

Specification

HAFAS ReST Interface

Easy access to

HAFAS journey planner systems

Version 1.10.1

2014-05-21

HaCon Ingenieurgesellschaft mbH
Lister Str. 15
30163 Hanover
Germany

<u>1</u>	<u>CHANGELOG</u>	<u>3</u>
<u>2</u>	<u>INTRODUCTION</u>	<u>4</u>
<u>3</u>	<u>GENERAL PRINCIPLES</u>	<u>5</u>
3.1	Coordinates	5
3.2	Date and time formats	5
3.3	Stateless service vs. data dependency	5
3.4	Route index	5
3.5	Realtime information	5
3.6	Versioning	6
3.7	Response Format	6
3.8	Authentication	7
<u>4</u>	<u>SERVICES</u>	<u>8</u>
4.1	Location Service	8
4.1.1	Location.name Service	8
4.1.2	Location.allstops Service	8
4.1.3	Location.nearbystops Service	9
4.1.4	Location.nearbyaddress Service	9
4.2	Trip service	10
4.2.1	Origin, destination and via	10
4.2.2	Date and time	10
4.2.3	Types of transport	10
4.2.4	Walks from and to coordinates	11
4.2.5	Bike and public transport	12
4.2.6	Car and public transport	13
4.2.7	Only walk, bike and car	13
4.2.8	Changes and change times	14
4.2.9	Medical Transport	15
4.2.10	Accessibility	15
4.2.11	Response formatting	16
4.2.12	Example request	17

HAFAS Public Access Gateway

4.2.13 Response	17
4.3 Stationboard services	17
4.4 Journey detail service	19
4.5 Geometry service	19
4.6 SystemInfo Service	20
5 RESPONSE FORMATS	21
5.1 Location response	21
5.2 Trip Response	23
5.3 Departure board response	30
5.4 Arrival board response	32
5.5 Journey detail response	34
5.6 Geometry Response	38
5.7 SystemInfo response	40
6 ERROR CODES	42

1 Changelog

Version	Date	Changes
1.07	2012-06-04	<ul style="list-style-type: none"> - Added Changelog - Added SystemInfo service
1.08	2013-01-07	<ul style="list-style-type: none"> - Corrected Notes/Note screenshot for TripResponse - Added Chapter 6 for Error Codes
1.09	2013-06-16	<ul style="list-style-type: none"> - Added short names to journeys (attribute "sname")
1.10	2014-04-27	<ul style="list-style-type: none"> - Added section 4.2.9: Medical transport journeys including taxi connections from start and/or to destination. - Added section 4.2.10: Journey search with accessibility criteria - Multimodal search that combines car, commuter parking and bicycle rides with line based public transport (section 4.2.5 and 4.2.6) - Itinerary display for walks, bike and car rides (section 4.2.11 and 4.5)
1.10.1	2014-05-21	<ul style="list-style-type: none"> - Added section 4.2.7: walk-, bike- and car-only-trips. - Added section 0: The location.nearbyaddress Service

2 Introduction

The public interface is implemented as a ReST¹ (**R**epresentational **S**tate **T**ransfer) interface which provides different methods for the different functionalities of the journey planner, which are the following services:

- Location
- Trip
- DepartureBoard
- ArrivalBoard
- JourneyDetail
- Geometry
- SystemInfo

While Location, Trip, ArrvialBoard and DepartureBoard can be called directly the JourneyDetail-Method can only be called by a reference given in a result of the Trip, or DepartureBoard services. Geometry-Method can only be called by a reference given in a result of the Trip or JourneyDetail request.

The system only 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 in ISO-8859-1 URL encoded. The result of each request will be delivered either as XML or JSON (see 3.7) response. If the encoding of URL parameters is not right, 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 are described based on this given base url *<baseurl>*.

¹ See <http://rest.elkstein.org/> for a tutorial on ReST interfaces.

3 General principles

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

3.1 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).

3.2 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.

3.3 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 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 therefore and can only be done on own risk for undetermined behaviour when reusing these IDs or references.

3.4 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.

3.5 Realtime information

Realtime information will be included in the service as far as it is available in the web based journey planner. It is always delivered in addition to the planned departures and arrivals.

3.6 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 should always use a special version of the api (v2 for example), your url would look like this: `http://<baseUrl>/v2/<servicename>`

3.7 Response Format

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

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": { "$": "foo" } } }`
- If there are multiple elements with the same name the JSON code contains an array for these element.
`<a>foo1foo2`
becomes
`{ "a": { "b": [{ "$": "foo1" }, { "$": "foo2" }] } }`

- Attribute names become object properties

```
<a atb="foo1">foo2</a>
```

becomes

```
{ "a": { "atb" : "foo1", "$" : "foo2" } }
```

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

XML:

```
<Trip>
  <Leg name="Buss 61" type="BUS" id="9015014006100032" direction="Vänernsberg Resecentrum">
    <Origin name="Trollhättan, Resecentrum, Trollhättan" type="ST" id="9022014081032014" routeldx="8" time="13:12"
date="2011-09-13" track="L " />
    <Destination name="NÄL, Trollhättan" type="ST" id="9022014081089002" routeldx="22" time="13:33" date="2011-09-13"
track="B " />
  </Leg>
</Trip>
```

JSON:

```
"Trip": {
  "Leg": {
    "name": "Buss 61",
    "type": "BUS",
    "id": "9015014006100032",
    "direction": "Vänernsberg Resecentrum",
    "Origin": {
      "name": "Trollhättan, Resecentrum, Trollhättan",
      "type": "ST",
      "id": "9022014081032014",
      "routeldx": "8",
      "time": "13:12",
      "date": "2011-09-13",
      "track": "L "
    },
    "Destination": {
      "name": "NÄL, Trollhättan",
      "type": "ST",
      "id": "9022014081089002",
      "routeldx": "22",
      "time": "13:33",
      "date": "2011-09-13",
      "track": "B "
    }
  }
}
```

3.8 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: **authKey=<your_key_here>**.

Please contact Västtrafik in order to request an authentication key.

4 Services

4.1 Location Service

There are 3 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 5.1 for further details).

4.1.1 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 service has only one GET parameter which is called `input`. This parameter contains a string with the user input. 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 URL to call the service is the following:

<http://<baseurl>/location.name?input=user%20input>

4.1.2 Location.allstops Service

The `location.allstops` service returns a list of all stops available in the journey planner. Be aware that a call of this service is very time consuming and should be only requested when it is really needed.

The URL to call the service is the following:

<http://<baseurl>/location.allstops>

4.1.3 Location.nearbystops Service

The `location.nearbystops` service returns a list of stops around a given center coordinate. The returned results are ordered by their distance to the center coordinate.

Possible parameters:

Name	Use	Range	Default	Description
originCoordLat	Mandatory	See 3.1	-	Latitude of center coordinate
originCoordLong	Mandatory	See 3.1	-	Longitude of center coordinate
maxNo	Optional	1-1000	10	Maximum number of returned stops
maxDist	Optional	1-3000	1000	Maximum distance from the center coordinate

The URL to call the service is the following:

<http://<baseurl>/location.nearbystops?originCoordLong=11.981211&originCoordLat=57.709792&maxNo=5>

4.1.4 Location.nearbyaddress Service

The `location.nearbyaddress` service returns the address nearest a given coordinate.

Possible parameters:

Name	Use	Range	Default	Description
originCoordLat	Mandatory	See 3.1	-	Latitude of coordinate
originCoordLong	Mandatory	See 3.1	-	Longitude of coordinate

4.2 Trip service

4.2.1 Origin, destination and via

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 (e.g. the current position of a mobile device).

Parameters specifying both origin and destination are mandatory in calls to the `trip` service. When specifying a stop as origin, the parameter `originId` is used, while `originCoordLat`, `originCoordLong`, and `originCoordName` are used to specify a (named) coordinate. For the destination, the corresponding parameters are named either `destId` or `destCoordLat`, `destCoordLong` and `destCoordName`. The `origin-/destCoordName` parameters are the names of the address at the specified coordinate.

It is also possible to define a via-stop/station. This forces the journey planner to search for trips which pass the defined station. The parameter is called `viaId`. When searching for a trip that goes via a coordinate, rather than a stop, two separate `trip` requests need to be combined into one.

4.2.2 Date and time

The departure time and date are defined with the parameters `date` and `time`. If the date is not set the current date will be used (server time). If the parameter `time` is not set the current server time will be used to perform the request.

To specify that the given time and date is not the departure time but the latest time to arrive at the destination you can use the parameter `searchForArrival=1`.

4.2.3 Types of transport

It is possible to switch off specific means of transport by using one of the following optional parameters:

HAFAS Public Access Gateway

- `useVas=0` // Vasttågen
- `useLDTrain=0` // Long Distance Trains
- `useRegTrain=0` // Regional Trains
- `useBus=0`
- `useBoat=0`
- `useTram=0`

The default value is that all means of transport are switched on (value 1). If no parameter is set this default value applies.

