, separate the codes by comma. E.g. filter for A and B operator: operators=A,B.
avoidPaths	Optional	One or more codes	-	Only path not having the given properties will be part of the result. Possible codes are Stairway SW Elevator EA Escalator ES Ramp RA Convey Belt CB E.g. use paths without ramp and stairway: avoidPaths=SW,RA.
originWalk	Optional	0 or 1	1	Enables/disables using footpaths in the beginning of a trip when searching from an address. To fine-tune the minimum and/or maximum distance to the next public transport station, provide these values separted by comma. These values are expressed in meters. E.g. 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.

originBike	Optional	0 or 1	1	Enables/disables using bike routes in the beginning of a trip when searching from an address. To fine-tune the minimum and/or maximum distance to the next public transport station, provide these values separted by comma. These values are expressed in meters. E.g. 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.
originCar	Optional	0 or 1	1	Enables/disables using car in the beginning of a trip when searching from an address. To fine-tune the minimum and/or maximum distance to the next public transport station, provide these values separted by comma. These values are expressed in meters. E.g. To enable car, minimum distance should be 2000 meters, maximum distance should be 100 kilometers set the parameter originCar=1,0,100000. If the default distance should be used, just put no value, e.g 1,,100000 to have car enabled, default minimum and 100 kilometers as maximum.

originTaxi	Optional	0 or 1	1	Enables/disables using taxi rides in the beginning of a trip when searching from an address. To fine-tune the minimum and/or maximum distance to the next public transport station, provide these values separted by comma. These values are expressed in meters. E.g. 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.
destWalk	Optional	0 or 1	1	Enables/disables using footpaths at the end of a trip when searching to an address. To fine-tune the minimum and/or maximum distance from the next public transport station, provide these values separted by comma. These values are expressed in meters. E.g. 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.

destBike	Optional	0 or 1	1	Enables/disables using bike routes at the end of a trip when searching to an address. To fine-tune the minimum and/or maximum distance from the next public transport station, provide these values separted by comma. These values are expressed in meters. E.g. 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.
destCar	Optional	0 or 1	1	Enables/disables using car routes at the end of a trip when searching to an address. To fine-tune the minimum and/or maximum distance from the next public transport station, provide these values separted by comma. These values are expressed in meters. E.g. To enable car, minimum distance should be 2000 meters, maximum distance should be 100 kilometers set the parameter originCar=1,0,100000. If the default distance should be used, just put no value, e.g 1,,100000 to have car enabled, default minimum and 100 kilometers as maximum.

destTaxi	Optional	0 or 1	1	Enables/disables using taxi rides at the end of a trip when searching to an address. To fine-tune the minimum and/or maximum distance from the next public transport station, provide these values separted by comma. These values are expressed in meters. E.g. 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.

2.4.2 Search algorithm

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

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

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

2.4.3 Scrolling

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

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

2.4.4 Response

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

2.5 Interval trip search service

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

2.5.1 Request Parameters

Name	Use	Range	Default	Description
accessId	Mandatory	-	-	Access ID for identifying the requesting client.
originId	Mandatory if originExtId and coordinate aren't specified	See 2.2 or 2.3	-	Specifies the station/stop ID (location reconstruction context) of the origin for the trip. Such ID can be retrieved from the location.name or location.nearbystops services.

originExtId	Mandatory if originId and coordinate aren't specified			Specifies the external station/stop ID of the origin for the trip. Such ID can be retrieved from the location.name or location.nearbystops services
originCoordLat	Mandatory if ID isn't specified	See 1.2.1 and 2.2 or 2.3	-	Latitude of station/stop coordinate of the trip's origin. The coordinate can be retrieved from the location.name or location.nearbystops services.
originCoordLong	Mandatory if ID isn't specified	See 1.2.1 and 2.2 or 2.3	-	Longitude of station/stop coordinate of the trip's origin. The coordinate can be retrieved from the location.name or location.nearbystops services.
destId	Mandatory if destExtId and coordinate aren't specified	See 2.2 or 2.3	-	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	Mandatory if destId and coordinate aren't specified			Specifies the external station/stop ID of the origin for the trip. Such ID can be retrieved from the location.name or location.nearbystops services
destCoordLat	Mandatory if ID isn't specified	See 1.2.1 and 2.2 or 2.3	-	Latitude of station/stop coordinate of the trip's destinationn. The coordinate can be retrieved from the location.name or location.nearbystops services.
destCoordLong	Mandatory if ID isn't specified	See 1.2.1 and 2.2 or 2.3	-	Longitude of station/stop coordinate of the trip's destination. The coordinate can be retrieved from the location.name or location.nearbystops services.

