RFC90 - Navigation Domain Model

Confidential, Draft

85 Pages

Abstract

This document specifies the navigation domain model: the key navigation entities.

0 Document Information

0.1 Table of Contents

All Page Within This Box

All Page Within This Box

0.2 Status

This document specifies the navigation domain model for the Open Services Gateway Initiative, and requests discussion and suggestions for improvements. Distribution of this document is unlimited within OSGi.

0.3 Acknowledgement

0.4 Terminology and Document Conventions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY" and "OPTIONAL" in this document are to be interpreted as described in [1].

```
Source code is shown in this typeface.
```

0.5 Revision History

The last named individual in this history is currently responsible for this document.

All Page Within This Box

Revision	Date	Comments
0.1	June 24, 2004	Document creation
0.2	August 18, 2004	Updates after Schaumburg meeting. • Add a new proposal for Address, Coordinate, Geometry, Location and Map definitions in Considered Alternatives section; • Add Siemens VDO Automotive technical solution for Route and Maneuver class in Considered Alternatives section
0.3	Sept 24, 2004	Updates after Amsterdam meeting: • Move alternative definition of Coordinate to section 5; • Move alternative definition of Map, POI and Position to section 5 and remove of alternatives; • Change the alternative definition of an Address.
0.4	November 24, 2004	Updates after Sophia Antipolis meeting: • Move Address, Geometry, Location to section 5. • Add Javadoc
0.5	June 3, 2005	Add some open issues Reorganize the document
0.6	Jan 4, 2006	Updates after Boeblingen meeting: • Add MOST definition for address keys; • Remove Polygon definition and rename Shape to Zone.
1.0	March 6, 2006	Updates after Sophia Antipolis Meeting: • Move Route and Maneuver classes to section 5; • Route, Location updates

All Page Within This Box

1.1	April 3, 2006	Updates after Eindhoven Meeting: • Remove constructors from Location and integrate this to a Navigation Service interface; • ManeuverGeometry and ManeuverSegment changed to JunctionGeometry and JunctionSegment; • RouteElement link removed; • DirectionOfTurn removed and add turn angle added; • NO_ACTION and CONTINUE values added to ActionType; • Map entity is removed from the model for the moment; • Introduction of Navigation Service interface; • Use Map instead of Dictionary in the RFC; • POI category should use '/' format to express hierarchy in categories.
1.2	April 14, 2006	• Add locale to POI search; • Remove getHints() from Route interface; • Add ZONE_FIRST constant to Location interface

All Page Within This Box

1 Introduction

This document is a very first draft of the navigation domain model. The goal is to describe the key entities of the domain model.

It is an API proposal for entities defined in the "RFP 37 – Navigation Domain Model".

Later these entities can be used to define some higher level API like Navigation Service.

2 Application Domain

2.1 Navigation domain

In order to explain the navigation domain we must define some terms.

The navigation domain concerned all information and services used in the vehicle to guide, to inform or to go with the driver.

The navigation domain model is the set of entities that defined all exchangeable data within the navigation domain.

As explained previously some navigation entities can be exchanged between different applications developed by different vendors that is why we have to come to a standard definition of this key entities.

Actors that will use the NDM are:

- Car OEM

  Car OEM can select or develop appropriate services for in-vehicle use.

- Car supplier

  Car supplier can easily integrate third party service or content provider in order to offer broader services.

- Service provider

  Services developed can be used and integrated by various car OEM and/or car supplier.

By defining such a model actors can:

- Reduce R&D costs
- Allow re-use or use of existing services
- Reduce development time frame
- Facilitate integration phase

All Page Within This Box

2.2 Entities

Address

An Address is a human-understandable symbolic reference of a location. Several different kinds of address reference systems exist, even in the same area (e.g., postal addresses). An address consists of a street address (or intersection), an ordered tree of place names (e.g. country, municipality, etc), postal code and house number. Address referencing systems tend to be unambiguous only in local areas. Multiple addresses can be associated with the same location.

Advice

An Advice is a message presented to the driver. An advice may be visual (e.g. icons representing the next turn), textual (e.g. next road name) or acoustical (e.g. speech messages). It must be presented at a well-defined point of time. An advice can be a decision advice related to a maneuver or a non-decision advice (e.g. traffic information). In the first case it represents the next action the driver has to perform in order to stay on the planned route.

Destination

A Destination is a location where the user wants to go and stop. Intermediate destinations are called via-points.

Itinerary

An Itinerary is a planned route containing at least the list of maneuvers to reach the destinations.

Journey

A Journey is a set of route that shares the same destination, list of via-points and list of way-points.

Location

A Location is a reference to a point on the surface of the earth (geographic reference: latitude, longitude and altitude).

Maneuver

A Maneuver is the action that the driver has to perform at a location (e.g. junctions). Several advices can be presented for the same maneuver.

Map

A Map is a topographical representation of an area (e.g. parts of a country) defined by roads, land use (e.g. forests, lakes …) and locations (e.g. POI).

A set of data that allows the generation of a graphical representation of a geographical area (e.g. roads, land use, locations …)

All Page Within This Box

Point of Interest (POI)

A Point of Interest represents a special location, like a hospital, a gas station, a theatre, a hotel, etc. It consists of a name, a category, a location and a variety of attributes that can be attached to it (e.g. importance, brand, opening hours …).

Position

A Position is the representation of the current location typically based on the WGS84 system.

Route

A Route is the description of the travel that the user wants to do. A route has different representations. Initially the route contains a destination, a list of via-points, a list of way-points with associated route preferences. When a route calculation is performed the route contains, in addition, the route geometry (the full list of intermediate locations) that leads to destinations.

Via-Point

A Via-Point is an intermediate destination.

Waypoint

A Waypoint is a location where the user wants to go nearby or avoid on his route.

All Page Within This Box

2.3 Standard References

The RFC should take advantage of already existing standards such as:

- AMI-C
  AMI-C is a worldwide organization of motor vehicle manufacturers created to facilitate the development, promotion and standardization of automotive multimedia interfaces to motor vehicle communication networks.
  Additional information can be found in AMI-C Website

- OpenLS
  The Open Location Services Initiative (OpenLS) is devoted to the development of interface specifications that facilitate the use of location and other forms of spatial information in the wireless Internet environment. The purpose of the Initiative is to produce open specifications for interoperable location application services that will integrate spatial data and processing resources into telecommunications and Internet services infrastructure.
  Additional information can be found in OGC Website

- MOST
  The MOST Cooperation is based on a partnership of carmakers, set makers, system architects and key component suppliers. Together they define and adopt a common multimedia network protocol and application object model.
  Additional information can be found in MOST Cooperation Website

- ADASIS
  Advance Driver Assistance Systems (ADAS) need to access and use map data, vehicle position, speed, as well as other sensors' data in order to improve the performance of these applications or apply them to new functionalities.
  The purpose of the ADASIS Forum is to:
  Define an open standardized data model and structure to represent map data in the vicinity of the vehicle position (i.e. the electronic horizon), in which map data are delivered by navigation and/or by a general map data server.
  Define open standardized API(s) to enable ADAS applications to access the electronic horizon and position-related data of the vehicle.
  ADASIS is an European Forum, coordinated by ERTICO.
  Additional information can be found in ADASIS / ERTICO Website

3 Problem Description

By defining a NDM, all exchangeable data between systems are clearly described and allow an easy integration of navigation features from various vendors.

All Page Within This Box

All actors already identified in the previous paragraph will benefit from this definition. The NDM needs to be defined in the OSGi context because more and more systems are or will use an OSGi Service Platform in the automotive industry.

3.1 Use Cases

3.1.1 Download POIs (Points Of Interest)

Actors	*Driver*	*In-car User*	*External user*	*Other Systems*	
		X			
Description	Transfer of POIs from a web site to the in-car system.				
Flow of events	1. The use case starts when Emma enters in the car She is on holidays in Bretagne (Brittany) and she would like to find a hotel for tonight 2. She starts the web browser and downloads some **POI** of her favourite hotel brands around her current car location 3. The system displays the **POIs** on the map 4. She selects one of these on the map and asks for some more information 5. The system downloads and displays an HTML page with a nice hotel picture 6. On the web page, she selects the "Phone to" link to make a reservation by phone. 7. She adds the hotel **address** as a further **destination** 8. The use case ends				
Must have devices	*An in-car system with an OSGi Service Platform installed and connected to Internet (e.g. using GSM)*				
May have devices	*N/A*				
Prior conditions	The in-car system and the external service provider must have the same definition of: • A Point Of Interest (POI) • An address • A destination				
Post conditions	The system stored the hotel address as next destination.				
Alternative flow 1	1. … 2. She starts the web browser and requests POI around a nice location not too far from her car 3. …				
Extension points	None				
Notes	None				
Date of creation	10/28/02				
Last modification	02/02/04				
Status	Final				

All Page Within This Box

3.1.2 Download a route

Actors	*Driver*	*In-car User*	*External user*	*Other Systems*	
		X			
Description	Download a route from an external service provider to the in-car system.				
Flow of events	1. The use case starts when Emma enters in the car She is on holidays in Bourgogne (Burgundy) and she would like to visit some famous "domaine" 2. Within her in-car system she starts the web browser and enters Michelin URL 3. She selects and download the **route** of the famous "Route des vins en Bourgogne" 4. She enters as **destination** the **address** of the house she has rent for holidays 5. A message is displayed asking her if she would like to be guided along this **route** 6. The use case ends when the guidance starts				
Must have devices	*An in-car system with an OSGi Service Platform installed and connected to Internet (e.g. GSM)*				
May have devices	*N/A*				
Prior conditions	The in-car system and the external service provider must have the same definition of: • A route • An address • A destination Route downloaded must be validated according to embedded map data (stored on a CD, DVD or else).				
Post conditions	1. A full itinerary is valid from the current car position to the final destination. 2. The guidance is started.				
Alternative flow 1	**Off-board Navigation** 3. … 4. She selects domaine's **addresses** as new **stages**. 5. These **stages** are transferred to a remote system where the best **itinerary** is calculated. The final result is transferred back to the in-car system. 6. …				
Alternative flow 2	**Itinerary with a ferry connection** 1. … 2. She is living in Corsica and she would like to visit some famous domain in Bourgogne 3. …. 6. The in-car system has automatically calculated the itinerary to reach the first part of the itinerary in Bourgogne. It has to take care of the ferry connection 7. …				

All Page Within This Box

Alternative flow 3	**Itinerary with an off-road part** 6. … 7. Emma is off-road (not on a digitized map). The in-car system guides her to proceed to the itinerary. 8. The use case ends
Extension points	None
Notes	The in-car system will guide the driver to the nearest entry point of the route and not the first point.
Date of creation	10/23/02
Last modification	03/15/04
Status	Final

All Page Within This Box

3.1.3 Transfer a map

Actors	*Driver*	*In-car User*	*External user*	*Other Systems*	
		X			
Description	Transfer a map from the in-car system to an external device (e.g. PDA).				
Flow of events	1. The use case starts when Emma enters in the car. She is on holidays and she is visiting Paris for the first time. 2. She would like to have the Paris **map** on her PDA. 3. She selects the Paris **map** on her in-car system and downloads it into her PDA. 4. The map is automatically centered on the current car location. 5. She can perform a scroll; zoom in and out on the complete area of Paris. 6. The use case ends when Emma leaves the car.				
Must have devices	✓ *An in-car system with an OSGi Service Platform installed* ✓ *An external device (e.g. PDA) with a rendering application installed*				
May have devices	*N/A*				
Prior conditions	The in-car system and the external device must have the same definition of: • A map.				
Post conditions	The external device stored the map of Paris.				
Alternative flow 1	1.				
Extension points	None				
Notes	Some geographical concepts must be known in order to draw properly the road network, land uses and facilities.				
Date of creation	10/23/02				
Last modification	12/02/02				
Status	Final				

All Page Within This Box

3.1.4 Scenic Guide

Actors	*Driver*	*In car user*	*External user*	*Other Sub-Systems*	
		X			
Description	Use of a third party scenic route with audible and visual messages in the car.				
Flow of events	1. The use case starts when Emma enters in the car 2. She is on holidays and she would like to visit the beautiful “châteaux de la Loire” 3. She inserts the DVD offered by the tourist office 4. An HTLM page appears in her in-car system proposing to select on of the scenic **route** along castles (**Point Of Interest**) 5. She selects one **route** 6. The in-car system guides her along this **route** 7. While approaching the Chenonceau castle a **visual advice** (e.g. HTLM page) is displayed and an **acoustical advice** is rendered: “*On your left you can discover the beautiful Chateau de Chenonceau...*” 8. The use case ends				
Must have devices	*An in-car system with an OSGi Service Platform installed*				
May have devices	*N/A*				
Prior conditions	The in-car system and the external service provider must have the same definition of: • A Point Of Interest • A route • A visual advice • An acoustical advice				
Post conditions	1. The system uses the downloaded itinerary as current one 2. The audible and visual advices are presented				
Alternative flow 1	4.				
Extension points	None				
Notes	The advice must be presented on a well-defined point of time.				
Date of creation	10/28/02				
Last modification	03/15/04				
Status	Final				