In addition to the flags above you can use `excludeDR=1` to exclude journeys which require tel. registration, by default they are included.

4.2.4 Walks from and to coordinates

When a search is performed from/to a coordinate, the journey planner will (by default) return journeys that include walks to/from a suitable nearby stop. In order to set the maximum walking distance from/to the coordinate, the parameter `maxWalkDist` can be used to pass the distance in meters (default value: 2000).

The walking speed can be influenced using the parameter `walkSpeed`. The passed value is given in percent of “normal” speed (valid range: 20%--180%). Västtrafik’s official implementations use the following values for its preset walking speeds:

- Slow: 85%
- Normal: 100% (Y km/h)
- Fast: 115%

NOTE: the settings for `maxWalkDist` and `walkSpeed` don’t apply to walks between stops when changing from a public transport leg to another, only to the walks from/to a origin or destination coordinate.

If no journeys with walks from/to coordinates should be displayed (e.g. when only journeys that combine Medical Transport Connections, car or bicycle with public

transport are of interest), the `trip` request parameters `originWalk=0` and `destWalk=0` can be used.

4.2.5 Bike and public transport

The journey planner supports requests for journeys where a bike can be taken to a nearby stop.

To search for trips with a bike ride from the origin to a nearby stop, where the journey continues using public transport, `originBike=1` should be added to the `trip` request parameters. Any trip that includes a bike ride that is between or equal to `maxWalkDist` and `maxBikeDist` (default value, in m: 3000) will then be displayed. It is possible to combine e.g. `originWalk`, `originBike`, `originCal` and `originCarWithParking` so that the journey planner will present the best (fastest and/or most convenient) journeys that either require the customer to walk, bike or go by car to a nearby stop.

Note that there is no `destBike` parameter, as very few lines allow carrying a bike onto the vehicle.

For bike rides, it is also possible to optimize for either the fastest route or a route that is made up of a larger percentage of bike road. The request parameter `bikeCriterion` is used to model this choice such that:

- `bikeCriterion=F` (Fastest) → minimize travel time (default setting)
- `bikeCriterion=D` (Dedicated bike roads) → maximize bike road use

The `bikeProfile` parameter determines the altitude profile of the route, based on a setting for how fast the user can bike when it is steep:

- `bikeProfile=E` (Easy) → minimize steepness
- `bikeProfile=N` (Normal) → default setting
- `bikeProfile=P` (Powerful) → allow more steepness

4.2.6 Car and public transport

The journey planner supports requests for journeys where a car can be taken to/from a nearby stop. Three different types of trips that combine car with public transport can be presented:

- **Car drop-off (aka kiss-and-ride) trips**, where the customer travels by car from the origin and is dropped off at a stop to continue the trip using public transport, may be presented when `originCar=1`.
- **Park-and-ride trips**, where the customer travels by car from the origin, parks at a commuter parking and walks to a nearby stop to continue the trip using public transport, may be presented when `originCarWithParking=1`.
- **Car pick-up trips**, where the customer travels by public transport to a stop where it gets picked up by a car and is taken to the destination, may be presented when `destCar`.

Car rides must be between or equal to `maxBikeDist` (default 3000) and `maxCarDist` (default 5000) meters long. If `origin-/destWalk` or `originBike` is also set, the journey planner will present the best (fastest and/or most convenient) journeys that either require the customer to walk, bike or go by car to a nearby stop.

4.2.7 Only walk, bike and car

The journey planner supports requests for walk-, bike- and car-only-trips using the following parameters:

- `onlyWalk=1`
- `onlyBike=1`
- `onlyCar=1`

If no trips that include Public Transportation are to be included in the response, the parameter `usePT=0` can be used.

4.2.8 Changes and change times

The maximum number of changes in the journeys returned by the trip service can be set using the parameter `maxChanges=<integer>`.

When calculating journey suggestions, the journey planner uses values for the minimal margin (in minutes) for changes between different public transport legs. This value, called *change margin*, is used to compensate for deviations from the planned time table times of the arriving and departing vehicles involved in a change, to minimize the risk of missing a connection. For each stop area, a default change margin has been defined. A common value of the default change margin is 5 minutes, but there are stops with both longer and shorter default change margins (in the approximate range 3-10 minutes).

In addition to the change margin, walk times between different stop areas are taken into account. Walk times between stop points within a stop area are currently not taken into account specifically, but are included implicitly in the default change margin.

In order to prolong the minimal change times between the public transport legs of the returned journeys the parameter `additionalChangeTime=<number of minutes>` can be set. The default value is 0 minutes. The minimal time of a change will thus be calculated as *default change margin* + `additionalChangeTime` + *walk time between stop areas*.

If the default change margin should be ignored, you can set the parameter `disregardDefaultChangeMargin=1` (default is "0"). The minimal time of a change will then be calculated as $0 + \text{additionalChangeTime} + \text{walk time between stop areas}$.

IMPORTANT NOTE: journeys that are presented when the default change margin has been disregarded are not covered by Västtrafik's travel warranty (Swedish: resegaranti). This will be clarified in the trip response by addition of the attribute `travelWarranty="false"` to each trip-tag. When travel warranty has been void, this must be clearly communicated in connection with the presented journey suggestions, eg. as *"The travel warranty is void for this journey."/"/"Resegaranti gäller ej för denna resa."*

4.2.9 Medical Transport

By default, Medical Transport (a.k.a. "RONDEN") Lines are included in search results, since they are considered to be regular buses and thus are included/excluded via the `useBus` parameter. It is possible to override the exclusion of Medical Transport Lines when `useBus=0`, by adding the parameter `useMedical=1`. This combination of parameter settings will result in no buses being included in journey results, except for the Medical Transport Lines.

When a customer has been granted permission to travel using Medical Transport, it is also possible to offer a connection using taxi or a special vehicle (suitable for wheelchair users). A Medical Transport Connection goes either from the origin of the trip to a nearby stop that allows connecting to, or to the destination from a similarly qualified stop. The parameter `originMedicalConn=1` enables searching for connections from the origin, and `destMedicalConn=1` enables connections to the destination of the trip. It is also possible to force the use of connections in either end of the trip, by setting `originWalk=0` and/or `destWalk=0` in combination with the corresponding parameter above.

4.2.10 Accessibility

For travellers with accessibility needs, it is possible to set the following requirements for equipment/capabilities of the vehicles that are used in journeys presented by the journey planner:

- `wheelChairSpace=1` → at least one wheelchair space is present in the vehicle
- `strollerSpace=1` → space for stroller, baby carriage or rollator in the vehicle
- `lowFloor=1` → the vehicle is equipped with a low floor section, but not necessarily a ramp or lift, see below
- `rampOrLift=1` → the vehicle is equipped with ramp or lift that allows fully barrier-free boarding and alighting

The parameters each correspond to one of the checkboxes that can be found under “More options” in the public web version of the journey planner. When both `lowFloor=1` and `rampOrLift=1`, the condition is that one of the two needs to be true for a journey to be included in the result (logical OR). All other parameters are unrelated and need to be fulfilled individually (logical AND).

When a journey departs within a couple of hours of when it is sought, real-time information about the vehicles that will be in traffic on the various legs of the journey may be available. This is signalled in the journey response by the presence of `rtDate` and `rtTime` attributes on the `Origin` and `Destination` of a `Leg`. The real-time information then also includes the above information about equipment/capabilities of the vehicle.

When journeys are sought longer in advance – e.g. for the next day – accessibility is calculated based on information about *all* vehicles that may operate on the upcoming trip. The rule is set to be conservative, requiring that 100 percent of the potential vehicles fulfil the requirements for equipment/capabilities.

4.2.11 Response formatting

If the reference URL for the Journey Detail Service (4.4) is not needed in the result, you can pass the parameter `needJourneyDetail=0`.

If the parameter `needGeo=1` is passed in the URL of the trip service, the result will contain a reference link for each leg of the resulting trips which can be used to request

the geometry (and – if `needItinerary=1` – the itinerary) of the leg using the Geometry service (4.5).

The number of the returned trips in the result can be specified using the parameter `numTrips=<integer>` with a valid range of 1 – 6. Note that this is an approximated number and the result may contain more trips than specified.

4.2.12 Example request

A trip request for a trip from Göteborg C to some coordinate on the 19th of September in 2011 at 7:02 am excluding busses as means of transport looks like this:

<http://<baseurl>/trip?originId=9022014008000000&destCoordLat=<lat>&destCoordLong=<long>&destCoordName=<NameOfDestination>&date=2011-09-19&time=07:02&useBus=0>

4.2.13 Response

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

4.3 Stationboard services

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

The service can only be called for stops/stations by using according ID retrieved by the `location` method. The parameter is called `id`. The time and date are defined with the parameters `date` and `time`.

It is possible to switch off certain means of transport by using one or several of the following optional switches

- `useVas=0` // Vasttågen
- `useLDTrain=0` // Long Distance Trains

HAFAS Public Access Gateway

- `useRegTrain=0` // Regional Trains
- `useBus=0`
- `useBoat=0`
- `useTram=0`

The default value of these switches is 1 (on) which also applies if the parameter isn't defined at all.