viaId	Optional	See 2.2 or 2.3		ID of a station/stop used as a via for the trip. Specifying a via station forces the trip search to look for trips which must pass through this station. Such IDs can be retrieved from the location.name or location.nearbystops services.
viaWaitTime	Optional	See 1.2.2	0	Defines the waiting time spent at via station in minutes.
changeTimePercent	Optional	100 - 500	100	Configures the walking speed when changing from one leg of the journey to the next one. It extends the time required for changes by a specified percentage. A value of 200 doubles the change time as initially calculated by the system.
maxChange	Optional	1-3		Max no of changes.
date	Mandatory	See 1.2.2	-	Sets the departure date for the search.
time	Mandatory	See 1.2.2	-	Sets the departure time for the trip search.
duration	Mandatory	1 to 1439	-	Set the interval size in minutes.
searchForArrival	Optional	0 or 1	0	If set, the date and time parameters specify the arrival time for the trip search instead of the departure time.
max	Optional	>0	-	Minimum number of trips after the search time.

products	Optional	-	-	Decimal value defining the product classes to be included in the search. It represents a bitmask combining bit number of a product as defined in the HAFAS raw data file *zugart*. For example, regional trains are product class 2 and local trains are class 3, while busses are 4. If you would like to search for local and regional trains only, you would need a bitmask where bits 2 and 3 are set. Calculation is 2^2 + 2^3 = 12 which would be the parameter value for "products". When searching for busses only, "products" need to be set to 16 = 2^4.
context	Optional	See 2.4	-	Defines the starting point for the scroll back or forth operation. Use the scrB value from a previous result to scroll backwards in time and use the scrF value to scroll forth.
poly	Optional	0 or 1	0	Enables/disables the calculation of the polyline for each leg of the trip.
passlist	Optional	0 or 1	0	Enables/disables the return of the passlist for each leg of the trip.
operators	Optional	All operator codes from HAFAS raw data file *betrieb*.	-	Only trips provided by the given operators are part of the result. To filter multiple operators, separate the codes by comma. E.g. filter for A and B operator: operators=A,B.

avoidPaths	Optional	One or more codes	-	Only path not having the given properties will be part of the result. Possible codes are Stairway SW Elevator EA Escalator ES Ramp RA Convey Belt CB E.g. use paths without ramp and stairway: avoidPaths=SW,RA.
originWalk	Optional	0 or 1	1	Enables/disables using footpaths in the beginning of a trip when searching from an address. To fine-tune the minimum and/or maximum distance to the next public transport station, provide these values separted by comma. These values are expressed in meters. E.g. 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.

originBike	Optional	0 or 1	1	Enables/disables using bike routes in the beginning of a trip when searching from an address. To fine-tune the minimum and/or maximum distance to the next public transport station, provide these values separted by comma. These values are expressed in meters. E.g. 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.
originCar	Optional	0 or 1	1	Enables/disables using car in the beginning of a trip when searching from an address. To fine-tune the minimum and/or maximum distance to the next public transport station, provide these values separted by comma. These values are expressed in meters. E.g. To enable car, minimum distance should be 2000 meters, maximum distance should be 100 kilometers set the parameter originCar=1,0,100000. If the default distance should be used, just put no value, e.g 1,,100000 to have car enabled, default minimum and 100 kilometers as maximum.