All Page Within This Box

3.1.5 Plan a route

Actors	*Driver*	*In-car User*	*External user*	*Other Systems*	
		X		**X**	
Description	Plan a route with the in-car system.				
Flow of events	1. The use case starts when Emma enters in the car. She is living in Nice; she would like to visit a friend in Lyon and Paris. Last night she has prepared her trip. She has selected in her PDA's address book her friend's **address** in Lyon as an intermediate **destination** and her friend's **address** in Paris as final **destination** of her trip 2. She transfers her selection into the in-car system 3. The system displays a message asking her if she would like to be guided to the address in Lyon first and finally to the address in Paris. By default the route preference is shortest for both destinations 4. She validates the selection and the system calculates the route 5. She reviews the detailed information about the route. The system provides the list of maneuvers where each maneuver may include the distance from the previous maneuver, the name of the road onto which the vehicle turns and an icon indicating the type of maneuver and turn angle. 6. The system provides the route summary information (e.g. total distance of route and estimated time of arrival) 7. She validates the route. 8. The use case ends when the guidance starts.				
Must have devices	✓ *An in-car system with an OSGi Service Platform installed*				
May have devices	✓ *An external device (e.g. PDA) that contains destinations* ✓ *An address book application on the in-car system*				
Prior conditions	The in-car system and external devices must have the same definition of: • Address • Destination (intermediate and final) Addresses must be "navigable". It means that addresses must be validated using embedded map data (stored on CD, DVD or else).				
Post conditions	1. The destinations have been validated as "navigable". 2. Driver receives the first advice				
Alternative flow 1	**Destination entered by using an Intelligent Speller** 1. … 2. She enters her friends addresses by means of an intelligent speller: it uses a database of addresses to suggest letters when typing and/or list of matching address parts (country, city, street names) 3. …				

All Page Within This Box

Alternative flow 2	**Destination entered by using an address book** 1. … 2. She selects her friends addresses from the in-car's address book where she has stored them 3. …
Alternative flow 3	**Destination entered by history list** 1. … 2. She selects her friends addresses from the history list because she has recently used these addresses. The number of destination stored in the history list is customizable. 3. …
Alternative flow 4	**Destination entered by using a POI (Point Of Interest)** 4. … 5. Before arriving in Lyon she wants to have lunch, so she looks up available restaurants in Lyon, and decided to have lunch at the Leon de Lyon before going top her friend. She selects this restaurant as her 1st destination on her journey. 6. …
Alternative flow 5	**Destination entered by using geographic coordinates** 1. … 2. She enters the destination by using the geographic coordinate, which she stored from a last visit. 3. …
Alternative flow 6	**Destination entered via map** 1. … 2. She does not remember where her friends leave exactly and she selects a point on a map 3. She selects a location from a list of relevant locations proposed by the system (near the selected point on the map) 4. …
Alternative flow 7	**Destination entered via voice input** 1. … 2. She speaks the name of the location 3. She selects a location from a list of relevant locations proposed by the system 4. …
Alternative flow 8	**Destination entered via free text** 1. … 2. She enters her friends address in free text. 3. She selects a location from a list of relevant locations proposed by the system. 4. …

All Page Within This Box

Alternative flow 9	**Add of a way point** 4. … 5. Next to her destinations, Emma decided to enter a way-point that will guide the route planning: i.e. she will enter Grenoble as way-point, to make sure that she will be guided to the east of the "Autoroute du Soleil". 6. Note: the route will not necessarily go through Grenoble. 7. …
Alternative flow 10	**Change route preferences** 4. … 5. She decides to change the default route preferences. She has the choice of the following route preferences: • Shortest • Fastest • Avoid highway • Avoid toll • Avoid ferry …
Alternative flow 11	**Make a detour** 7. … 8. She would like to avoid the first part of the route and indicates to the system to avoid the 20 first kilometers. 9. The system calculates a new route and mark as a "black spot" the entered distance for the route calculation 10. She validates the route\ 11. …
Alternative flow 12	**Avoid a road** 7. … 8. She selects several road to avoid from the maneuver list. 9. The system calculates a new route and mark as a "black spot" the road selected 10. She validates the route 11. …
Extension points	None
Notes	None
Date of creation	10/23/02
Last modification	02/02/04
Status	Final

All Page Within This Box

3.1.5 Provide Guidance Information

Actors	*Driver*	*In-car User*	*External user*	*Other Systems*	
		X			
Description	Provide information about the upcoming maneuver				
Flow of events	1. System provides the detailed information for the current/upcoming maneuver, landmark, or waypoint 2. System provides the name of the road on which the vehicle is currently located 3. System provides the name of the city in which the vehicle is located 4. System provides the distance to next maneuver/landmark/waypoint, and the next road name 5. System provides the destination direction 6. System provides the distance to the destination 7. System provides the estimated time to destination 8. Periodically (about once per second), the system provides updated information				
Must have devices	*N/A*				
May have devices	*N/A*				
Prior conditions	User is driving on route				
Post conditions	Guidance information is provided and periodically updated				
Alternative flow 1					
Extension points	None				
Notes	None				
Date of creation	10/10/03				
Last modification	10/10/03				
Status	Final				

3.1.6 Provide Facility Information

Actors	*Driver*	*In-car User*	*External user*	*Other Systems*	
		X			
Description	Provide information about facilities alongside the route				
Flow of events	1. The system maintains a list of facilities (e.g. exit, parking area, service area, …) along the planned route, ordered according to their appearance 2. The list is made available to other applications or presented to the user together with the distance to each facility 3. If a facility is passed by the vehicle, the list is updated				
Must have devices					
May have devices	*N/A*				
Prior conditions	User is driving on route				
Post conditions					
Alternative flow 1					
Extension points	None				
Notes	None				
Date of creation	10/10/03				
Last modification	10/10/03				
Status	Final				

All Page Within This Box

4 Requirements

The NDM must be in line with some general requirements:

- The model should be limited to the key entities.
  We must identify and define the key entities of the model. It cannot be too broad.
- The model must not be linked to any map database references
  The abstraction level of the NDM must be high enough to avoid any link to map databases.
- The model entities must have a limited size
  Devices which will import or export entities from the model can have limited resources. By defining heavyweight entities we will automatically limit the device scope.
- The model should take advantage of already existing standards
  These standards are already mentioned in Application Domain / Look at section.

This RFP is covering the following requirements.

ReqNDM#1 - Download destination

Download an address as a final destination from an external device (e.g. PDA) to the in-vehicle system. This address can be picked up from an address book (e.g. friend's address) or can be retrieved from e-mail (e.g. an appointment for an important meeting).

ReqNDM#2 - Download journey

Download a journey from an external device (e.g. PDA) to the in-vehicle system. A prepared journey is downloaded in the in-vehicle system and used as input for route calculation and then guidance.

ReqNDM#3 - Download route

Download a route from a content provider to the in-vehicle system. The driver can download from a web site a predefined route and use it as an input for guidance given by the in-vehicle system.

ReqNDM#4 - Download POI

Download of POIs from a web site to the in-vehicle system. POIs can be displayed on the in-vehicle map. The POI phone number can be used to make a phone call and the POI address can be used to be a destination.

ReqNDM#5 - Transfer map

Transfer a map from the in-vehicle system to an external device (e.g. PDA). This transferred map can be used for a pedestrian walk within a city.

ReqNDM#6 - Scenic Guide

A third party provider can distribute scenic route containing POIs with advice (audible and/or visual). These advice are rendered as soon as the driver is entering a well-defined zone around POI.

ReqNDM#7 - Plan a route

The system must provide an ordered list of locations (route) including maneuvers to be able to guide the driver to a destination.

ReqNDM#8 - Provide Guidance Information

The system must provide all advice information to follow the planned route.

ReqNDM#9 - Provide Facility Information

The system must provide all relevant information pertaining to facilities.

ReqNDM#10 - Multimodal travel

Travelers can use a complete itinerary while using available modes of transportation such as bicycles, automobiles, walking, ferries, trains and commercial aircraft flights.

ReqNDM#11 - Tank stop

By using a list of gas station offered by a third party provider (POI) the driver can select one of them and add it as an intermediate destination.

ReqNDM#12 - Fleet management

Retrieve navigation information about a fleet such as current destination, current itinerary, and current position.

ReqNDM#13 - Breakdown Call

When a driver is unable to continue his journey because some necessary component has failed a call is sent with the current car position.

ReqNDM#14 - Emergency Call

Send a call with the current car position to an emergency phone number.

ReqNDM#15 - Off-board navigation

Download of a corridor and an itinerary from a remote server into the in-vehicle system in order to be guided along this itinerary. Until the driver stays in this corridor the in-vehicle system does not access anymore the remote server.

4.1 Future requirements

ReqNDM#16 - Import external information

The system should be able to import and use external information. The information could be travel-related information (traffic jams, weather conditions, road work areas) that is broadcast / received in any number of ways (dedicated short-range communication, mobile phone)

ReqNDM#17 - Provide GPS information

The system should be able to provide GPS information such as number of satellites.

ReqNDM#18 - Provide Traffic information

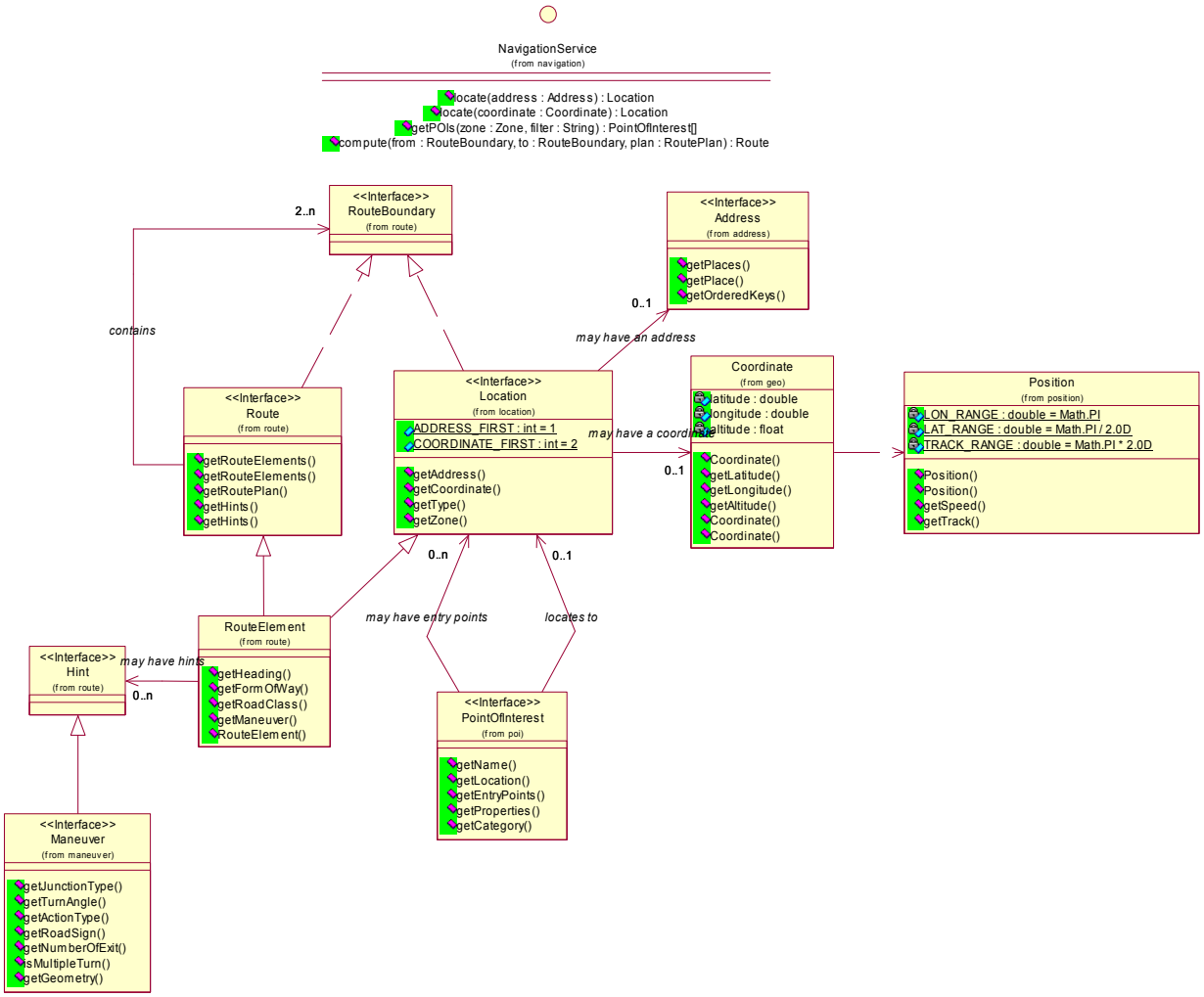
The system should be able to provide Traffic Information independent from the source it comes from (TMC, TPEG, GATS, ….).

ReqNDM#19 - Provide ADAS information

The system should be able to provide ADAS (Advance Driver Assistance System) information to any services that require them. ADAS information is mainly road network in front of the car and the position of the car on this road network.

5 Technical Solution

5.1 Overview



The class diagram below presents the main classes of the Navigation Domain Model and their relationships.