In addition to the flags above you can use `excludeDR=1` to exclude journeys which require tel. registration, by default they are included.

The parameter `timeSpan` can be used to get the next departures in a specified timespan of up to 24 hours (unit: minutes, maximum value: 1439). If this parameter is not set, the result will contain the next 20 departures. If `timeSpan` is set you can further reduce the number of returned journeys by adding the parameter `maxDeparturesPerLine`, which will cause only the given number of journeys for every combination of line and direction.

If the reference URL for the Journey Detail Service (4.4) is not needed in the result, you can pass the parameter `needJourneyDetail=0`.

In order to get only departures of vehicles with a specified direction you can use the parameter `direction=<stopid>`.

A departure board for Göteborg C main station for the next 20 departures on 19th September, 2011 at 07:02am excluding all busses can be retrieved by calling

```
http://<baseurl>/departureBoard?id=9022014008000000&date=2011-09-19&time=07:02&useBus=0
```

As a response the service will return a result according to `hafasRestDepartureBoard.xsd`. This will contain a list of departures with train/line number, type of transport, vehicle accessibility (if available), departure times (incl. realtime), 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.

In addition to departure boards the service `arrivalBoard` delivers arriving journeys at a specified stop. The parameters are identical to the parameters of the `departureBoard` service.

As a response the service will return a result according to `hafasRestArrivalBoard.xsd`. This will contain a list of arrival with train/line number, type of transport, vehicle accessibility (if available), arrival times (incl. realtime), departure stop/stations (might be different from requested stop), the name of the origin stop and a track information if available. Every arrival will also contain a reference to the journey detail service.

4.4 Journey detail service

The `journeyDetail` service will deliver information about the complete route of a vehicle. This service can't be called directly but only by reference URLs in a result of a `trip` or `departureBoard` request. It contains a list of all stops/stations of this journey including all departure and arrival times (with realtime data if available) and additional information like specific attributes about facilities and other texts.

The response will be returned a result according to the format described in `hafasRestJourneyDetails.xsd`.

4.5 Geometry service

The Geometry service will return the polyline for a leg. This service can't be called directly but only by reference URLs in a result of a `trip` or `JourneyDetail` request. The result contains a list of WGS84 coordinates which can be used to display the polyline on a map.

The response will be returned as a result according to the format described in `hafasRestGeometry.xsd`.

If the parameter `needItinerary=1` is passed in the URL of the `trip` request that contained the reference to the Geometry service, the geometry reference link will also contain an itinerary for walk, bike and car legs, that can be used to generate turn-by-turn instructions.

4.6 SystemInfo Service

The SystemInfo service provides information about the travelplanner system and the underlying data. It will return the begin of end of the timetable period and the creation date of the timetable data. It is called without any parameters (except the authentication key):

```
http://<baseurl>/systeminfo?authKey=<your authentication key>
```

5 Response formats

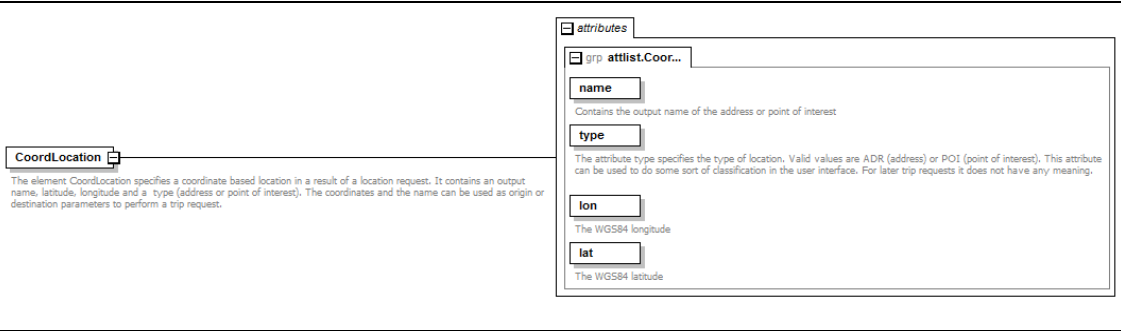
All services return their responses either in XML or JSON format (see 3.7). Every response is defined in a separate XSD file. The following sections will describe the responses more in detail. The formats might be enhanced in the future so the implementation of the parsing should be implemented in view of future possible changes.

5.1 Location response

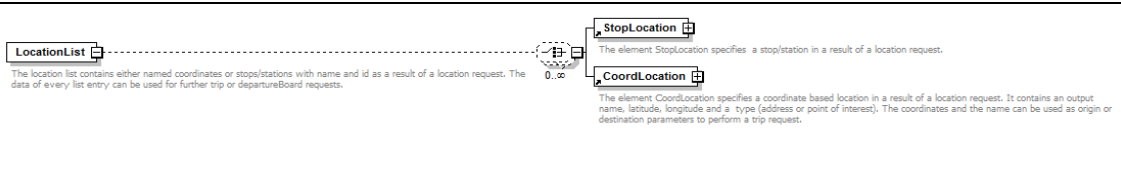
The location consists of a list of entries, which are either stops/stations or named coordinates. The root element of the response is `LocationList`.

Schema `hafasRestLocation.xsd`


element `CoordLocation`

<p>diagram</p>  <p>The diagram shows the <code>CoordLocation</code> element with the following attributes:</p> <ul style="list-style-type: none"> <code>name</code>: Contains the output name of the address or point of interest. <code>type</code>: The attribute type specifies the type of location. Valid values are ADR (address) or POI (point of interest). This attribute can be used to do some sort of classification in the user interface. For later trip requests it does not have any meaning. <code>lon</code>: The WGS84 longitude. <code>lat</code>: The WGS84 latitude. 	<p>The element <code>CoordLocation</code> specifies a coordinate based location in a result of a location request. It contains an output name, latitude, longitude and a type (address or point of interest). The coordinates and the name can be used as origin or destination parameters to perform a trip request.</p>
<p>used by</p>	<p>element LocationList</p>

element `LocationList`

<p>diagram</p>  <p>The diagram shows the <code>LocationList</code> element containing a list of <code>StopLocation</code> and <code>CoordLocation</code> elements. The list is represented by a dashed line with a plus sign and a range of 0..∞.</p>	<p>The location list contains either named coordinates or stops/stations with name and id as a result of a location request. The data of every list entry can be used for further trip or departureBoard requests.</p> <p>The element <code>StopLocation</code> specifies a stop/station in a result of a location request.</p> <p>The element <code>CoordLocation</code> specifies a coordinate based location in a result of a location request. It contains an output name, latitude, longitude and a type (address or point of interest). The coordinates and the name can be used as origin or destination parameters to perform a trip request.</p>
<p>children</p>	<p>StopLocation CoordLocation</p>

element **StopLocation**

<p>diagram</p> 	<div style="border: 1px solid black; padding: 5px;"> <p>attributes</p> <p>grp attlist.Stop...</p> <p>id This ID can either be used as originId or destId to perform a trip request or to call a departure board.</p> <p>name Contains the output name of this stop or station</p> <p>lon The WGS84 longitude</p> <p>lat The WGS84 latitude</p> <p>track Track information, if available.</p> <p>weight This value specifies some kind of importance of this stop. The more traffic at this stop the higher the weight. The range is between 0 and 32767. This attribute is just contained in the location.allstops response.</p> </div>
<p>used by</p>	<p>element LocationList</p>

Example Response

```
<LocationList xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="..." servertime="10:30" serverdate="2012-01-25" >
  <StopLocation name="Brunnered, Trollhättan" lon="12.401176" lat="58.262900"
id="9021014081188000" />
  <StopLocation name="Brunnaliden, Trollhättan" lon="12.272810" lat="58.252095"
id="9021014081871000" />
  <CoordLocation name="BRUNSOPPEVÄGEN, 461 54 TROLLHÄTTAN" lon="12.283085" lat="58.262469"
type="ADR" />
  <!-- ... -->
</LocationList>
```

5.2 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`.