originTaxi	Optional	0 or 1	1	Enables/disables using taxi rides in the beginning of a trip when searching from an address. To fine-tune the minimum and/or maximum distance to the next public transport station, provide these values separted by comma. These values are expressed in meters. E.g. 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.
destWalk	Optional	0 or 1	1	Enables/disables using footpaths at the end of a trip when searching to an address. To fine-tune the minimum and/or maximum distance from the next public transport station, provide these values separted by comma. These values are expressed in meters. E.g. 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.

destBike	Optional	0 or 1	1	Enables/disables using bike routes at the end of a trip when searching to an address. To fine-tune the minimum and/or maximum distance from the next public transport station, provide these values separted by comma. These values are expressed in meters. E.g. 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.
destCar	Optional	0 or 1	1	Enables/disables using car routes at the end of a trip when searching to an address. To fine-tune the minimum and/or maximum distance from the next public transport station, provide these values separted by comma. These values are expressed in meters. E.g. To enable car, minimum distance should be 2000 meters, maximum distance should be 100 kilometers set the parameter originCar=1,0,100000. If the default distance should be used, just put no value, e.g 1,,100000 to have car enabled, default minimum and 100 kilometers as maximum.

destTaxi	Optional	0 or 1	1	Enables/disables using taxi rides at the end of a trip when searching to an address. To fine-tune the minimum and/or maximum distance from the next public transport station, provide these values separted by comma. These values are expressed in meters. E.g. 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.

2.5.2 Response

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

2.6 Reconstruction service

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

2.6.1 Request Parameters

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

poly	Optional	0 or 1	0	Enables/disables the calculation of the polyline for each leg of the trip.
date	Optional			This parameter will force the service to reconstruct the trip on that specific date. If the trip is not available on that date, because it does not operate, the error code H890 will be returned.

2.6.2 Example

Request:

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

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

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

2.7 Stationboard services

The station board can be retrieved by a call to the `departureBoard` or `arrivalBoard` services. This method will return the next 20 departures (or less if not existing) or arrivals from a given point in time.

2.7.1 Request Parameters

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

id	Mandatory if extId is not specified	See 2.2 or 2.3	-	Specifies the station/stop ID for which the departures or arrivals shall be retrieved. Such ID can be retrieved from the location.name or location.nearbystops services.
extId	Mandatory if id is not specified			Specifies the external station/stop ID. Such ID can be retrieved from the location.name or location.nearbystops services
direction	Optional	See 2.2 or 2.3		If only vehicles departing or arriving from a certain direction shall be returned, specify the direction by giving the station/stop ID of the last stop on the journey.
date	Optional	See 1.2.2	Current server date	Sets the start date for which the departures or arrivals shall be retrieved.
time	Optional	See 1.2.2	Current server time	Sets the start time for which the departures or arrivals shall be retrieved.
products	Optional	-	-	Decimal value defining the product classes to be included in the search. It represents a bitmask combining bit number of a product as defined in the HAFAS raw data file *zugart*. For example, regional trains are product class 2 and local trains are class 3, while busses are 4. If you would like to search for local and regional trains only, you would need a bitmask where bits 2 and 3 are set. Calculation is 2^2 + 2^3 = 12 which would be the parameter value for "products". When searching for busses only, "products" need to be set to 16 = 2^4.

operators	Optional	All operator codes from HAFAS raw data file *betrieb*.	-	Only journeys provided by the given operators are part of the result. To filter multiple operators, separate the codes by comma. E.g. filter for A and B operator: operators=A,B.
maxJourneys	Optional		-	Maximum number of journeys to be returned.

As a response, each service will return a result according to `hafasRestDepartureBoard.xsd` or `hafasRestArrivalBoard.xsd` respectively. This will contain a list of departures/arrivals with train/line number, type of transport, departure or arrival times (incl. real-time), departure or arrival stop/stations (might be different from requested stop), direction text and a track information if available. Every departure or arrival will also contain a reference to the journey detail service.