Several important concepts must be highlighted from this diagram:

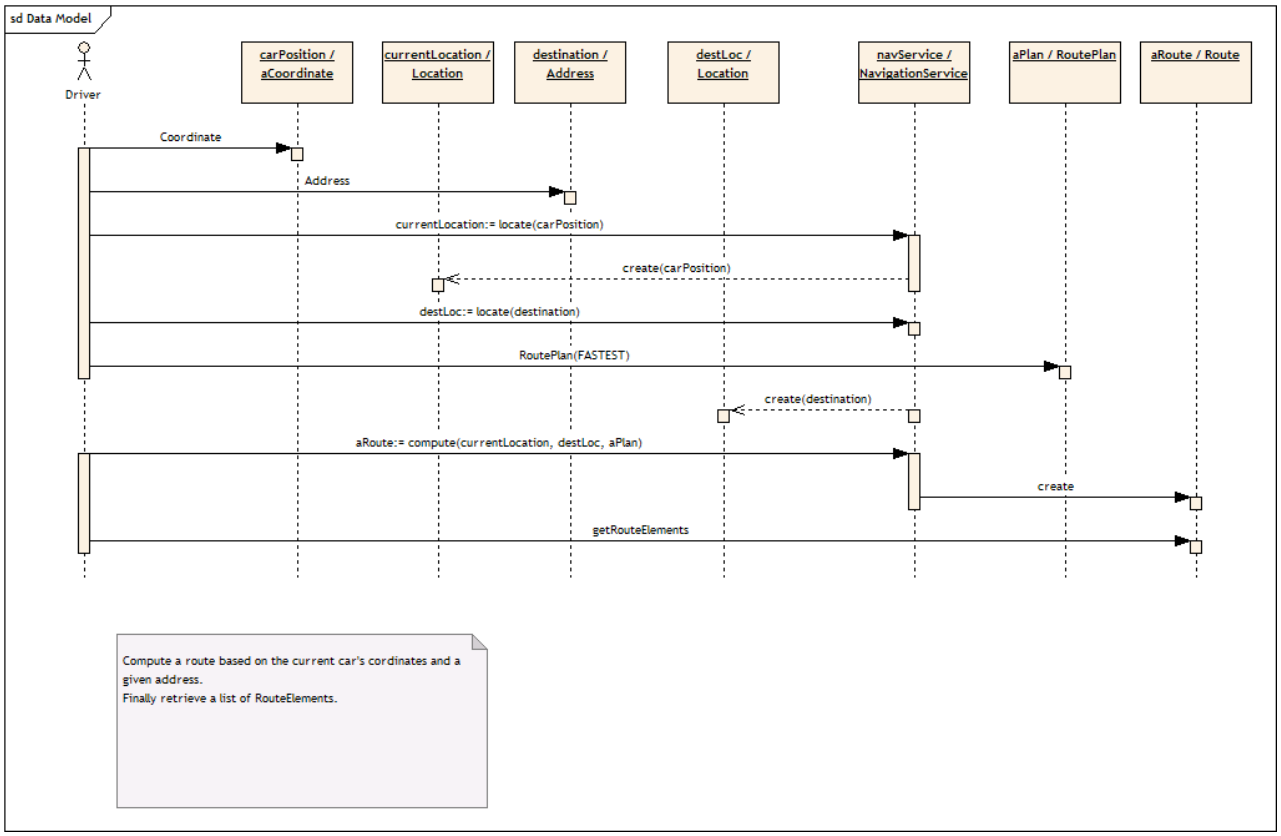
- Most of the navigation classes are linked to the notion of Location. The Location is the central notion within the model
- A Location is composed by a Coordinate and/or an Address

All these concepts will be developed in the following sections.

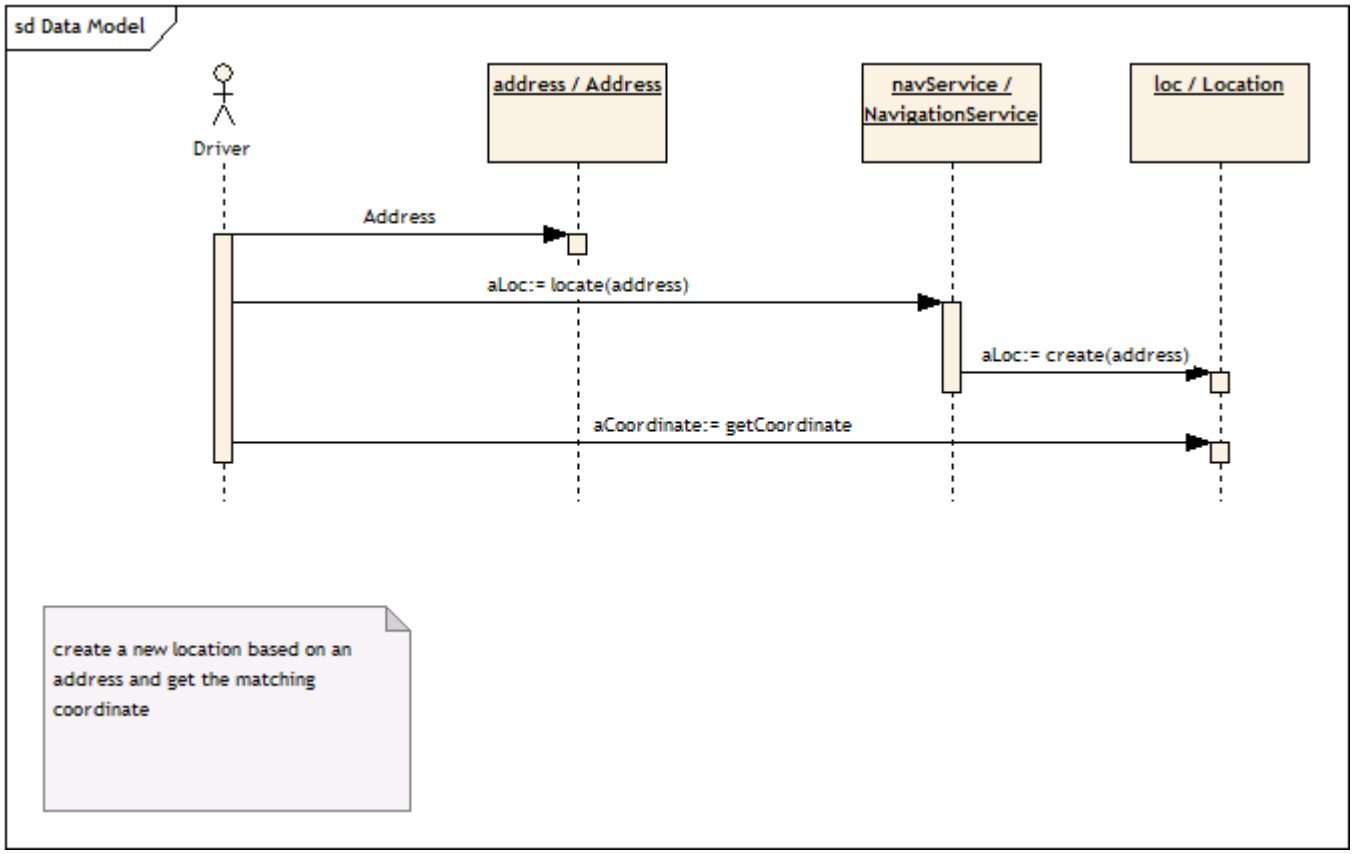
All Page Within This Box

5.2 Use case realizations

5.3 Plan a route



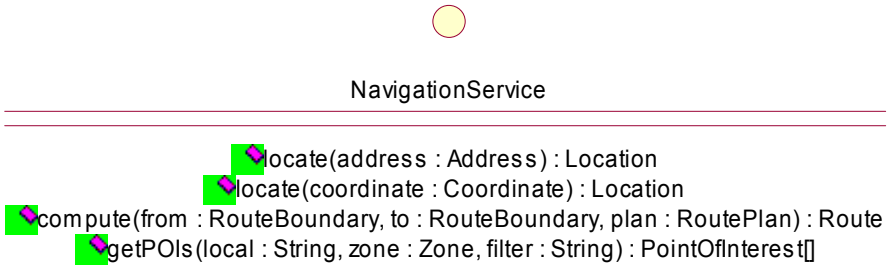
5.4 Resolve an address



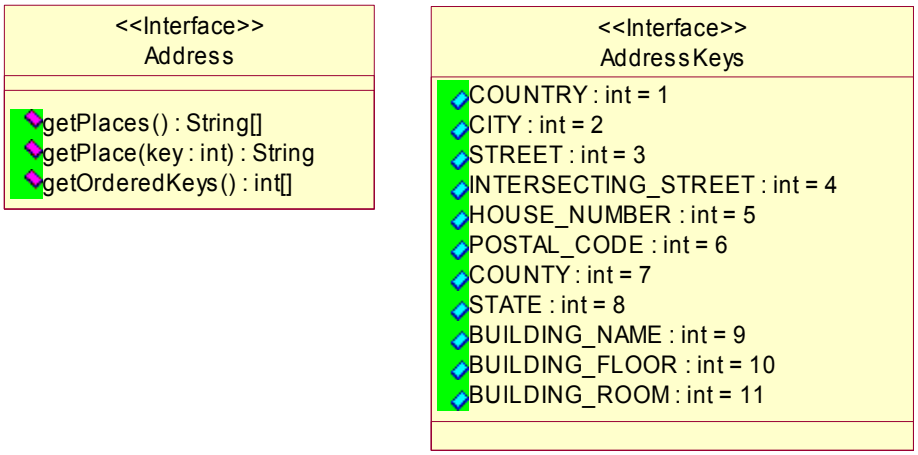
5.5 Get POI information

5.6 Class Diagrams

5.6.1 Navigation Service



5.6.2 Address



The `Address` class is made of a list of places. Each place is represented by a key (one of the integer values contained in `AddressKeys` interface) and a value (a `String` object) which is the place name.

The address format can be retrieved by using the `getOrdreredKeys()` method that returns you an ordered list of keys.

A place name can be retrieved by using:

- `getPlace(int key)` with the key as parameter
- `getPlaces()`, in this case all values are returned at once

The `AddressKeys` interface contains all possible values for address place keys.

For a specific Country (or even within the same country) an ordered list of keys is defined and helps the user to read the address received.

All Page Within This Box

Conversion to VCard format

The address part of the VCard format is a concatenation of 7 fields:

- Post Office Address (first field)
- Extended Address (second field)
- Street (third field)
- Locality (fourth field)
- Region (fifth field)
- Postal Code (six field)
- Country (seventh field)

We can identify the following conversion

VCard Field	Address Key
Post Office Address	Not used
Extended Address	Not used
Street	STREET
Locality	CITY
Region	STATE? COUNTY?
Postal Code	POSTAL_CODE
Country	COUNTRY

MOST Address Definition

Value	Value Definitions
1	Country
2	Town
3	Street
4	Junction
5	Housenumber
6	ZIP Code
7	Building
8	Telephone Number
9	Province
10	District

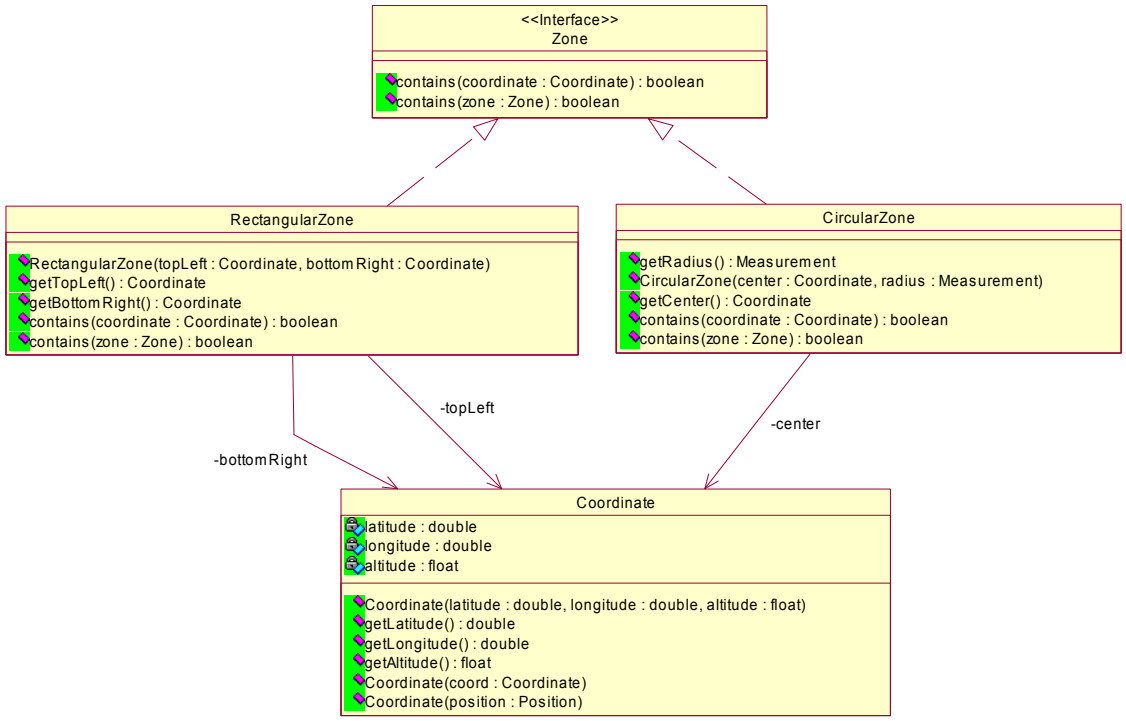
References

- JSR 179 Location API

All Page Within This Box

- VCard Specification V2.1
- MOST Navigation API

5.6.3 Geometry



The Geometry package defines an important entity of the Navigation which is the Coordinate and a set of predefined zones based on this coordinate definition. It is easier to manipulate such objects and perform some operations between different zones and/or coordinates

We have identified two types of useful zones to use:

- RectangularZone defined by two coordinates: the top left and bottom right.
- CircularZone defined by a center and a radius.