Schema `hafasRestTrip.xsd`


element `Destination`

diagram	
used by	element Leg

element `GeometryRef`

diagram	
used by	element Leg

element **JourneyDetailRef**

diagram	 <p>The diagram shows the structure of the JourneyDetailRef element. It is a box containing the text "Reference to journey details of this leg." To its right is a larger box representing its content model, which includes an attributes container and a grp attlist.Jour... container. The grp attlist.Jour... container contains a ref element, which is described as "Contains a URL to call the ReST interface for journey details."</p>
used by	element Leg

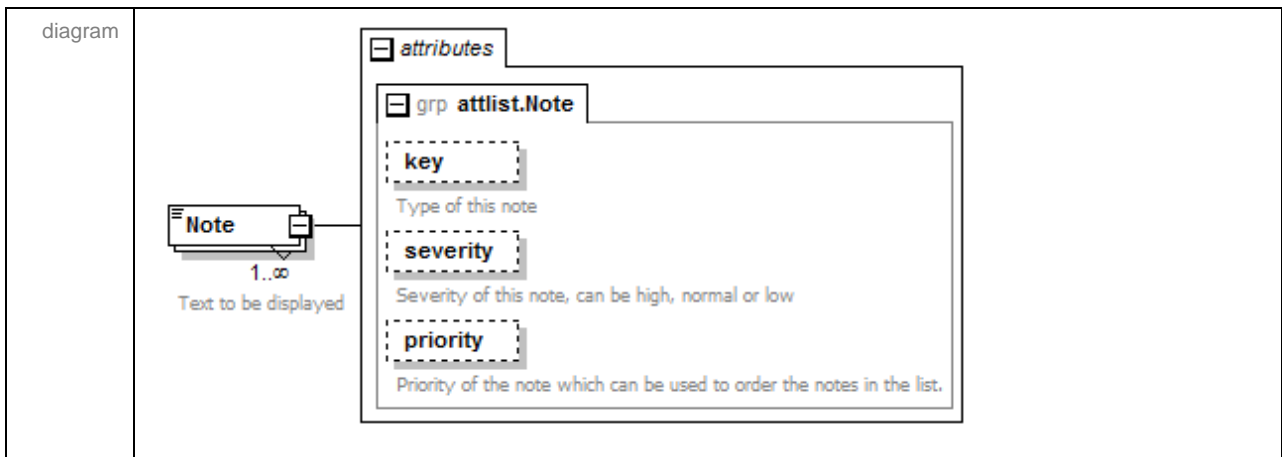
element Leg

<p>diagram</p>	<p>attributes</p> <ul style="list-style-type: none"> grp attlist.Leg <ul style="list-style-type: none"> name: The attribute name specifies the name of the leg (e.g. "Bus 100"). sname: Short name of the leg (e.g. "LERS", "11") type: The attribute type specifies the type of the leg. Valid values are VAS, LDT (Long Distance Train), REG (Regional train), BUS, BOAT, TRAM, TAXI (Taxi/Telebus). Furthermore it can be of type WALK, BIKE and CAR if these legs are routes on the street network. id: ID of the journey cancelled: Will be true if this journey is cancelled reachable: Will be true if this journey is not reachable due to delay of the feeder direction: Direction information. booking: Will be 1 if this journey needs to be booked night: Will be 1 if this journey is a night journey fgColor: Foregroundcolor of this line bgColor: Backgroundcolor of this line stroke: Stroke style of this line accessibility: will only be set if the vehicle has wheelchair + ramp/lift or lowfloor according to realtime data kcal: Energy use percentBikeRoad: Percentage of the route that is made up of bike roads Origin: Origin of a leg including location name, location type, location route index (if available), departure time and date, realtime departure (if available), track and realtime track (if available) Destination: Destination of a leg including location name, location type, location route index (if available), arrival time and date, realtime arrival time (if available), track and realtime track (if available) Notes: Contains a list with notes. JourneyDetailRef: Reference to journey details of this leg. GeometryRef: Reference to polyline of this leg.
children	Origin Destination Notes JourneyDetailRef GeometryRef
used by	element Trip

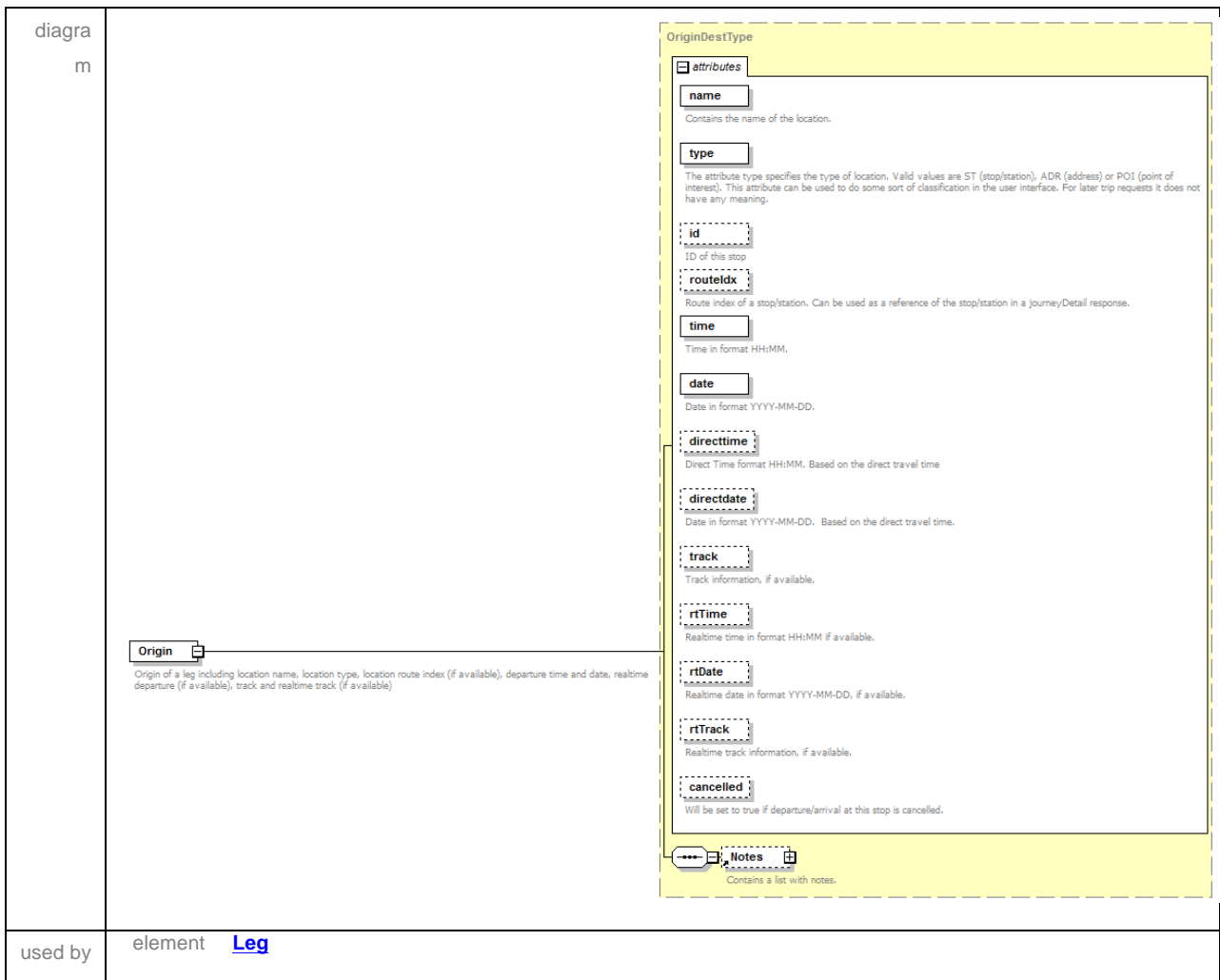
element Notes

<p>diagram</p>	<p>children</p> <p>Note</p>
used by	element Leg

element Notes/Note



element Origin



element Trip

<p>diagram</p>	<p>The element Trip contains all legs of the computed trip.</p> <p>attributes</p> <ul style="list-style-type: none"> grp attlist.Trip <ul style="list-style-type: none"> alternative: The type indicates whether this is an original connection or an realtime alternative. valid: The state indicates if the trip is still possible to ride based on the current realtime situation. travelWarranty: IMPORTANT NOTE: journeys that are presented when the default change margin has been disregarded are not covered by Västtrafik's travel warranty (Swedish: resegaranti). This will be clarified in the trip response by addition of the attribute travelWarranty="false" to each trip-tag. When travel warranty has been void, this must be clearly communicated in connection with the presented journey suggestions, eg, as "The travel warranty is void for this journey."/>"Resegaranti gäller ej för denna resa." type: Type of trip <p>Leg (0..12): A leg is one part of a trip. It can be either a walk, a bike or car ride or (in most cases) a journey by bus, train or some other means of transport.</p>
<p>children</p>	<p>Leg</p>
<p>used by</p>	<p>element TripList</p>

element TripList

<p>diagram</p>	<p>The trip list contains all found trips as a result of the trip request.</p> <p>attributes</p> <ul style="list-style-type: none"> grp attlist.TripL... <ul style="list-style-type: none"> error: If some problem occurs while calculating the trip you can find an error code here. Note: These error codes are not suitable for end users but only for reporting purposes. Most of the errors do not indicate a system failure but a reason, why no trip could be calculated for the request parameters. <p>Trip (0..∞): The element Trip contains all legs of the computed trip.</p>
<p>children</p>	<p>Trip</p>