2.7.2 Example

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

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

Result: (abbreviated)

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

2.8 Journey Detail Service

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

2.8.1 Request Parameters

Name	Use	Range	Default	Description
accessId	Mandatory	-	-	Access ID for identifying the requesting client.
id	Mandatory	See 2.2 or 2.3	-	Specifies the station/stop ID for which the departures or arrivals shall be retrieved. Such ID can be retrieved from the location.name or location.nearbystops services. It may be necessary to escape the `|` character by its URL encoding `%7C`.
date	Optional	See 1.2.2	Current server date	Day of operation

2.8.2 Example

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

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

Result: (abbreviated)

```
<JourneyDetail xmlns="hafas_rest_v1">
    <Stops>
        <Stop id="A=1@O=Drammen@X=10204842@Y=59740160@U=70
            @L=7601421@" name="Drammen" routeIdx="0" extId="7601421"
            lon="10.204842" lat="59.74016" depDate="2014-06-01"
            depTime="17:22:00" track="3"/>
        <Stop id="A=1@O=Asker@X=10434552@Y=59833747@U=70@L=7601413@"
            name="Asker" routeIdx="1" extId="7601413"
            lon="10.434552" lat="59.833747" depDate="2014-06-01"
            depTime="17:35:00" track="3"/>
        <Stop id="A=1@O=Sandvika@X=10526017@Y=59893022@U=70
            @L=7601408@" name="Sandvika" routeIdx="2"
```

```
            extId="7601408" lon="10.526017" lat="59.893022"
            depDate="2014-06-01" depTime="17:41:00" track="4"/>    …
    </Stops>
    <Names>
        <Name name="F2" routeIdxFrom="0" routeIdxTo="8"
            number="3781" category="5"/>
    </Names>
    <Directions>
        <Direction routeIdxFrom="0" routeIdxTo="8">
            Gardermoen
        </Direction>
    </Directions>
</JourneyDetail>
```

2.9 XSD Service

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

2.9.1 Example

Request: List all available XSD files

`<baseurl>/xsd?list`

Response:

- `hafasRestArrivalBoard.xsd`
- `hafasRestDepartureBoard.xsd`
- `hafasRestJourneyDetail.xsd`
- `hafasRestLocation.xsd`
- `hafasRestTrip.xsd`
- `hafasRestPolyline.xsd`
- `hafasRestError.xsd`

Request: Return XSD for Location service

`<baseurl>/xsd?hafasRestLocation.xsd`

Result:

<Content of hafasRestLocation.xsd>

2.10 Status Service

The status service will provide information about the system health. At least, it will provide an HTTP response code 200 if the system is up and running. It is strongly recommended to block access to this service for external access.

The response format will be JSON only. It provides information about each connector, internal as well as external. Following fields will be filled:

- CamelId: Internal ID of this connector. Following IDs should be present:
  - locationNameContext for Location.name service
  - locationNearbyContext for Location.nearby service
  - depBoardContext for Departure Board service
  - arrBoardContext for Arrival Board service
  - tripContext for Trip service
  - journeyDetailsContext for Journey Detail service
  - ifpContext for Itineraries for period service
  - xsdContext for XSD service
  - statusContext for Status service
  - hciContext for the internal connection to the HAFAS server
  - coreContext for the internal state of ReST Interface
- Uptime: Time the connector is up and running
- State: Connector is “Started” or “Stopped”. In a full working environment, it should always be “Started”.
- ExchangesTotal: Total amount of exchanges on this particular connector.
- ExchangesCompleted: Amount of completed exchanges on this particular connector.
- ExchangesFailed: Amount of failed exchanges on this particular connector.
- LastExchangeCompletedTimestamp: At which server time was the last exchange completed on that connector.
- MinProcessingTime: Minimum processing time of an exchange using this connector in milliseconds.
- MaxProcessingTime: Maximum processing time of an exchange using this connector in milliseconds.
- MeanProcessingTime: Mean processing time of an exchange using this connector in milliseconds.