The Coordinate class defines a WGS84 coordinate, it contains:

- the latitude value in radians
- the longitude value in radians
- the altitude value in meters

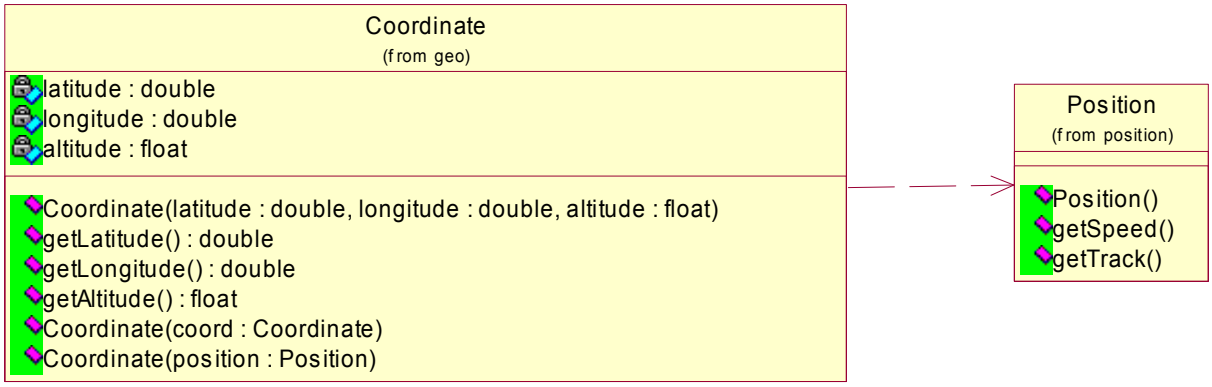
To save memory space the Coordinate definition is not based on `org.osgi.util.measurement` package.

A coordinate can be created:

- by using another coordinate object
- by using a `Position` object
- by using a set of values (latitude, longitude, altitude)

All Page Within This Box

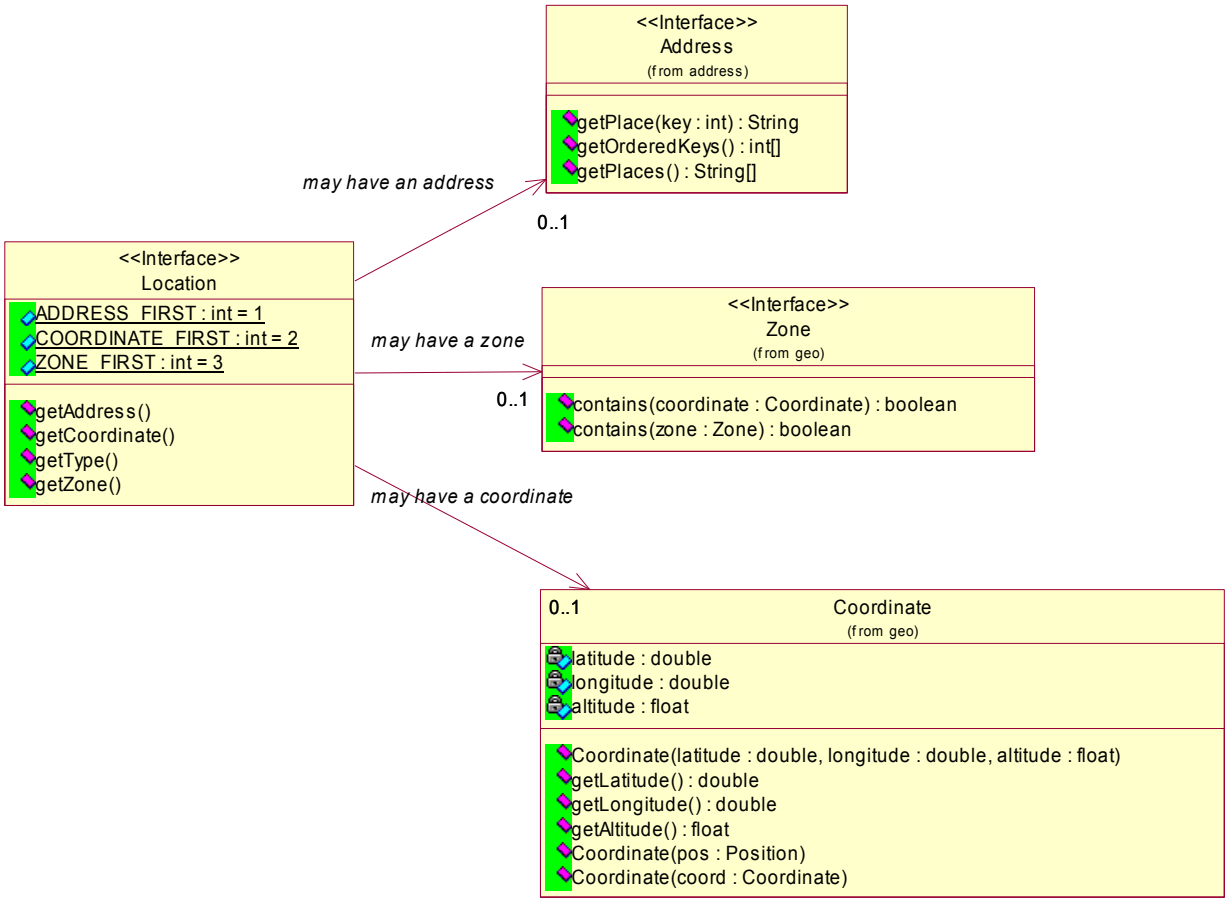
Relationship with org.osgi.util.position.Position



The Position class from the `org.osgi.util.position` package is unchanged but there is a dependency between `Coordinate` and `Position` because we should be able to create a `Coordinate` object from a `Position` object.

All Page Within This Box

5.6.4 Location



The `Location` class as mentioned earlier is a central class in the Navigation Domain Model because most of the classes are location based.

The Location object may contain:

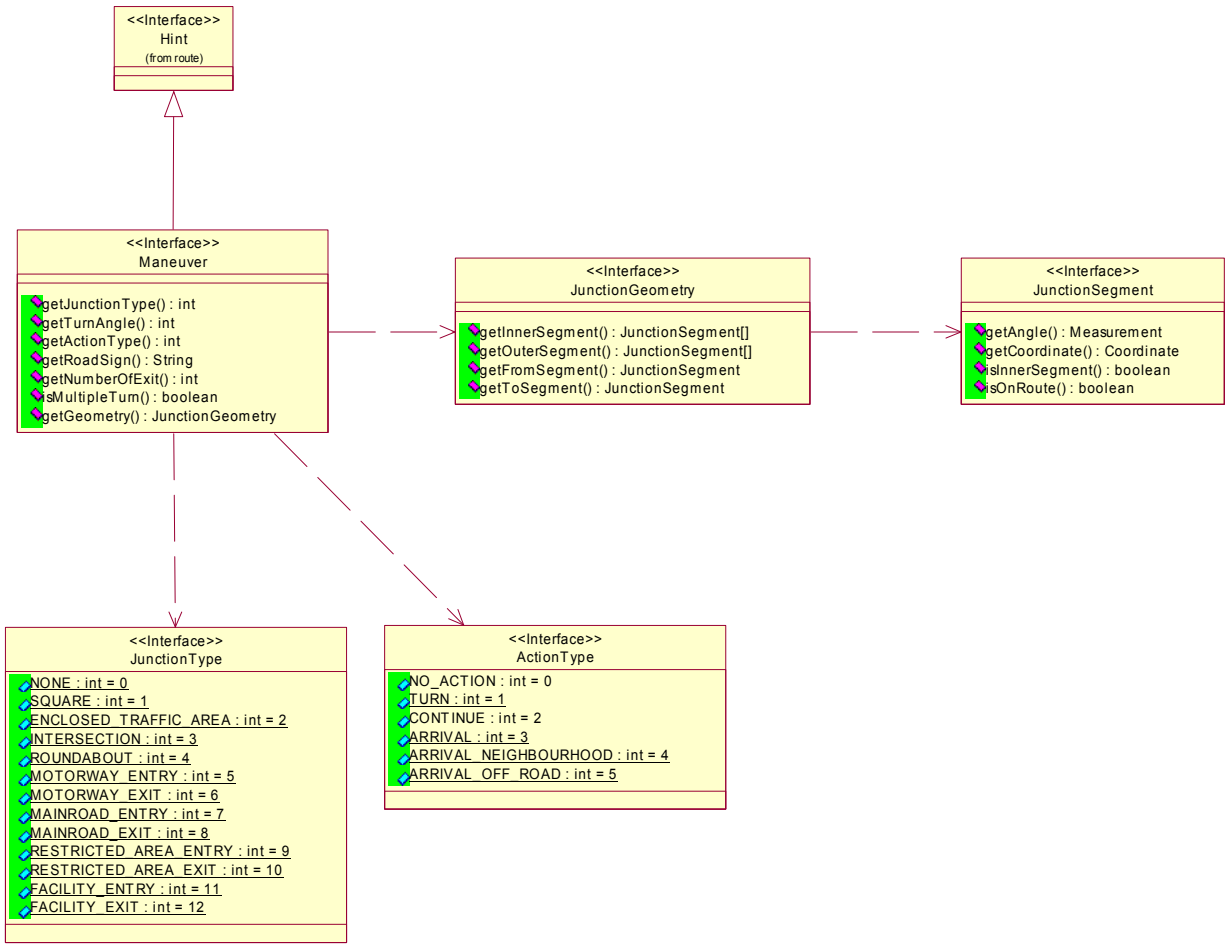
- A `Coordinate` object
- An `Address` object
- A `Zone` object

A `Location` must contain at least one of the following values: an address, a coordinate or a shape.

There is as well an additional attribute called Type that indicates if the location has been created by entering an address (`ADDRESS_FIRST`) or the coordinate (`COORDINATE_FIRST`). This information is important when the address has to be validated against an internal address database in order to be used as a destination.

All Page Within This Box

5.6.5 Maneuver



A `Maneuver` object describes the action that the traveler has to perform at a junction.

The important points in this package are to clearly define:

- A set of action performed;
- A set of junction type;
- A `JunctionGeometry`.

The `JunctionGeometry` object is define by a list of `JunctionSegment` of four types:

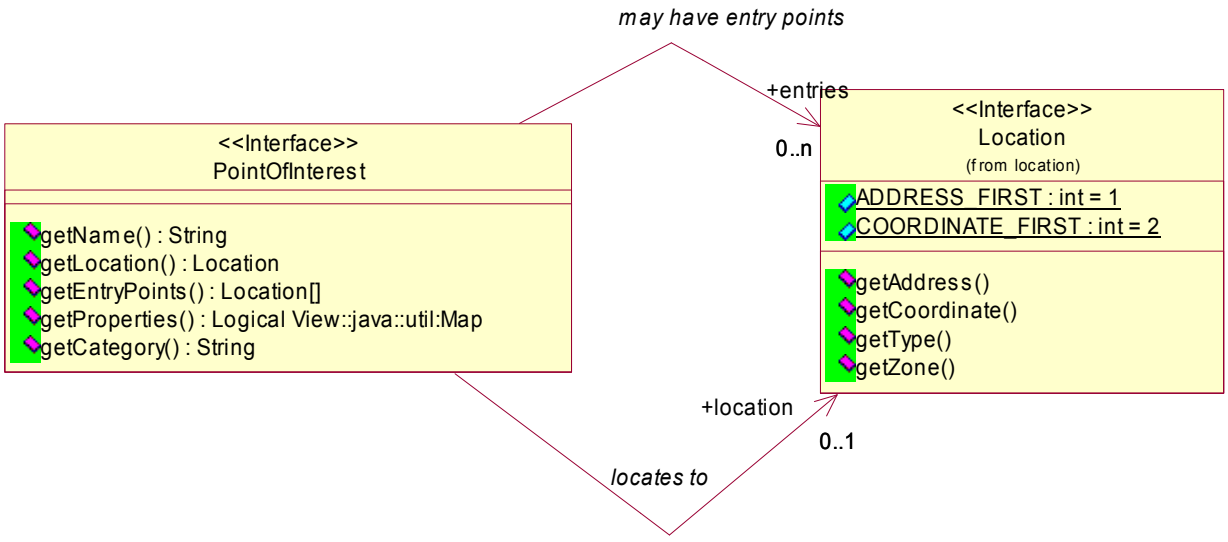
- `getToSegment()` returns the segment where the traveler is leaving the junction
- `getFromSegment()` returns the segment from where the traveler is entering the junction

All Page Within This Box

- `getInnerSegment()` defines the inside geometry of the junction. The *inner segments* are the segments that are connected at both ends with another segment. If all outer segments of a junction are connected to each other in one point, there are no inner segments.