Example Response

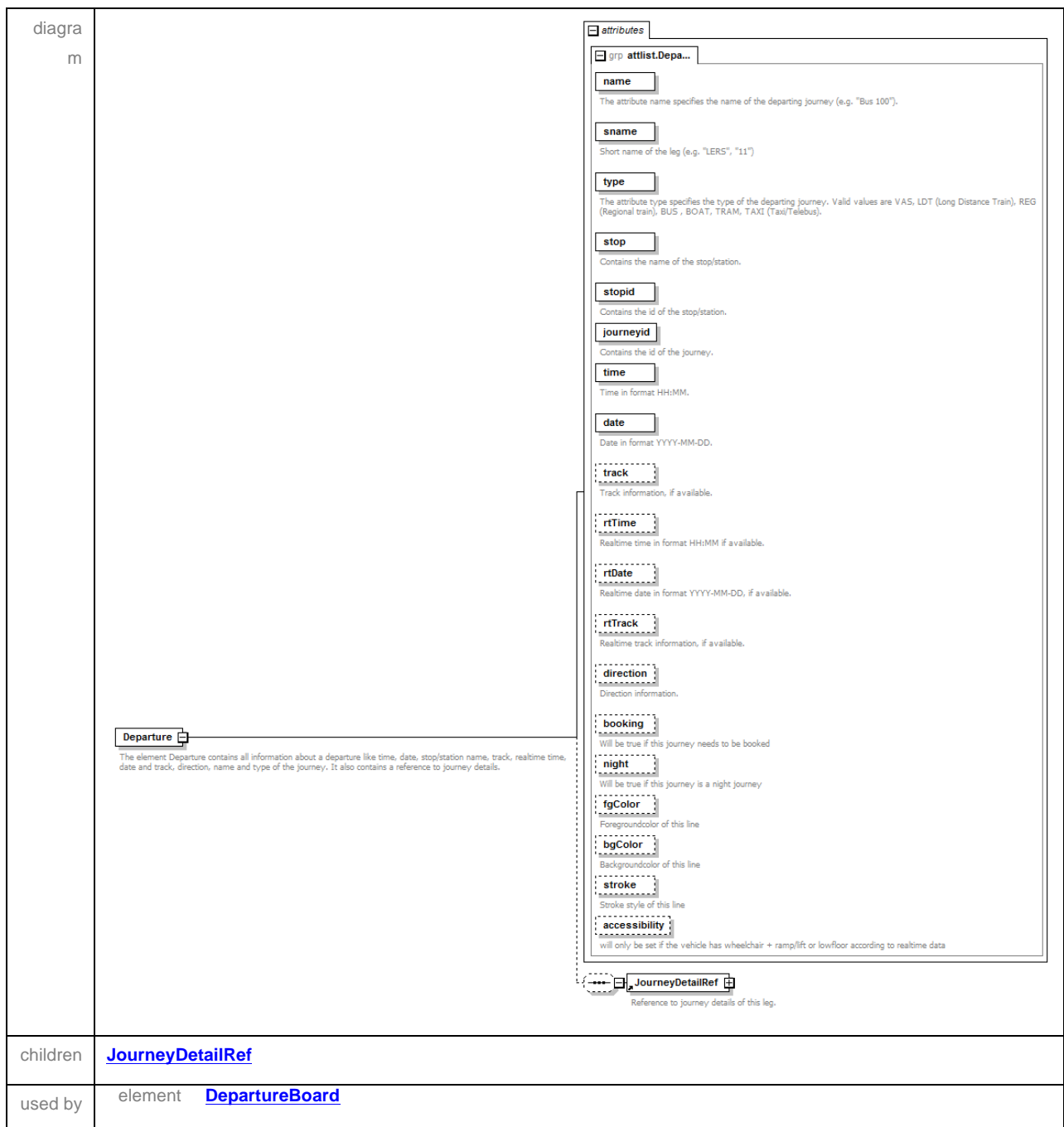
```
<TripList xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="..." servertime="10:30" serverdate="2012-01-25" >
  <Trip>
    <Leg name="Walk" type="WALK">
      <Origin name="Sahlgrenska Huvudentré, Göteborg" type="ST" id="9022014005600001"
time="11:35" date="2011-09-16" />
      <Destination name="Sahlgrenska, Göteborg" type="ST" id="9022014005597001" time="11:40"
date="2011-09-16" />
    </Leg>
    <Leg name="9 RONDEN" type="BUS" id="9015014890900002" direction="Åmål" booking="1"
fgColor="#003273" bgColor="#ffffff" stroke="Solid">
      <Origin name="Sahlgrenska, Göteborg" type="ST" id="9022014005597001" routeIdx="0"
time="11:40" date="2011-09-16" track="A " />
      <Destination name="NÄL, Trollhättan" type="ST" id="9022014081089003" routeIdx="17"
time="13:25" date="2011-09-16" track="C " />
      <Notes>
        <Note key="booking" severity="high" priority="1">Turen måste förbeställas på tel: 020-91
90 90 .|Ronden är avsedd för sjukresor, särskild taxa gäller.</Note>
      </Notes>
      <JourneyDetailRef ref=" http://..." />
    </Leg>
  </Trip>
</TripList>
```

5.3 Departure board response

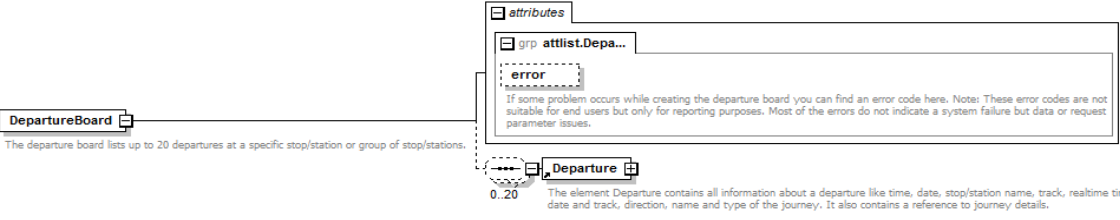
The departure board response contains a list of departures incl. all information concerning times, tracks, realtime data and journey. It also contains reference URLs to get more details for the different journeys. The root element is `DepartureBoard`.

Schema `hafasRestDepartureBoard.xsd`


element `Departure`



element DepartureBoard

diagram m	
children	Departure

element JourneyDetailRef

diagram	
used by	element Departure

Example Response

```

<DepartureBoard xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="..." servertime="10:30" serverdate="2012-01-25" >
  <Departure name="ÖRESUNDSTÅG" type="LDT" stopid="9022014008000001" stop="Göteborg C, Göteborg"
time="14:42" date="2011-09-16" journeyid="9015074110101087" direction="Malmö" track="1"
fgColor="#003273" bgColor="#ffffff" stroke="Solid">
    <JourneyDetailRef ref="http://..." />
  </Departure>
  <Departure name="VÄSTTÅGEN" type="VAS" stopid="9022014008000001" stop="Göteborg C, Göteborg"
time="14:47" date="2011-09-16" journeyid="9015014130103758" direction="Skee" track="1"
fgColor="#003273" bgColor="#ffffff" stroke="Solid">
    <JourneyDetailRef ref=" http://..." />
  </Departure>
</... ->
</DepartureBoard>
  
```


5.4 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 URLs to get more details for the different journeys. The root element is `ArrivalBoard`.

element `Arrival`

<p>diagram</p>	<p>Arrival</p> <p>The element <code>Arrival</code> contains all information about a arrival like <code>time</code>, <code>date</code>, <code>stop/station name</code>, <code>track</code>, <code>realtime time</code>, <code>date</code> and <code>track</code>, <code>origin</code>, <code>name</code> and <code>type</code> of the journey. It also contains a reference to journey details.</p>
<p>children</p>	<p>JourneyDetailRef</p>
<p>used by</p>	<p>element ArrivalBoard</p>

element ArrivalBoard

diagram	
children	Arrival

element JourneyDetailRef

diagram	
used by	element Arrival

Example Response

```

<ArrivalBoard xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="..." servertime="10:30" serverdate="2012-01-25" >
  <Arrival name="PENDELTÅG" type="VAS" stopid="9022014008000001" stop="Göteborg C, Göteborg"
time="14:42" date="2011-09-16" journeyid="9015014131103553" origin="Alingsåsterminalen, Alingsås"
track="1" fgColor="#003273" bgColor="#ffffff" stroke="Solid">
    <JourneyDetailRef ref="http://..." />
  </Arrival>
  <Arrival name="PENDELTÅG" type="VAS" stopid="9022014008000014" stop="Göteborg C, Göteborg"
time="14:42" date="2011-09-16" journeyid="9015014132103056" origin="Kungsbacka station,
Kungsbacka" track="14" fgColor="#003273" bgColor="#ffffff" stroke="Solid">
    <JourneyDetailRef ref="http://..." />
  </Arrival>
  <Arrival name="VÄSTTÅGEN" type="VAS" stopid="9022014008000001" stop="Göteborg C, Göteborg"
time="14:57" date="2011-09-16" journeyid="9015014160208829" origin="Skövde Resecentrum Tåg,
Skövde" track="1" fgColor="#003273" bgColor="#ffffff" stroke="Solid">
    <JourneyDetailRef ref="http://..." />
  </Arrival>
<!-- ... -->
</ArrivalBoard/>

```

5.5 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 realtime 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.