2.10.1 Example

Request:

```
<baseurl>/status
```

Response:

```
[
    {"Uptime":"4 days 5 hours", "MinProcessingTime":"0",
        "CamelId":"xsdContext", "ExchangesTotal":"6",
        "State":"Started",
        "LastExchangeCompletedTimestamp":"Mon May 19 15:38:25
            CEST 2014",
        "MeanProcessingTime":"1", "ExchangesFailed":"0",
        "ExchangesCompleted":"6", "MaxProcessingTime":"3"},
    {"Uptime":"4 days 5 hours", "MinProcessingTime":"0",
        "CamelId":"ifpContext", "ExchangesTotal":"0",
        "State":"Started",
        "LastExchangeCompletedTimestamp":"",
        "MeanProcessingTime":"0",
        "ExchangesFailed":"0", "ExchangesCompleted":"0",
        "MaxProcessingTime":"0"},
    …
]
```


2.11 Service overview

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

Request:

```
<baseurl>/
```

Responses

All services return their responses either in XML or JSON format (see 1.2.7). Every response is defined in a separate XSD file. The following sections will describe only a brief description of the responses. For more details please refer to the corresponding xsd which is available vi an base URL and adding the name of the xsd.

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

2.12 Location response

This is the response of the `location.name`, `location.nearbystops` and `location.nearbypoi` 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`.

For more details please refer to the Schema <baseURL>`/xsd?`**hafasRestLocation.xsd**.

2.13 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.

For more details please refer to the Schema <baseURL>`/xsd?`**hafasRestTrip.xsd**

2.14 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`.

For more details please refer to the Schema
<baseURL>`/xsd?`**hafasRestDepartureBoard.xsd**

2.15 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`.

For more details please refer to the Schema <baseURL>`/xsd?`**hafasRestArrivalBoard.xsd**

2.16 Journey detail response

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

For more details please refer to the Schema
<baseURL>`/xsd?`**hafasRestJourneyDetail.xsd**

2.17 Polyline response structure

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

For more details please refer to the Schema <baseURL>`/xsd?`**hafasRestPolyline.xsd**

3 Error codes and messages

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

3.1 ReST Request Errors

Code	Text
R0001	Unknown service method
R0002	Invalid or missing request parameters
R0007	Internal communication error
R5000	Access denied

3.2 Backend Server Errors

Code	Text
S1	The desired connection to the server could not be established or was not stable.

3.3 Trip Search Errors

Code	Text
H9380	Dep./Arr./Intermed. or equivalent stations defined more than once
H9360	Error in data field
H9320	The input is incorrect or incomplete
H9300	Unknown arrival station
H9280	Unknown intermediate station
H9260	Unknown departure station
H9250	Part inquiry interrupted
H9240	Unsuccessful search

H9230	An internal error occurred
H9220	Nearby to the given address stations could not be found
H900	Unsuccessful or incomplete search (timetable change)
H892	Inquiry too complex (try entering less intermediate stations)
H891	No route found (try entering an intermediate station)
H890	Unsuccessfully search.
H500	Because of too many trains the connection is not complete
H460	One or more stops are passed through multiple times.
H455	Prolonged stop
H410	Display may be incomplete due to change of timetable
H390	Departure/Arrival replaced by an equivalent station
H895	Departure/Arrival is too near
H899	Unsuccessful or incomplete search (timetable change)

3.4 Departure and Arrival Board Errors

Code	Text
SQ001	No station board available.
SQ002	There was no journey found for the requested board or time.

3.5 Journey Details Errors

Code	Text
TI001	No trip journey information available.

4 Document Version

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

15.04.2015	1.20	rhu/mfr	Removing xsd from this document Location.name Service.Request: Filter “type” added Trip search service/Interval trip search service.Request: - parameter passlist added - parameters originExtId/destExtId added - parameter maxChange added - parameters originName/destName removed Trip search service/Interval trip search service.Response: - arrival and departure time added to the passlist - prognosis data for arrival and departure time added to the passlist - track data added to the passlist
17.04.2015	1.21	mfr	Stationboard service - Removing use* query parameters from station board service. - Add extId parameter Trip service - Add originExtId parameter - Add destExtId parameter Service overview - Add the description of this service General - Add error code R5000, access denied