- `getOuterSegment()` defines the outside geometry of the junction. The *outer segments* are the segments by which the junction can be entered or left.

5.6.6 POI



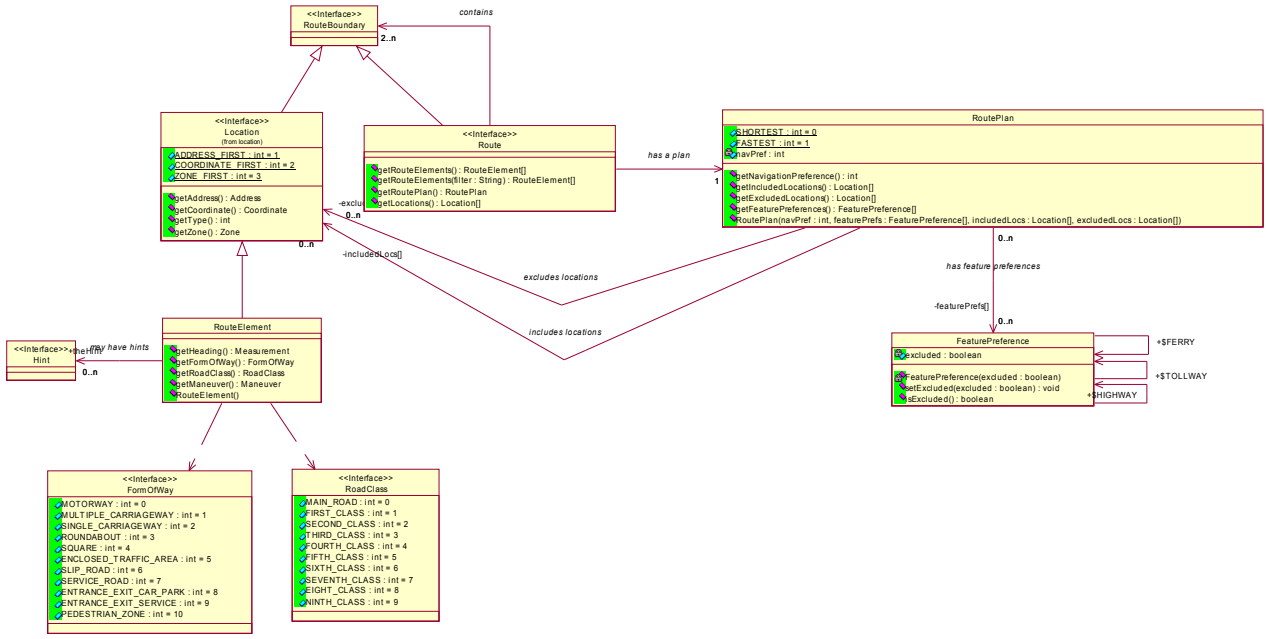
A `PointOfInterest` object contains the following information:

- A name
- A location. The exact location of the POI as delivered by the POI provider.
- A list of entry points. They are locations of all possible entry points to the POI.

A Point of Interest can contain other specific information that is delivered in a dictionary. This information can be stored in a XML file.

All Page Within This Box

5.6.7 Route



The `Route` can be represented by two `Location` objects (e.g. the current car position and the destination) or by a list of `RouteElement` objects that formed the complete description of the route after computation. The concept is the same it is the same route in different states (before computation and after).

This concept allow to an easily compute a route:

- From one location to another (normal case)
- From one location to a predefined route (e.g. join a scenic route)
- From one predefined route to another (e.g. join two scenic routes)
- From our car to another car (moving destination)

A set of preferences can be set to influence the route computation.

These preferences are contained in a `RoutePlan` object. There are three kinds of preferences:

- Directly contained in the `RoutePlan`
- A `FeaturePreference` that defines feature that the traveler would like to include or exclude
- An `AreaPreference` that defines areas that the traveler would like to include or exclude

A `RoutePlan` object is linked to a route.

All Page Within This Box

As soon as the driver leaves the route computed, the route is not valid anymore and another route must be computed.

6 API Specification

6.1 Navigation packages

Packages	
org.osgi.nursery.service.navigation	Contains the interface to the Navigation Service.
org.osgi.nursery.util.address	Contains the address definition.
org.osgi.nursery.util.geo	Contains geodetic utility classes.
org.osgi.nursery.util.location	Contains the location definition.
org.osgi.nursery.util.maneuver	Contains the Maneuver definition.
org.osgi.nursery.util.poi	Contains the Point of Interest definition.
org.osgi.nursery.util.route	Contains the Route definition.

6.2 Package org.osgi.nursery.service.navigation

6.2.1 Interface NavigationService

Method Summary	
Route	**compute**(RouteBoundary from, RouteBoundary to, RoutePlan plan) Computes a route between two Route boundaries (from and to) with the given route plan.
PointOfInterest[]	**getPOIs**(java.lang.String local, Zone zone, java.lang.String filter) Returns a list of POI (Point Of Interest) in a certain zone.
Location	**locate**(Address address) Returns a Location object relative to the Address given as parameter.
Location	**locate**(Coordinate coordinate) Returns a Location object relative to the coordinate given as parameter.

All Page Within This Box

Method Detail

locate

public Location **locate**(Address address)

Returns a Location object relative to the Address given as parameter.

Parameters:

address - The textual address where the user wants to go.

Returns:

org.osgi.nursery.util.location.Location

---

locate

public Location **locate**(Coordinate coordinate)

Returns a Location object relative to the coordinate given as parameter.

Parameters:

coordinate - The WGS84 coordinate where the user wants to go.

Returns:

org.osgi.nursery.util.location.Location

---

getPOIs

public PointOfInterest[] **getPOIs**(java.lang.String local,

Zone zone,

java.lang.String filter)

Returns a list of POI (Point Of Interest) in a certain zone. The user can filter the number of POIs by using their categories.

Parameters:

local - The locale used to retreive POI information

zone - The zone of the search.

filter - The filter to apply to the search.

Returns:

org.osgi.nursery.util.poi.PointOfInterest[]

---

compute

public Route **compute**(RouteBoundary from,

RouteBoundary to,

RoutePlan plan)

Computes a route between two Route boundaries (from and to) with the given route plan.

Parameters:

from - The source.

to - The destination.

All Page Within This Box

plan - The route plan that contains criteria for route computation.

Returns:

org.osgi.nursery.util.route.Route

6.3 Package org.osgi.nursery.util.address

Interface Summary

Address	The Address is made of a list of places.
AddressKeys	AddressKeys interface defines all possible values used as key for a place level.

6.3.1 Interface Address

public interface **Address**

The Address is made of a list of places. Each place is represented by a key (one of the integer values contained in AddressKeys interface) and a value (a String object) which is the place name. The address format can be retrieved by using the getOrdreredKeys() method that returns you an ordered list of keys. A place name can be retrieved by using: • getPlace(int key) with the place key as parameter • getPlaces(), in this case all values are returned at once The AddressKeys interface contains all possible values for address place keys. For a specific Country (or even within the same country) an ordered list of keys is defined and helps the user to read the address received.

Method Summary

int[]	**getOrderedKeys**() Returns the ordered list of keys used to define the address.
java.lang.String	**getPlace**(int key) Returns the place name of a specified place level.
java.lang.String[]	**getPlaces**() Returns the list of place names that define the address.

Method Detail

getPlaces

public java.lang.String[] **getPlaces**()

Returns the list of place names that define the address.

Returns:

All Page Within This Box

List of place names

---

getPlace

public java.lang.String **getPlace**(int key)

Returns the place name of a specified place level. The key must be one the values contained in AddressKey interface.

Parameters:

key - The key of the requested place name

Returns:

String

---

getOrderedKeys

public int[] **getOrderedKeys**()

Returns the ordered list of keys used to define the address. Each key represents a place level.

Returns:

The ordered list of keys

6.3.2 Interface AddressKeys

---

public interface **AddressKeys**

AddressKeys interface defines all possible values used as key for a place level. All these must allow the definition of any type of postal address. The address format is defined by a list of ordered keys made of these predefined values.

Field Summary

static int	**BUILDING_FLOOR** Address key referencing the building floor.
static int	**BUILDING_NAME** Address key referencing the building name.
static int	**BUILDING_ROOM** Address key referencing the building room.
static int	**CITY** Address key referencing the city name.
static int	**COUNTRY** Address key referencing the country name.
static int	**COUNTY** Address key referencing the county.
static int	**HOUSE_NUMBER** Address key referencing the house number.

All Page Within This Box

static int	**INTERSECTING_STREET** Address key referencing the name of an intersecting street.
static int	**POSTAL_CODE** Address key referencing the postal code.
static int	**STATE** Address key referencing the state.
static int	**STREET** Address key referencing the street name.

Field Detail

COUNTRY

public static final int **COUNTRY**

Address key referencing the country name.
See Also:
Constant Field Values

CITY

public static final int **CITY**

Address key referencing the city name.
See Also:
Constant Field Values

STREET

public static final int **STREET**

Address key referencing the street name.
See Also:
Constant Field Values

INTERSECTING_STREET

public static final int **INTERSECTING_STREET**

Address key referencing the name of an intersecting street.
See Also:
Constant Field Values

HOUSE_NUMBER

public static final int **HOUSE_NUMBER**

Address key referencing the house number.
See Also:
Constant Field Values

POSTAL_CODE

public static final int **POSTAL_CODE**

Address key referencing the postal code.
See Also:
Constant Field Values

COUNTY

public static final int **COUNTY**

Address key referencing the county.
See Also:
Constant Field Values

STATE

public static final int **STATE**

Address key referencing the state.
See Also:
Constant Field Values

BUILDING_NAME

public static final int **BUILDING_NAME**

Address key referencing the building name.
See Also:
Constant Field Values

BUILDING_FLOOR

public static final int **BUILDING_FLOOR**

Address key referencing the building floor.
See Also:
Constant Field Values

BUILDING_ROOM

public static final int **BUILDING_ROOM**

Address key referencing the building room.
See Also:
Constant Field Values

6.4 Package org.osgi.nursery.util.geo

Interface Summary

Zone	Defines a geographical area.

Class Summary

CircularZone	A circle is defined by a center expressed by a coordinate and a radius expressed in meter.
Coordinate	Defines a WGS84 coordinate with latitude, longitude and altitude.
RectangularZone	A Box is a rectangular area defined by two coordinates: the top left coordinate and the bottom right coordinate.

6.4.1 Interface Zone

All Known Implementing Classes:

CircularZone, RectangularZone

---

public interface **Zone**

Defines a geographical area.

Method Summary

boolean	**contains**(Coordinate coordinate) Indicates if the coordinate is contained in the this object.
boolean	**contains**(Zone zone) Indicates if the Shape object given as parameter is in this object.

Method Detail

contains

public boolean **contains**(Coordinate coordinate)

Indicates if the coordinate is contained in the this object.

Parameters:

coordinate - The coordinate that need to be checked

Returns:

true if the coordinate is inside the object, otherwise false

---

contains

All Page Within This Box

public boolean **contains**(Zone zone)

Indicates if the Shape object given as parameter is in this object.
Parameters:

zone -

Returns:
true if the shape is completely inside the object, otherwise false

6.4.2 Class CircularZone

public class **CircularZone**
extends java.lang.Object
implements Zone

A circle is defined by a center expressed by a coordinate and a radius expressed in meter.

Constructor Summary

CircularZone(Coordinate center, org.osgi.util.measurement.Measurement radius) Create a new Circle object.	

Method Summary

boolean	**contains**(Coordinate coordinate) Indicates if the coordinate is contained in the this object.
boolean	**contains**(Zone zone) Indicates if the area is contained in this object.
Coordinate	**getCenter**() Returns the center of the circle expressed by a Coordinate object.
org.osgi.util.measurement.Measurement	**getRadius**() Returns the radius of the circle expressed in meter.

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

All Page Within This Box

CircularZone

public **CircularZone**(Coordinate center,
org.osgi.util.measurement.Measurement radius)
Create a new Circle object.

Parameters:

center - The center of the circle.

radius - The radius of the circle expressed in meter.

Method Detail

getRadius

public org.osgi.util.measurement.Measurement **getRadius**()
Returns the radius of the circle expressed in meter.
Returns:
Measurement

---

getCenter

public Coordinate **getCenter**()
Returns the center of the circle expressed by a Coordinate object.
Returns:
org.osgi.nursery.util.geo.Coordinate

---

contains

public boolean **contains**(Coordinate coordinate)
Indicates if the coordinate is contained in the this object.
Specified by:
`contains` in interface `Zone`
Parameters:

coordinate - The coordinate that need to be checked

Returns:
boolean

---

contains

public boolean **contains**(Zone zone)
Indicates if the area is contained in this object.
Specified by:
`contains` in interface `Zone`
Parameters:

zone -

Returns:
boolean

All Page Within This Box

6.4.3 Class Coordinate

public class **Coordinate**
extends java.lang.Object

Defines a WGS84 coordinate with latitude, longitude and altitude.