Schema **hafasRestJourneyDetail.xsd**

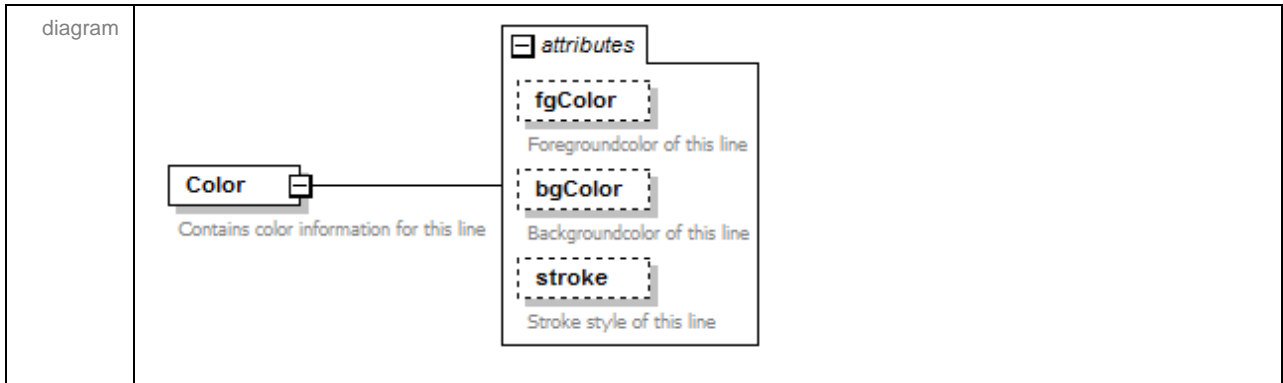
element **GeometryRef**

<p>diagram</p>	<p>GeometryRef Reference to polyline of this leg.</p> <p>attributes</p> <p>ref Contains a URL to call the ReST interface for geometry of this journey</p>
<p>used by</p>	<p>element JourneyDetail</p>

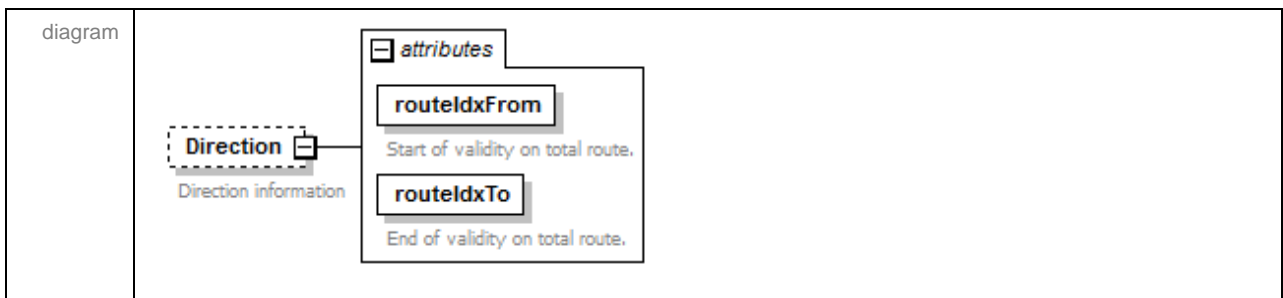
element **JourneyDetail**

<p>diagram</p>	<p>JourneyDetail The journey details contain a list of stops/stations and notes. They also contain the journeys names and types.</p> <ul style="list-style-type: none"> Stop (2..∞) The element Stop contains the name of the stop/station, the route index, the latitude, the longitude, the departure time and date, the arrival time and date, the track, the realtime departure time and date, the realtime arrival time and date and the realtime track. GeometryRef Reference to polyline of this leg. JourneyName (1..∞) Contains name of journey. JourneyType (1..∞) Contains type of journey. Color Contains color information for this line JourneyId (1..∞) Contains the id of the journey. Note (0..∞) Contains a text with notes to be displayed for this leg. Direction Direction information
<p>children</p>	<p>Stop GeometryRef JourneyName JourneyType Color JourneyId Note Direction</p>