Constructor Summary

Coordinate(Coordinate coord) Create a new Coordinate object from another Coordinate object.
Coordinate(double latitude, double longitude, float altitude) Create a new Coordinate object.
Coordinate(org.osgi.util.position.Position pos) Create a new Coordinate object from a Position object.

Method Summary

float	**getAltitude**() Returns the altitude of this coordinate in meters.
double	**getLatitude**() Returns the latitude of this coordinate in radians.
double	**getLongitude**() Returns the longitude of this coordinate in radians.

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

Coordinate

public **Coordinate**(org.osgi.util.position.Position pos)

Create a new Coordinate object from a Position object.

Coordinate

public **Coordinate**(Coordinate coord)

Create a new Coordinate object from another Coordinate object.

Parameters:

coord - a Coordinate object where values are copied from.

Coordinate

public **Coordinate**(double latitude,
double longitude,
float altitude)

Create a new Coordinate object.

Parameters:

latitude - a double value specifying the latitude in radians

longitude - a double value specifying the longitude in radians

altitude - a float value specifying the altitude in meters

Method Detail

getLatitude

public double **getLatitude**()

Returns the latitude of this coordinate in radians.
Returns:
double

---

getLongitude

public double **getLongitude**()

Returns the longitude of this coordinate in radians.
Returns:
double

---

getAltitude

public float **getAltitude**()

Returns the altitude of this coordinate in meters.
Returns:
float

6.4.4 Class RectangularZone

---

public class **RectangularZone**
extends java.lang.Object
implements Zone

A Box is a rectangular area defined by two coordinates: the top left coordinate and the bottom right coordinate.

Constructor Summary

RectangularZone(Coordinate topLeft, Coordinate bottomRight) Create a new Box object.	

Method Summary

boolean	**contains**(Coordinate coordinate) Indicates if the coordinate is in the box.
boolean	**contains**(Zone zone) Indicates if the Shape is in the box.
Coordinate	**getBottomRight**() Returns the bottom right coordinate of the box.
Coordinate	**getTopLeft**() Returns the top left coordinate of the box.

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

RectangularZone

public **RectangularZone**(Coordinate topLeft,
Coordinate bottomRight)
Create a new Box object.

Parameters:

topLeft - The top left coordinate of the box.

bottomRight - The bottom right coordinate of the box.

Method Detail

getTopLeft

All Page Within This Box

public Coordinate **getTopLeft**()

Returns the top left coordinate of the box.
Returns:
org.osgi.nursery.util.geo.Coordinate

---

getBottomRight

public Coordinate **getBottomRight**()

Returns the bottom right coordinate of the box.
Returns:
org.osgi.nursery.util.geo.Coordinate

---

contains

public boolean **contains**(Coordinate coordinate)

Indicates if the coordinate is in the box.
Specified by:
`contains` in interface `Zone`
Parameters:
coordinate - The coordinate that need to be checked
Returns:
boolean

---

contains

public boolean **contains**(Zone zone)

Indicates if the Shape is in the box.
Specified by:
`contains` in interface `Zone`
Parameters:
zone -
Returns:
boolean

6.5 Package org.osgi.nursery.util.location

6.5.1 Interface Location

Interface to a location entity. A location can contain: a coordinate, an address and an area.

Field Summary

static int	**ADDRESS_FIRST** Indicates that the address has been entered first and it is the relevant part of the location.
static int	**COORDINATE_FIRST** Indicates that the coordinate has been entered first and it is the relevant part of the location
static int	**ZONE_FIRST** Indicates that the zone has been entered first and it is the relevant part of the location

Method Summary

Address	**getAddress**() Returns the address of this location or null if the address is not found.
Coordinate	**getCoordinate**() Returns the WGS84 coordinate of this location or null if the coordinate is not known.
int	**getType**() Indicates the most relevant part of the address: ADDRESS_FIRST or COORDINATE_FIRST.
Zone	**getZone**() Returns the shape of this location or null if the shape is not known.

Field Detail

6.5.2 ADDRESS_FIRST

public static final int **ADDRESS_FIRST**

Indicates that the address has been entered first and it is the relevant part of the location.
See Also:
Constant Field Values

6.5.3 COORDINATE_FIRST

public static final int **COORDINATE_FIRST**

Indicates that the coordinate has been entered first and it is the relevant part of the location
See Also:
Constant Field Values

All Page Within This Box

6.5.4 ZONE_FIRST

public static final int **ZONE_FIRST**

Indicates that the zone has been entered first and it is the relevant part of the location
See Also:
Constant Field Values

Method Detail

6.5.5 getAddress

public Address **getAddress**()

Returns the address of this location or null if the address is not found.
Returns:
The address at this location, null if the address is not found

6.5.6 getCoordinate

public Coordinate **getCoordinate**()

Returns the WGS84 coordinate of this location or null if the coordinate is not known.
Returns:
The WGS84 coordinate of this location, null if the coordinate is not known.

6.5.7 getType

public int **getType**()

Indicates the most relevant part of the address: ADDRESS_FIRST or COORDINATE_FIRST. This information is important to resolve the address against the map database.
Returns:
int The address type.

6.5.8 getZone

public Zone **getZone**()

Returns the shape of this location or null if the shape is not known.
Returns:
The shape of this location, null if it is not known.

All Page Within This Box

6.6 Package org.osgi.nursery.util.maneuver

Interface Summary	
ActionType	Contains all possible values for an action type in a maneuver.
JunctionGeometry	A JunctionGeometry is the physical description of the junction that the driver is about to pass by performing the maneuver.
JunctionSegment	A JunctionSegment is a part of the JunctionGeometry.
JunctionType	Contains all possible junction type.
Maneuver	A maneuver is the action that the driver has to perform at a junction.

6.6.1 Interface ActionType

public interface **ActionType**

Contains all possible values for an action type in a maneuver.

Field Summary	
static int	**ARRIVAL** Indicates that the driver will arrive exaclty at the destination at the next maneuver.
static int	**ARRIVAL_NEIGHBOURHOOD** Indicates that the driver will arrive near the destination the next maneuver.
static int	**ARRIVAL_OFF_ROAD** Indicates that the driver will arrive near the destination at the next maneuver but the destination is off-road.
static int	**CONTINUE** Indicates that the driver has to continue on the same direction.
static int	**NO_ACTION** Indicates that there is no action at this maneuver.
static int	**TURN** Indicates that the action is a turn.

Field Detail

NO_ACTION

public static final int **NO_ACTION**

Indicates that there is no action at this maneuver.

See Also:

Constant Field Values

All Page Within This Box

TURN

public static final int **TURN**

Indicates that the action is a turn. The direction of turn is indicated by the turn angle. It is up to the client to decide the right advice.
See Also:
Constant Field Values

CONTINUE

public static final int **CONTINUE**

Indicates that the driver has to continue on the same direction.
See Also:
Constant Field Values

ARRIVAL

public static final int **ARRIVAL**

Indicates that the driver will arrive exaclty at the destination at the next maneuver.
See Also:
Constant Field Values

ARRIVAL_NEIGHBOURHOOD

public static final int **ARRIVAL_NEIGHBOURHOOD**

Indicates that the driver will arrive near the destination the next maneuver.
See Also:
Constant Field Values

ARRIVAL_OFF_ROAD

public static final int **ARRIVAL_OFF_ROAD**

Indicates that the driver will arrive near the destination at the next maneuver but the destination is off-road.
See Also:
Constant Field Values

6.6.2 Interface JunctionGeometry

---

public interface **JunctionGeometry**

A JunctionGeometry is the physical description of the junction that the driver is about to pass by performing the maneuver.

Method Summary

All Page Within This Box

JunctionSegment	**getFromSegment**() Returns the segment from where the traveler is entering the maneuver
JunctionSegment[]	**getInnerSegment**() Returns the list of segments that are inside the maneuver geometry.
JunctionSegment[]	**getOuterSegment**() Returns the list of segments that are outside the maneuver geometry.
JunctionSegment	**getToSegment**() Returns the segment where the traveler is leaving the maneuver.

Method Detail

getInnerSegment

public JunctionSegment[] **getInnerSegment**()

Returns the list of segments that are inside the maneuver geometry.
Returns:
org.osgi.nursery.util.maneuver.JunctionSegment[]

---

getOuterSegment

public JunctionSegment[] **getOuterSegment**()

Returns the list of segments that are outside the maneuver geometry.
Returns:
org.osgi.nursery.util.maneuver.JunctionSegment[]

---

getFromSegment

public JunctionSegment **getFromSegment**()

Returns the segment from where the traveler is entering the maneuver
Returns:
org.osgi.nursery.util.maneuver.JunctionSegment

---

getToSegment

public JunctionSegment **getToSegment**()

Returns the segment where the traveler is leaving the maneuver.
Returns:
org.osgi.nursery.util.maneuver.JunctionSegment

6.6.3 Interface JunctionSegment

---

public interface **JunctionSegment**

All Page Within This Box

A JunctionSegment is a part of the JunctionGeometry.

Method Summary

org.osgi.util.measurement.Measurement	**getAngle**() Returns the angle of this segment with the North in degrees.
Coordinate	**getCoordinate**() Returns the coordinate where this segment is attached to the maneuver.
boolean	**isInnerSegment**() Returns true if this JunctionSegment object is part of the inner segments otherwise false.
boolean	**isOnRoute**() Returns true if this JunctionSegment object is in the itinerary otherwise false.

Method Detail

getAngle

public org.osgi.util.measurement.Measurement **getAngle**()

Returns the angle of this segment with the North in degrees.
Returns:
Measurement

---

getCoordinate

public Coordinate **getCoordinate**()

Returns the coordinate where this segment is attached to the maneuver.
Returns:
org.osgi.nursery.util.geo.Coordinate

---

isInnerSegment

public boolean **isInnerSegment**()

Returns true if this JunctionSegment object is part of the inner segments otherwise false.
Returns:
boolean

---

isOnRoute

public boolean **isOnRoute**()

Returns true if this JunctionSegment object is in the itinerary otherwise false.
Returns:

All Page Within This Box

Boolean

6.6.4 Interface JunctionType

public interface **JunctionType**

Contains all possible junction type.

Field Summary

static int	**ENCLOSED_TRAFFIC_AREA** Indicates that there is a enclosed traffic area at the maneuver.
static int	**FACILITY_ENTRY** Indicates that the driver will access a facility at the next maneuver.
static int	**FACILITY_EXIT** Indicates that the driver will exit from a facility at the next maneuver.
static int	**INTERSECTION** Indicates that there is normal intersection at the maneuver.
static int	**MAINROAD_ENTRY** Indicates that the driver will access a main road at the next maneuver.
static int	**MAINROAD_EXIT** Indicates that the driver will exit from a main road at the next maneuver.
static int	**MOTORWAY_ENTRY** Indicates that the driver will access a motorway at the next maneuver.
static int	**MOTORWAY_EXIT** Indicates that the driver will xit from a motorway at the next maneuver.
static int	**NONE** Indicates that there is no junction at the maneuver.
static int	**RESTRICTED_AREA_ENTRY** Indicates that the driver will access a restricted area at the next maneuver.
static int	**RESTRICTED_AREA_EXIT** Indicates that the driver will exit a restricted area at the next maneuver.
static int	**ROUNDABOUT** Indicates that there is a roundabout at the maneuver.
static int	**SQUARE** Indicates that there is a square at the maneuver.

Field Detail

NONE

public static final int **NONE**

Indicates that there is no junction at the maneuver.

All Page Within This Box

See Also:
Constant Field Values

SQUARE

public static final int **SQUARE**

Indicates that there is a square at the maneuver.
See Also:
Constant Field Values

ENCLOSED_TRAFFIC_AREA

public static final int **ENCLOSED_TRAFFIC_AREA**

Indicates that there is a enclosed traffic area at the maneuver. An enclosed traffic area is a confined area within which unstructured traffic movements are allowed.
See Also:
Constant Field Values

INTERSECTION

public static final int **INTERSECTION**

Indicates that there is normal intersection at the maneuver.
See Also:
Constant Field Values

ROUNDABOUT

public static final int **ROUNDABOUT**

Indicates that there is a roundabout at the maneuver.
See Also:
Constant Field Values

MOTORWAY_ENTRY

public static final int **MOTORWAY_ENTRY**

Indicates that the driver will access a motorway at the next maneuver.
See Also:
Constant Field Values

MOTORWAY_EXIT

public static final int **MOTORWAY_EXIT**

Indicates that the driver will xit from a motorway at the next maneuver.
See Also:
Constant Field Values

MAINROAD_ENTRY

public static final int **MAINROAD_ENTRY**

Indicates that the driver will access a main road at the next maneuver.

See Also:
Constant Field Values

MAINROAD_EXIT

public static final int **MAINROAD_EXIT**

Indicates that the driver will exit from a main road at the next maneuver.
See Also:
Constant Field Values

RESTRICTED_AREA_ENTRY

public static final int **RESTRICTED_AREA_ENTRY**

Indicates that the driver will access a restricted area at the next maneuver.
See Also:
Constant Field Values

RESTRICTED_AREA_EXIT

public static final int **RESTRICTED_AREA_EXIT**

Indicates that the driver will exit a restricted area at the next maneuver.
See Also:
Constant Field Values

FACILITY_ENTRY

public static final int **FACILITY_ENTRY**

Indicates that the driver will access a facility at the next maneuver.
See Also:
Constant Field Values

FACILITY_EXIT

public static final int **FACILITY_EXIT**

Indicates that the driver will exit from a facility at the next maneuver.
See Also:
Constant Field Values

6.6.5 Interface Maneuver

---

public interface **Maneuver**
extends Hint

A maneuver is the action that the driver has to perform at a junction. Several advices can be rendered at a maneuver.

All Page Within This Box

Method Summary

int	**getActionType**() Returns the action performed at the maneuver.
JunctionGeometry	**getGeometry**() Returns the junction geometry defined by the JunctionGeometry class.
int	**getJunctionType**() Returns the junction type where the maneuver will be performed.
int	**getNumberOfExit**() Returns the number of the segment by which the planned route is leaving the junction.
java.lang.String	**getRoadSign**() Returns the road sign information towards the itinerary.
int	**getTurnAngle**() Returns the direction of turn if the action of the maneuver is a turn.
boolean	**isMultipleTurn**() In case of turn action at a maneuver, this flag indicates if there several possibility (roads) to turn.

Method Detail

getJunctionType

public int **getJunctionType**()

Returns the junction type where the maneuver will be performed. The value must be one of the value defined in JunctionType interface.

Returns:

int

---

getTurnAngle

public int **getTurnAngle**()

Returns the direction of turn if the action of the maneuver is a turn. The value must be one of the value defined by DirectionOfTurn interface.

Returns:

int

---

getActionType

public int **getActionType**()

Returns the action performed at the maneuver. The value must be one of the value defined by ActionType interface.

Returns:

All Page Within This Box

int

---

getRoadSign

public java.lang.String **getRoadSign**()

Returns the road sign information towards the itinerary. It is usuallly a city / municipality names.
Returns:
String

---

getNumberOfExit

public int **getNumberOfExit**()

Returns the number of the segment by which the planned route is leaving the junction. The numbering starts at the from-segment (number 0). For right-turning possibilities, numbering proceeds in counter-clockwise order. For left-turning possibilities, numbering proceeds in clockwise order.
Returns:
int

---

isMultipleTurn

public boolean **isMultipleTurn**()

In case of turn action at a maneuver, this flag indicates if there several possibility (roads) to turn. This information is important to render advice like "At the next junction take the xxth on the left/right".
Returns:
boolean

---

getGeometry

public JunctionGeometry **getGeometry**()

Returns the junction geometry defined by the JunctionGeometry class. This geometry is used to render one or several detailed advices.
Returns:
org.osgi.nursery.util.maneuver.JunctionGeometry

6.7 Package org.osgi.nursery.util.poi

Interface Summary

PointOfInterest	A point of interest represents external service provider, like a hospital, a gas station, a theatre, a hotel, etc

6.7.1 Interface PointOfInterest

public interface **PointOfInterest**

A point of interest represents external service provider, like a hospital, a gas station, a theatre, a hotel, etc. It consists of a name, a brand, an address, a type, and a location. Optionally a point of interest contains an advice: a visual representation (e.g. HTML page) and an audible representation (e.g. mpeg file). It can contain as well a phone number, an influence zone where the advice can be rendered. Points of interest are further separated in national and local POI. National (or country specific) POI means of "national" importance such as international airports, Disney Land, London Tower, etc.

Method Summary

java.lang.String	**getCategory**() Returns the POI Category.
Location[]	**getEntryPoints**() Returns the location (WGS84 coordinates) of possible entries to the POI.
Location	**getLocation**() Returns the location of the POI.
java.lang.String	**getName**() Returns the POI name.
java.util.Dictionary	**getProperties**() Returns additional properties of the POI.

Method Detail

getName

public java.lang.String **getName**()
Returns the POI name.

All Page Within This Box

Returns:
The name of the POI

---

getLocation

public Location **getLocation**()

Returns the location of the POI.
Returns:
The location of the POI

---

getEntryPoints

public Location[] **getEntryPoints**()

Returns the location (WGS84 coordinates) of possible entries to the POI. These location can be used to determine a route.
Returns:
The list of entry points to the POI

---

getProperties

public java.util.Dictionary **getProperties**()

Returns additional properties of the POI.
Returns:
The list of specific properties of the POI, null if properties are not provided

---

getCategory

public java.lang.String **getCategory**()

Returns the POI Category. There is no recommended classification. The third party provider has to deliver the categories it can handled.
Returns:
String

6.8 Package org.osgi.nursery.util.route

Interface Summary	
FormOfWay	Defines form of way values according to GDF standard.
Hint	
RoadClass	Defines road class values according to GDF standard.
Route	Interface to the route definition.
RouteBoundary	

Class Summary	
FeaturePreference	Defines a feature that the user would like to include or exclude.
RouteElement	
RoutePlan	Defines the complete route plan that the user would like to realise.

6.8.1 Interface FormOfWay

public interface **FormOfWay**

Defines form of way values according to GDF standard.

Field Summary	
static int	**ENCLOSED_TRAFFIC_AREA** Indicates that the road segment is part of an Enclosed Traffic Area.
static int	**ENTRANCE_EXIT_CAR_PARK** Indicates that the road segment is an entrance to or an exit of Car Park.
static int	**ENTRANCE_EXIT_SERVICE** Indicates that the road segment is an entrance to or an exit to Service.
static int	**MOTORWAY** Indicates that the road segment is part of a Motorway.
static int	**MULTIPLE_CARRIAGEWAY** Indicates that the road segment is part of a Multiple Carriageway which is not a motorway.
static int	**PEDESTRIAN_ZONE** Indicates that the road segment is part of a Pedestrian Zone.
static int	**ROUNDABOUT** Indicates that the road segment is part of a Roundabout.

All Page Within This Box

static int	**SERVICE_ROAD** Indicates that the road segment is part of a Service Road.
static int	**SINGLE_CARRIAGEWAY** Indicates that the road segment is part of a Single Carriageway.
static int	**SLIP_ROAD** Indicates that the road segment is part of a Slip Road.
static int	**SQUARE** Indicates that the road segment is part of a Traffic Square.

Field Detail

MOTORWAY

public static final int **MOTORWAY**

Indicates that the road segment is part of a Motorway.
See Also:
Constant Field Values

MULTIPLE_CARRIAGEWAY

public static final int **MULTIPLE_CARRIAGEWAY**

Indicates that the road segment is part of a Multiple Carriageway which is not a motorway.
See Also:
Constant Field Values

SINGLE_CARRIAGEWAY

public static final int **SINGLE_CARRIAGEWAY**

Indicates that the road segment is part of a Single Carriageway.
See Also:
Constant Field Values

ROUNDABOUT

public static final int **ROUNDABOUT**

Indicates that the road segment is part of a Roundabout.
See Also:
Constant Field Values

SQUARE

public static final int **SQUARE**

Indicates that the road segment is part of a Traffic Square.
See Also:
Constant Field Values

ENCLOSED_TRAFFIC_AREA

public static final int **ENCLOSED_TRAFFIC_AREA**

Indicates that the road segment is part of an Enclosed Traffic Area.
See Also:
Constant Field Values

SLIP_ROAD

public static final int **SLIP_ROAD**

Indicates that the road segment is part of a Slip Road.
See Also:
Constant Field Values

SERVICE_ROAD

public static final int **SERVICE_ROAD**

Indicates that the road segment is part of a Service Road.
See Also:
Constant Field Values

ENTRANCE_EXIT_CAR_PARK

public static final int **ENTRANCE_EXIT_CAR_PARK**

Indicates that the road segment is an entrance to or an exit of Car Park.
See Also:
Constant Field Values

ENTRANCE_EXIT_SERVICE

public static final int **ENTRANCE_EXIT_SERVICE**

Indicates that the road segment is an entrance to or an exit to Service.
See Also:
Constant Field Values

PEDESTRIAN_ZONE

public static final int **PEDESTRIAN_ZONE**

Indicates that the road segment is part of a Pedestrian Zone.
See Also:
Constant Field Values

6.8.2 Interface RoadClass

---

public interface **RoadClass**

Defines road class values according to GDF standard.

All Page Within This Box

Field Summary

static int	**EIGHT_CLASS** Indicates that the road segment is a eight class road.
static int	**FIFTH_CLASS** Indicates that the road segment is a fitfh class road.
static int	**FIRST_CLASS** Indicates that the road segment is a first class road.
static int	**FOURTH_CLASS** Indicates that the road segment is a fourth class road.
static int	**MAIN_ROAD** Indicates that the road segment is a main road.
static int	**NINTH_CLASS** Indicates that the road segment is a ninth class road.
static int	**SECOND_CLASS** Indicates that the road segment is a second class road.
static int	**SEVENTH_CLASS** Indicates that the road segment is a seventh class road.
static int	**SIXTH_CLASS** Indicates that the road segment is a sixth class road.
static int	**THIRD_CLASS** Indicates that the road segment is a third class road.

Field Detail

MAIN_ROAD

public static final int **MAIN_ROAD**

Indicates that the road segment is a main road.
See Also:
Constant Field Values

FIRST_CLASS

public static final int **FIRST_CLASS**

Indicates that the road segment is a first class road.
See Also:
Constant Field Values

SECOND_CLASS

public static final int **SECOND_CLASS**

Indicates that the road segment is a second class road.
See Also:
Constant Field Values

All Page Within This Box

THIRD_CLASS

public static final int **THIRD_CLASS**

Indicates that the road segment is a third class road.
See Also:
Constant Field Values

FOURTH_CLASS

public static final int **FOURTH_CLASS**

Indicates that the road segment is a fourth class road.
See Also:
Constant Field Values

FIFTH_CLASS

public static final int **FIFTH_CLASS**

Indicates that the road segment is a fitfh class road.
See Also:
Constant Field Values

SIXTH_CLASS

public static final int **SIXTH_CLASS**

Indicates that the road segment is a sixth class road.
See Also:
Constant Field Values

SEVENTH_CLASS

public static final int **SEVENTH_CLASS**

Indicates that the road segment is a seventh class road.
See Also:
Constant Field Values

EIGHT_CLASS

public static final int **EIGHT_CLASS**

Indicates that the road segment is a eight class road.
See Also:
Constant Field Values

NINTH_CLASS

public static final int **NINTH_CLASS**

Indicates that the road segment is a ninth class road.
See Also:
Constant Field Values

6.8.3 Interface Route

Interface to the route definition. The route has several states:

Method Summary

Location[]	**getLocations**()
RouteElement[]	**getRouteElements**()
RouteElement[]	**getRouteElements**(java.lang.String filter) Returns the list of Route Elements that match to the filter.
RoutePlan	**getRoutePlan**()

Method Detail

getRouteElements

public RouteElement[] **getRouteElements**()

Returns:
org.osgi.nursery.util.route.RouteElement[]

---

getRouteElements

public RouteElement[] **getRouteElements**(java.lang.String filter)

Returns the list of Route Elements that match to the filter.

Parameters:

filter -

Returns:
org.osgi.nursery.util.route.RouteElement[]

---

getRoutePlan

public RoutePlan **getRoutePlan**()

Returns:
org.osgi.nursery.util.route.RoutePlan

---

getLocations

public Location[] **getLocations**()

Returns:
org.osgi.nursery.util.location.Location[]

All Page Within This Box

6.8.4 Class FeaturePreference

public class **FeaturePreference**
extends java.lang.Object

Defines a feature that the user would like to include or exclude. A feature can be set only once.

If a user exclude Highway then we must not include them afterwards.

Field Summary

static FeaturePreference	**FERRY** FeaturePreference object to set FERRY preference.
static FeaturePreference	**HIGHWAY** FeaturePreference object to set HIGHWAY preference.
static FeaturePreference	**TOLLWAY** FeaturePreference object to set TOLLWAY preference.

Method Summary

boolean	**isExcluded**() Returns true if this FeaturePreference object is excluded otherwise false.
void	**setExcluded**(boolean excluded) Exclude this FeaturePreference object.

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Field Detail

HIGHWAY

public static final FeaturePreference **HIGHWAY**
FeaturePreference object to set HIGHWAY preference.

TOLLWAY

public static final FeaturePreference **TOLLWAY**

All Page Within This Box

FeaturePreference object to set TOLLWAY preference.

FERRY

public static final FeaturePreference **FERRY**

FeaturePreference object to set FERRY preference.

Method Detail

setExcluded

public void **setExcluded**(boolean excluded)

Exclude this FeaturePreference object.

Parameters:

excluded -

---

isExcluded

public boolean **isExcluded**()

Returns true if this FeaturePreference object is excluded otherwise false.

Returns:

boolean

6.8.5 Class RouteElement

---

public class **RouteElement**
extends java.lang.Object
implements Location

Field Summary

Fields inherited from interface org.osgi.nursery.util.location.Location
ADDRESS_FIRST, COORDINATE_FIRST

Constructor Summary

RouteElement(Coordinate coord)

Method Summary

All Page Within This Box

Address	**getAddress**() Returns the address of this location or null if the address is not found.
Coordinate	**getCoordinate**() Returns the WGS84 coordinate of this location or null if the coordinate is not known.
FormOfWay	**getFormOfWay**()
org.osgi.util.measurement.Measurement	**getHeading**()
Maneuver	**getManeuver**()
RoadClass	**getRoadClass**()
int	**getType**() Indicates the most relevant part of the address: ADDRESS_FIRST or COORDINATE_FIRST.
Zone	**getZone**() Returns the shape of this location or null if the shape is not known.