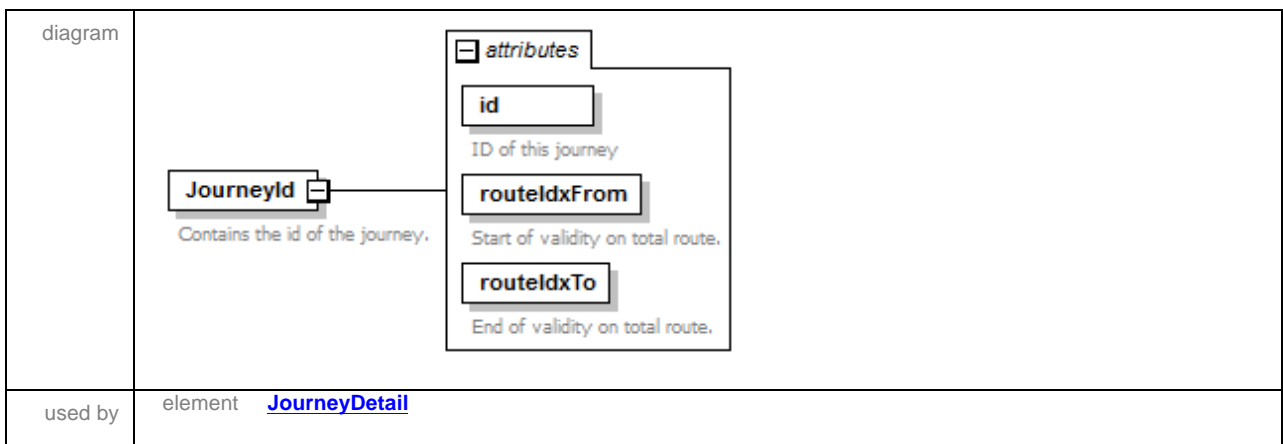
element **JourneyDetail/Color**



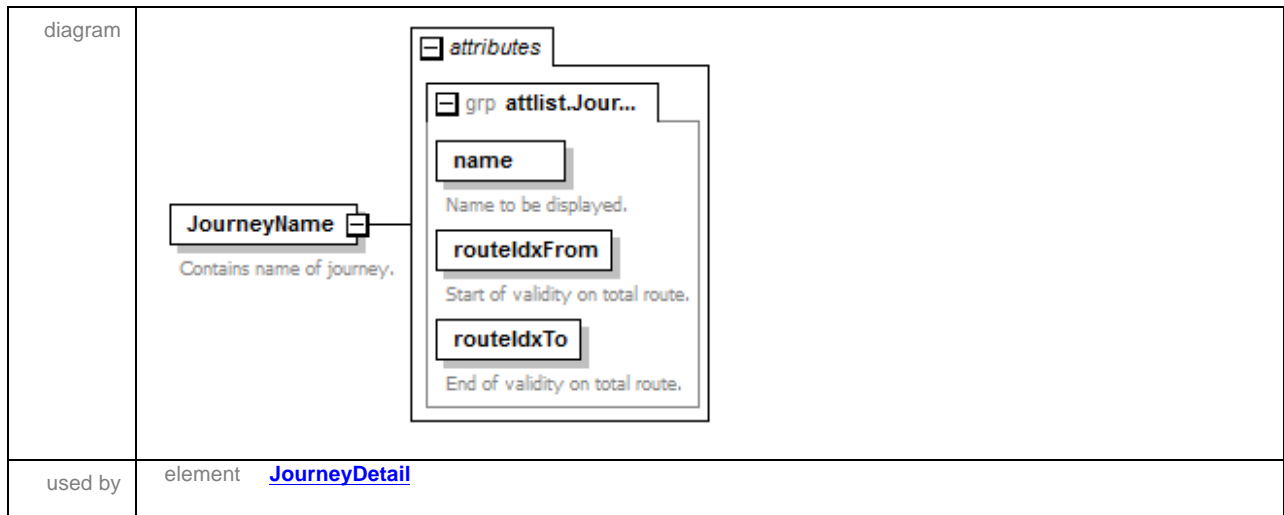
element **JourneyDetail/Direction**



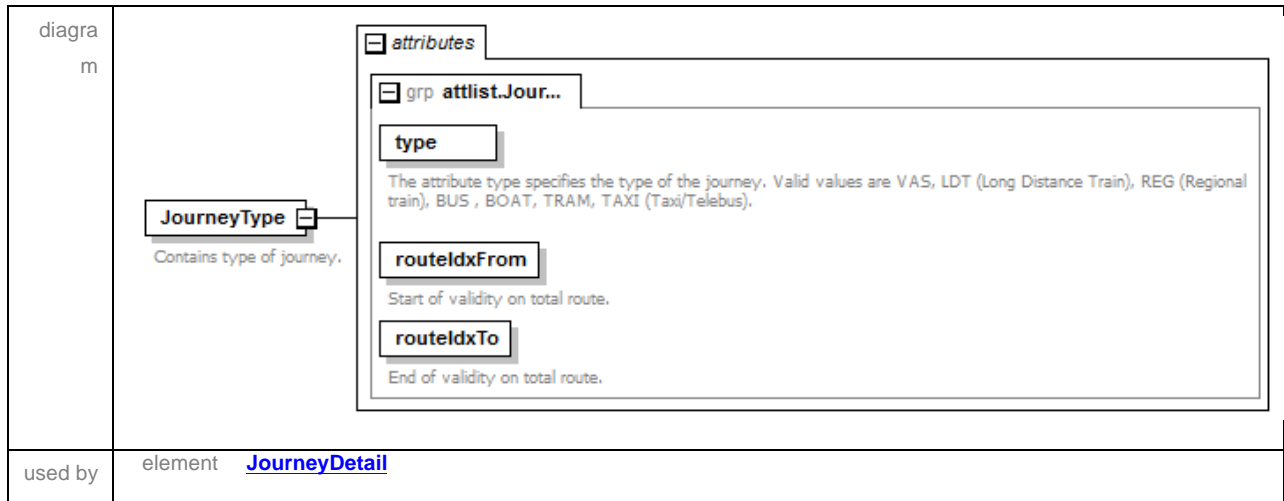
element **JourneyId**



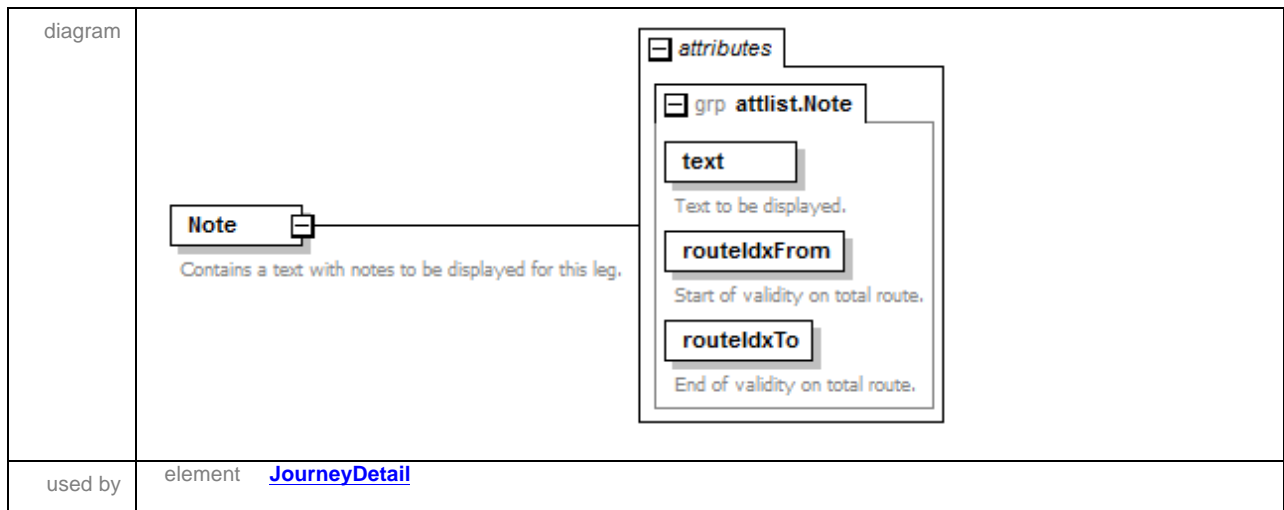
element **JourneyName**




element **JourneyType**



element **Note**



element **Stop**

<p>diagram</p>  <p>The element Stop contains the name of the stop/station, the route index, the latitude, the longitude, the departure time and date, the arrival time and date, the track, the realtime departure time and date, the realtime arrival time and date and the realtime track.</p>	<div data-bbox="906 344 1433 1272"> <p>attributes</p> <p>grp attlist.Stop</p> <p>name Contains the name of the stop/station.</p> <p>routeIdx Route index of a stop/station. Can be used as a reference of the stop/station in a journeyDetail response.</p> <p>lon The WGS84 longitude</p> <p>lat The WGS84 latitude</p> <p>depTime Departure time in format HH:MM, if available.</p> <p>depDate Departure date in format YYYY-MM-DD, if available.</p> <p>arrTime Arrival time in format HH:MM, if available.</p> <p>arrDate Arrival date in format YYYY-MM-DD, if available.</p> <p>track Track information, if available.</p> <p>rtDepTime Realtime departure time in format HH:MM if available.</p> <p>rtDepDate Realtime departure date in format YYYY-MM-DD, if available.</p> <p>rtArrTime Realtime arrival time in format HH:MM if available.</p> <p>rtArrDate Realtime arrival date in format YYYY-MM-DD, if available.</p> <p>rtTrack Realtime track information, if available.</p> </div>
<p>used by</p>	<p>element JourneyDetail</p>

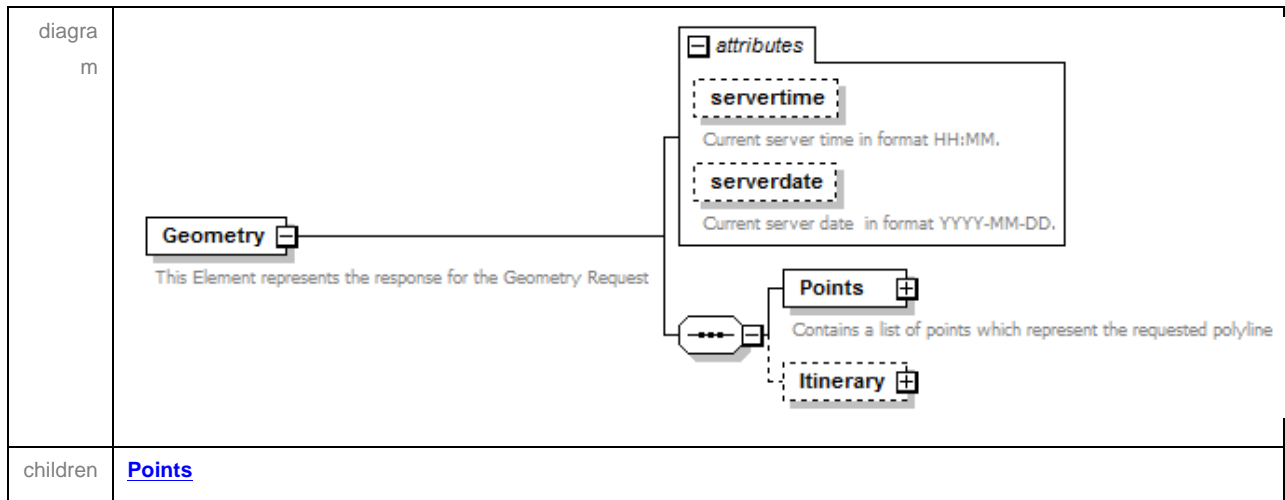
Example Response

```
<?xml version="1.0" encoding="iso-8859-1"?>
<JourneyDetail xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="http://..." servertime="10:30" serverdate="2012-01-25" >
  <Stop name="Alingsåsterminalen, Alingsås" id="9022014017767013" lon="12.529866" lat="57.926991"
routeIdx="0" depTime="14:03" depDate="2011-09-16" track="0" />
</... ->
  <Stop name="Sävenås Station, Göteborg" id="9022014006605001" lon="12.025337" lat="57.724734"
routeIdx="10" arrTime="14:36" arrDate="2011-09-16" depTime="14:36" depDate="2011-09-16" track="1"
/>
  <Stop name="Göteborg C, Göteborg" id="9022014008000001" lon="11.974503" lat="57.708859"
routeIdx="11" arrTime="14:42" arrDate="2011-09-16" track="1" />
  <Color fgColor="#003273" bgColor="#ffffff" stroke="Solid" />
  <GeometryRef ref="http://..." />
  <JourneyName name="Lok TÅG" routeIdxFrom="0" routeIdxTo="11" />
  <JourneyType type="VAS" routeIdxFrom="0" routeIdxTo="11" />
  <JourneyId id="9015014131103553" routeIdxFrom="0" routeIdxTo="11" />
  <Direction routeIdxFrom="0" routeIdxTo="11">Göteborg</Direction>
</JourneyDetail>
```

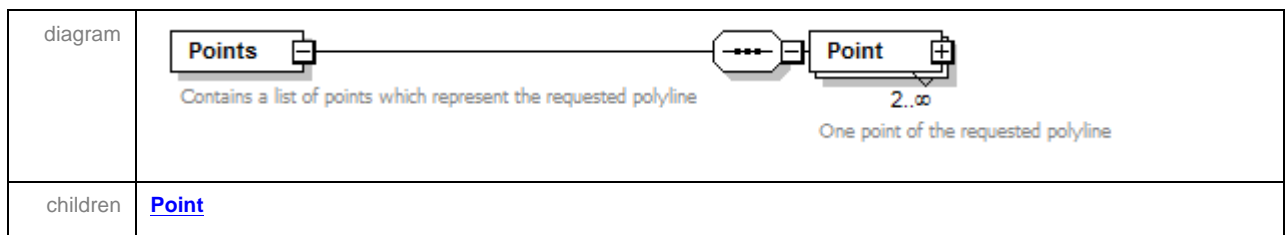
5.6 Geometry Response

The Geometry service will return the polyline for a leg. The result contains a list of WGS84 coordinates which can be used to display the polyline on a map.

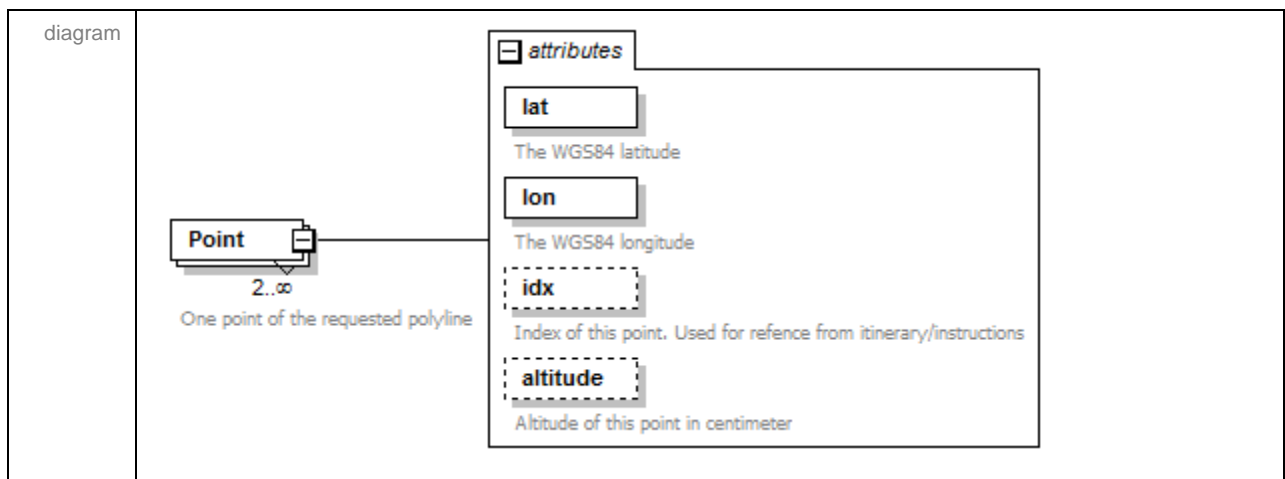
element **Geometry**



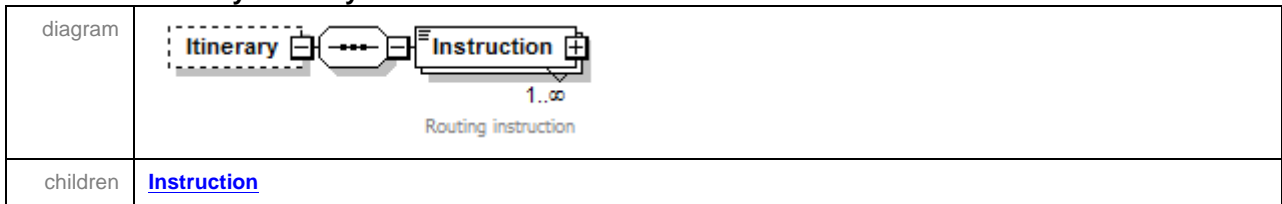
element **Geometry/Points**



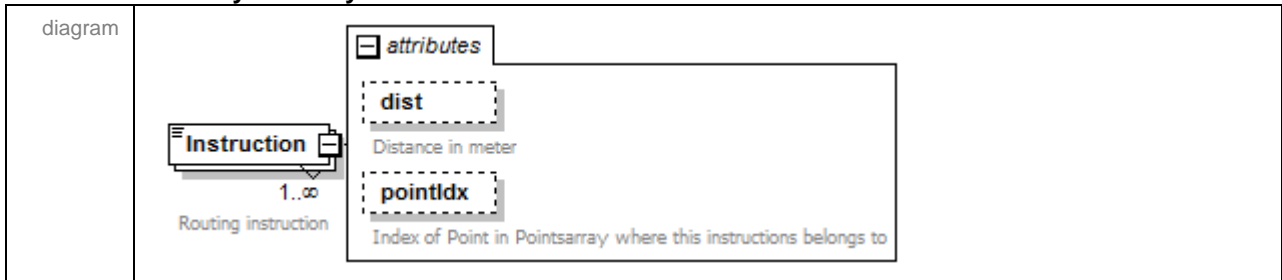
element **Geometry/Points/Point**



element **Geometry/Itinerary**



element **Geometry/Itinerary/Instruction**



Example Response

```
<?xml version="1.0" encoding="iso-8859-1"?>
<Geometry xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="http://..." servertime="10:30" serverdate="2012-01-25" >
  <Points>
    <Point lat="57.795569" lon="12.049248" />
    <Point lat="57.795524" lon="12.049212" />
    ...
    <Point lat="57.680543" lon="11.901744" />
    <Point lat="57.680534" lon="11.901708" />
  </Points>
</Geometry>
```


5.7 SystemInfo response

The SystemInfo response will contain information about the travelplanner system and the underlying data.

element SystemInfo

diagram	<p>The diagram shows a box labeled 'SystemInfo' with a small square icon on its right side. A line connects it to a larger box labeled 'TimetableInfo' with a small square icon on its right side. A dashed line with three dots is positioned between the two boxes, indicating a containment relationship. Below 'SystemInfo' is the text 'Routelement for systeminfo result. Contains information about the system and timetable data'. Below 'TimetableInfo' is the text 'Contains information about the timetable data'.</p>
children	TimetableInfo

element SystemInfo/TimetableInfo

diagram	<p>The diagram shows a box labeled 'TimetableInfo' with a small square icon on its right side. A line connects it to a larger box labeled 'TimeTablePeriod' with a small square icon on its right side. A dashed line with three dots is positioned between them. From the right side of 'TimeTablePeriod', two lines branch out to two boxes: 'TimeTablePeriod' and 'TimeTableData', both with small square icons on their right sides. Below 'TimeTablePeriod' is the text 'Contains information about the timetable period'. Below 'TimeTableData' is the text 'Contains information about the timetable data'.</p>
children	TimeTablePeriod TimeTableData


element SystemInfo/TimetableInfo/TimeTablePeriod

diagram	<p>The diagram shows a box labeled 'TimeTablePeriod' with a small square icon on its right side. A line connects it to a larger box labeled 'DateBegin' with a small square icon on its right side. A dashed line with three dots is positioned between them. From the right side of 'DateBegin', two lines branch out to two boxes: 'DateBegin' and 'DateEnd', both with small square icons on their right sides. Below 'DateBegin' is the text 'Begin of timetable period in format YYYY-MM-DD'. Below 'DateEnd' is the text 'End of timetable period in format YYYY-MM-DD'.</p>
children	DateBegin DateEnd


element SystemInfo/TimetableInfo/TimeTablePeriod/DateBegin

diagram	<p>The diagram shows a box labeled 'DateBegin' with a small square icon on its right side. Below it is the text 'Begin of timetable period in format YYYY-MM-DD'.</p>
---------	---


element **SystemInfo/TimetableInfo/TimeTablePeriod/DateEnd**

diagram	 <p>End of timetable period in format YYYY-MM-DD</p>
---------	---

element **SystemInfo/TimetableInfo/TimeTableData**

diagram	 <p>Contains information about the timetable data</p> <p>Creation date of timetable data in format YYYY-MM-DD</p>
children	CreationDate

element **SystemInfo/TimetableInfo/TimeTableData/CreationDate**

diagram	 <p>Creation date of timetable data in format YYYY-MM-DD</p>
---------	---

Example Response

```
<?xml version="1.0" encoding="iso-8859-1"?>
<SystemInfo xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="http://.../hafasRestSysteminfo.xsd">
  <TimetableInfo>
    <TimeTablePeriod>
      <DateBegin>2011-12-11</DateBegin>
      <DateEnd>2012-06-30</DateEnd>
    </TimeTablePeriod>
    <TimeTableData>
      <CreationDate>2012-05-16</CreationDate>
    </TimeTableData>
  </TimetableInfo>
</SystemInfo>
```

6 Error Codes

The following table lists all the possible error codes and corresponding explanatory error messages that can be returned by the service, in the attributes `error` and `errorText`, respectively.

General ReST Request Errors	
Error code	Error message
R0001	Unknown service
R0002	Invalid or missing request parameters
R0007	Internal communication error
Backend Server Errors	
S1	The desired connection to the server could not be established or was not stable.
Connection Request Errors	
H9380	Dep./Arr./Intermed defined more than once
H9360	Error in date field
H9320	The input is incorrect or incomplete
H9300	Unknown arrival station
H9280	Unknown intermediate station
H9260	Unknown departure station
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	No connections found
H895	Departure/Arrival are too near