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

RouteElement

public **RouteElement**(Coordinate coord)

Method Detail

getHeading

public org.osgi.util.measurement.Measurement **getHeading**()

Returns:
Measurement

All Page Within This Box

getFormOfWay

public FormOfWay **getFormOfWay**()

Returns:
org.osgi.nursery.util.route.FormOfWay

---

getRoadClass

public RoadClass **getRoadClass**()

Returns:
org.osgi.nursery.util.route.RoadClass

---

getManeuver

public Maneuver **getManeuver**()

Returns:
org.osgi.nursery.util.maneuver.Maneuver

---

getAddress

public Address **getAddress**()

Description copied from interface: Location
Returns the address of this location or null if the address is not found.
Specified by:
getAddress in interface Location
Returns:
See Also:
Location.getAddress()

---

getCoordinate

public Coordinate **getCoordinate**()

Description copied from interface: Location
Returns the WGS84 coordinate of this location or null if the coordinate is not known.
Specified by:
getCoordinate in interface Location
Returns:
See Also:
Location.getCoordinate()

---

getType

public int **getType**()

Description copied from interface: Location
Indicates the most relevant part of the address: ADDRESS_FIRST or COORDINATE_FIRST. This information is important to resolve the address against the map database.
Specified by:
getType in interface Location
Returns:
See Also:

All Page Within This Box

Location.getType()

---

getZone

public Zone **getZone**()

Description copied from interface: Location
Returns the shape of this location or null if the shape is not known.
Specified by:
getZone in interface Location
Returns:
See Also:
Location.getZone()

6.8.6 Class RoutePlan

---

public class **RoutePlan**
extends java.lang.Object

Defines the complete route plan that the user would like to realise. This plan can be associated to the complete Journey or to a specific via-point. If so then it overrides global navigation plan (if any).

Field Summary

static int	**FASTEST** Indicates that the fastest way must be found.
static int	**SHORTEST** Indicates that the shortest way must be found.

Constructor Summary

RoutePlan(int navPref,	FeaturePreference[] featurePrefs,	Location[] includedLocs,
Location[] excludedLocs)		

Method Summary

Location[]	**getExcludedLocations**() Returns the list of LocationPreference objects.

All Page Within This Box

FeaturePreference[]	**getFeaturePreferences**() Returns the list of FeaturePreference objects.
Location[]	**getIncludedLocations**() Returns the list of LocationPreference objects.
int	**getNavigationPreference**() Returns the navigation preference set for this NavigationPlan object.

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Field Detail

SHORTEST

public static final int **SHORTEST**

Indicates that the shortest way must be found.
See Also:
Constant Field Values

FASTEST

public static final int **FASTEST**

Indicates that the fastest way must be found.
See Also:
Constant Field Values

Constructor Detail

RoutePlan

public **RoutePlan**(int navPref,

FeaturePreference[] featurePrefs,

Location[] includedLocs,

Location[] excludedLocs)

Parameters:

navPref -

featurePrefs -

includedLocs -

excludedLocs –

Method Detail

getNavigationPreference

public int **getNavigationPreference**()

Returns the navigation preference set for this NavigationPlan object. The value is one of the value defined by this class.

Returns:

int

---

getIncludedLocations

public Location[] **getIncludedLocations**()

Returns the list of LocationPreference objects.

Returns:

org.osgi.nursery.util.location.Location[]

---

getExcludedLocations

public Location[] **getExcludedLocations**()

Returns the list of LocationPreference objects.

Returns:

org.osgi.nursery.util.location.Location[]

---

getFeaturePreferences

public FeaturePreference[] **getFeaturePreferences**()

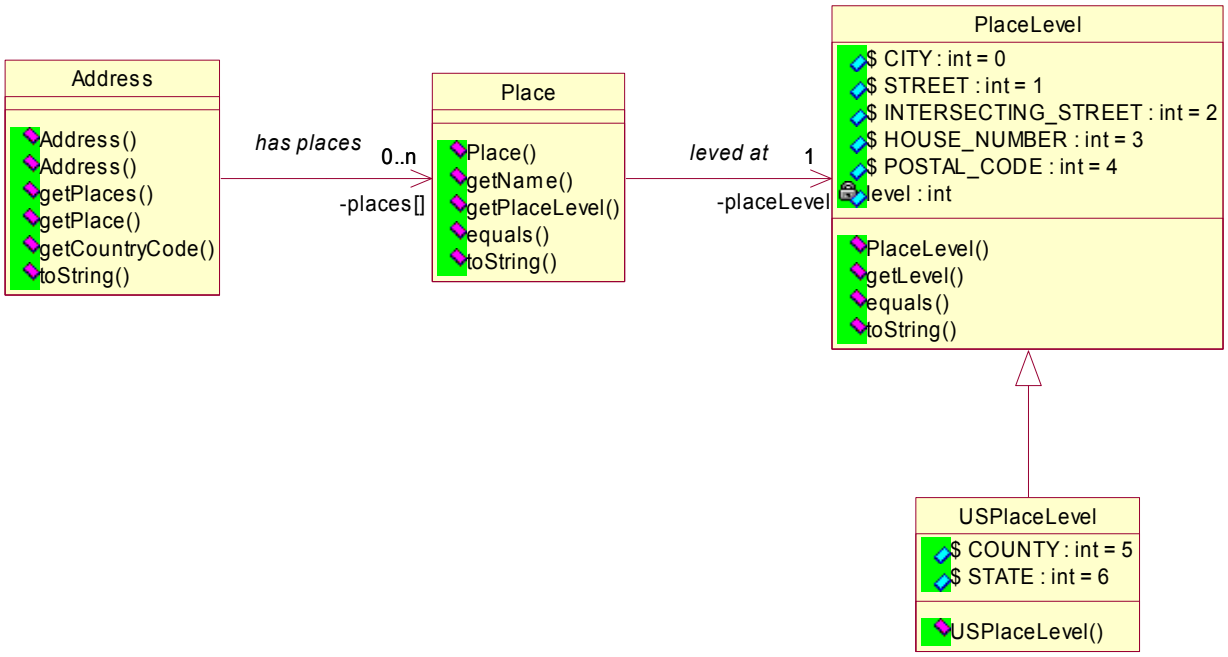
Returns the list of FeaturePreference objects.

Returns:

org.osgi.nursery.util.route.FeaturePreference[]

7 Considered Alternatives

7.1 Address [Siemens VDO Automotive]

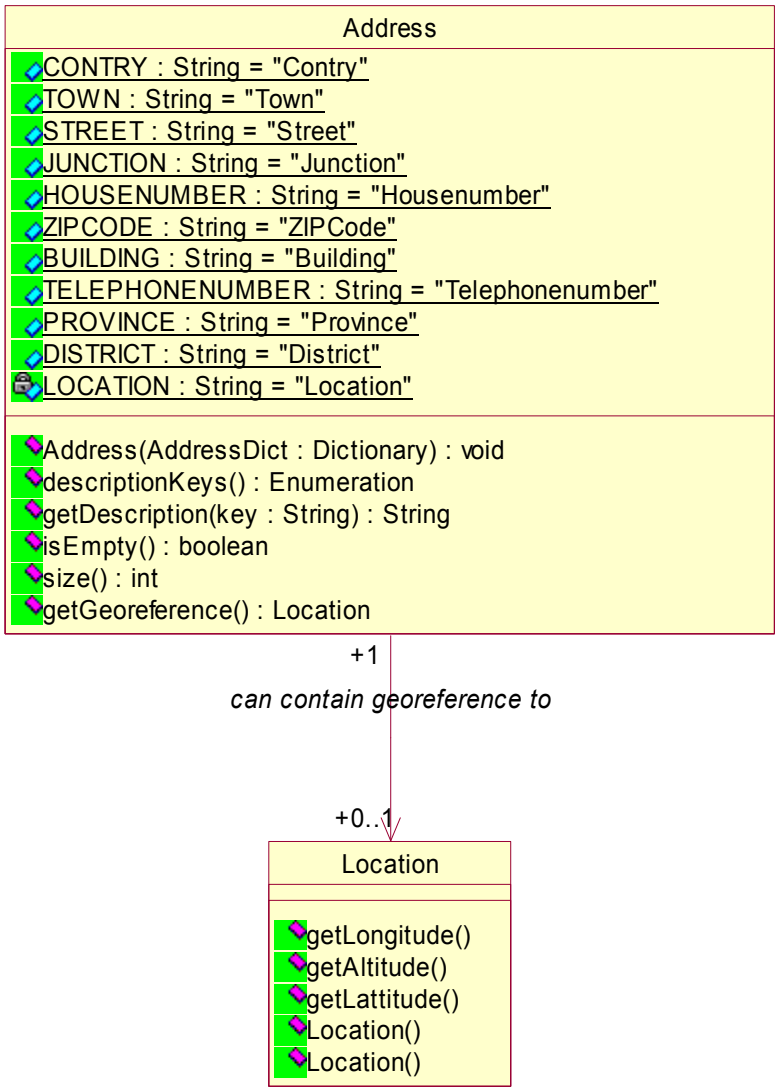


The definition of an address varies from one country to another. These differences of format do not allow having a well-defined model.

To allow this flexibility we have defined each part of the address as a place. A hierarchy of places can be defined for each Country. This model provides a basic hierarchy the PlaceLevel class but this list can be extended (e.g. for US).

The Country (defined by the country code) is the key that determines the place hierarchy.

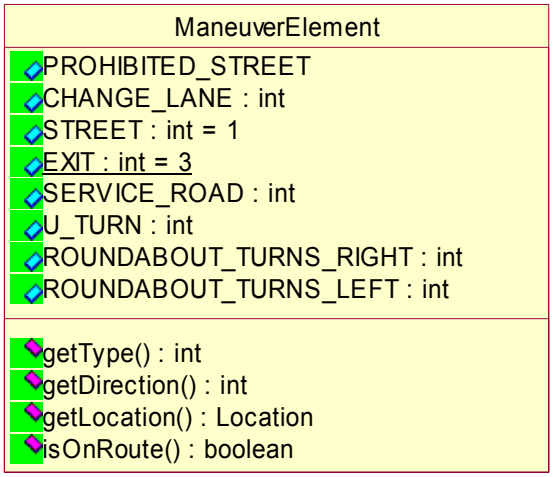
7.2 Address [BOSCH]



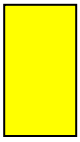
The `Address` is made of a list of description items. This list is in fact a dictionary with Key/Value pairs. A basic set of keys are defined as constants in the Address interface. This list can be extended by am implementation.

The Address is created with a Dictionary filled with the address description key-value pairs. The Address interface has for reading the same methods as the Dictionary interface but to be immutable all writing and modifying methods are left away. It also provides a method to ask for a geo reference of the address. This method returns the navigation system interpretation of the address if available.

7.3 Maneuver [BOSCH]

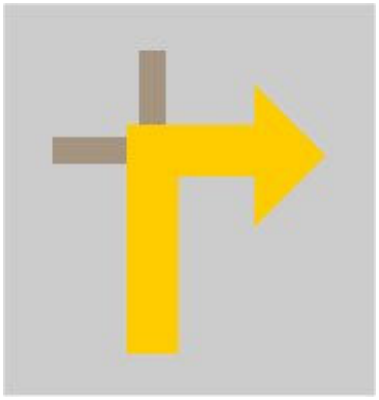


A Maneuver is described by an array of ManeuverElements. Each ManeuverElement class describes one element of the complete maneuver information which can be visualized as an arrow. For example a crossing where the driver should turn right would consist from the following elements:

ID	Direction	isOnRoute	Symbol
STREET	0	true	
STREET	90	false	
STREET	0	false	
STREET	270	true	

The complete arrow would look like this:

All Page Within This Box



8 Security Considerations

TO BE DEFINED

9 Document Support

9.1 References

[1]. Bradner, S., Key words for use in RFCs to Indicate Requirement Levels, RFC2119, March 1997.

[2]. Software Requirements & Specifications. Michael Jackson. ISBN 0-201-87712-0

[3]. RFP Navigation Domain Model, Olivier Pavé, March 2003

9.2 Author's Address

Name	Pavé Olivier
Company	Siemens VDO Automotive
Address	Batiment Alpha 80, route des lucioles – BP 305 06906 Sophia-Antipolis Cedex, France
Voice	+33 (0)4-9238-1129
e-mail	olivier.pave@siemens.com

Name	Rob van den Berg
Company	Siemens VDO Automotive

All Page Within This Box

Address	Luchthavenweg 48 5657 EB Eindhoven, Netherlands
Voice	+31 (40) 844-4859
e-mail	rob.vandenberg@siemens.com

Name	Wolfgang Kopmann
Company	Robert Bosch GmbH
Address	Postfach 777777 31134 Hildesheim Germany
Voice	*49 (5121) 49 4023
e-mail	wolfgang.kopmann@de.bosch.com

9.3 Acronyms and Abbreviations

9.4 End of Document

All Page Within This Box